The Workshop Programme

9:15–9:30	Opening: Aims of the workshop
9:30–10:00	*Relational evaluation schemes* Ted Briscoe, John Carroll, Jonathan Graham, Ann Copestake
10:00–10:30	*Towards a dependency-oriented evaluation for partial parsing* Sandra Kübler, Heike Telljohann
10:30–11:00	*LinGO Redwoods—A rich and dynamic treebank for HPSG* Stephan Oepen, Ezra Callahan, Dan Flickinger, Christoper D. Manning
11:00–11:30	Coffee break
11:30–12:30	Panel: Parser evaluation in context John Carroll, Patrick Paroubek, Owen Rambow, Hans Uszkoreit
12:30–14:00	Lunch break
14:00–14:30	*A test of the leaf-ancestor metric for parse accuracy* Geoffrey Sampson, Anna Babarczy
14:30–15:00	*Evaluating parser accuracy using edit distance* Brian Roark
15:00–15:10	Short break
15:10–15:40	*Evaluating syllabification: One category shared by many grammars* Karin Müller
15:40–16:10	*Towards comparing parsers from different linguistic frameworks: An information theoretic approach* Gabriele Musillo, Khalil Sima'an
16:10–16:40	*Evaluation of the Gramotron parser for German* Franz Beil, Detlef Prescher, Helmut Schmid, Sabine Schulte im Walde
16:40–17:10	Coffee break
17:10–17:40	*Evaluating a wide-coverage CCG parser* Stephen Clark, Julia Hockenmaier
17:40–18:10	*A comparison of evaluation metrics for a broad-coverage stochastic parser* Richard Crouch, Ronald M. Kaplan, Tracy H. King, Stefan Riezler
18:10–20:00	Wrap up and kick-off: Initiatives and action plans (open end)

Workshop Organisers

John Carroll	University of Sussex, UK
Anette Frank	DFKI GmbH, Saarbrücken, Germany
Dekang Lin	University of Alberta, Canada
Detlef Prescher	DFKI GmbH, Saarbrücken, Germany
Hans Uszkoreit	DFKI GmbH and Saarland University, Saarbrücken, Germany

Workshop Programme Committee

Salah Ait-Mokhtar	XRCE Grenoble
Gosse Bouma	Rijksuniversiteit Groningen
Thorsten Brants	Palo Alto Research Center
Ted Briscoe	University of Cambridge
John Carroll	University of Sussex
Jean-Pierre Chanod	XRCE Grenoble
Michael Collins	AT&T Labs—Research
Anette Frank	DFKI Saarbrücken
Josef van Genabith	Dublin City University
Gregory Grefenstette	Clairvoyance, Pittsburgh
Julia Hockenmaier	University of Edinburgh
Dekang Lin	University of Alberta
Chris Manning	Stanford University
Detlef Prescher	DFKI Saarbrücken
Khalil Sima’an	University of Amsterdam
Hans Uszkoreit	DFKI Saarbrücken and Saarland University

Table of Contents

Beyond PARSEVAL — Towards improved evaluation measures for parsing systems
John Carroll, Anette Frank, Dekang Lin, Detlef Prescher, Hans Uszkoreit 1

Relational evaluation schemes
Ted Briscoe, John Carroll, Jonathan Graham, Ann Copestake 4

Towards a dependency-oriented evaluation for partial parsing
Sandra Kübler, Heike Telljohann 9

LinGO Redwoods — A rich and dynamic treebank for HPSG
Stephan Oepen, Ezra Callahan, Dan Flickinger, Christoper D. Manning 17

A test of the leaf-ancestor metric for parse accuracy
Geoffrey Sampson, Anna Babarczy 23

Evaluating parser accuracy using edit distance
Brian Roark 30

Evaluating syllabification: One category shared by many grammars
Karin Müller 37

Towards comparing parsers from different linguistic frameworks: An information theoretic approach
Gabriele Musillo, Khalil Sima'an 44

Evaluation of the Gramotron parser for German
Franz Beil, Detlef Prescher, Helmut Schmid, Sabine Schulte im Walde 52

Evaluating a wide-coverage CCG parser
Stephen Clark, Julia Hockenmaier 60

A comparison of evaluation metrics for a broad-coverage stochastic parser
Richard Crouch, Ronald M. Kaplan, Tracy H. King, Stefan Riezler 67

Author Index

Babarczy, Anna 23

Beil, Franz 52

Briscoe, Ted 4

Callahan, Ezra 17

Carroll, John 1, 4

Clark, Stephen 60

Copestake, Ann 4

Crouch, Richard 67

Flickinger, Dan 17

Frank, Anette 1

Graham, Jonathan 4

Hockenmaier, Julia 60

Kaplan, Ronald M. 67

King, Tracy H. 67

Kübler, Sandra 9

Lin, Dekang 1

Manning, Christoper D. 17

Müller, Karin 37

Musillo, Gabriele 44

Oepen, Stephan 17

Prescher, Detlef 1, 52

Riezler, Stefan 67

Roark, Brian 30

Sampson, Geoffrey 23

Schmid, Helmut 52

Schulte im Walde, Sabine 52

Sima’an, Khalil 44

Telljohann, Heike 9

Uszkoreit, Hans 1

— Beyond PARSEVAL —
Towards Improved Evaluation Measures for Parsing Systems

John Carroll[1], **Anette Frank**[2], **Dekang Lin**[3], **Detlef Prescher**[2], **Hans Uszkoreit**[2]

[1] Cognitive and Computing Sciences
University of Sussex
Falmer, Brighton BN1 9QH
UK

[2] Language Technology Lab
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken
Germany

[3] Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada, T6G 2H1

1. Current Situation in Stochastic Parsing

The earliest corpus-based approaches to stochastic parsing (e.g. Sampson et al. (1989), Fujisaki et al. (1989), Sharman et al. (1990), Black (1992)) used a variety of data resources and evaluation techniques. With the creation of the Penn Treebank of English (Marcus et al., 1993) and the parser evaluation measures established by the PARSEVAL initiative (Black, 1992), new approaches to stochastic parsing and uniform evaluation regimes emerged (Magerman (1995), Charniak (1996), Collins (1996)), leading to impressive improvements in parser accuracy (Collins (1997), Charniak (2000), Bod (2001)).

In the meantime, annotated corpora have been built for several other languages, most notably the Prague Dependency Treebank for Czech (Hajic, 1998), and the NEGRA corpus for German (Skut et al., 1997). Well-known, but smaller corpora for English are the ATIS Corpus and SUSANNE. Many more corpora are available or under construction, e.g. the Penn treebanks for Chinese and Korean, the TIGER corpus for German, as well as corpora for Bulgarian, French, Italian, Portugese, Spanish, Turkish, etc. Annotation schemes in these treebanks vary, often motivated by language-specific characteristics. For example, dependency-based annotation is generally preferred for languages with relatively free word order.

More recently, in line with increasing interest in more fine-grained syntactic and semantic representations, stochastic parsing has been applied to several higher-order syntactic frameworks, such as unification-based grammars (Johnson et al., 1999), tree-adjoining grammars (Chen et al., 1999) and combinatory categorial grammars (Hockenmaier, 2001). In parallel, due to the lack of appropriate large-scale annotated training corpora, unsupervised methods have been investigated, i.e. training of manually written (context-free or unification-based) grammars on free text (Beil et al. (1999), Riezler et al. (2000), Bouma et al. (2001)).

As opposed to the PARSEVAL measures — which are based on phrase structure tree match — most of these novel parsing approaches use other evaluation measures, such as dependency-based, valence-based, exact, or selective category match.

2. Challenges for Parser Evaluation

Despite the emergence of stochastic parsing approaches using alternative syntactic frameworks, the currently established paradigm for evaluating stochastic parsing still consists of the combination of Penn Treebank English (Section 23) with PARSEVAL measures.

However, in practice (especially if we count industrial labs) parsing systems using treebank grammars are not representative of the field. Moreover, a strong trend in stochastic parsing is away from treebank grammars and towards higher-level syntactic frameworks and hand-built grammars.

Research in stochastic parsing with higher-order syntactic frameworks is therefore confronted with a lack of a common evaluation metrics: neither do the PARSEVAL measures straightforwardly correspond to dependency structures or other valence-based representations, nor have these alternative approaches come up with a common, agreed-on standard for evaluation. Furthermore, no common evaluation corpora exist for many alternative languages. To some extent, this problem has been circumvented by building small theory-specific treebanks (with the obvious drawbacks for supervised training and inter-comparability). In sum, the growing field in stochastic parsing with alternative syntactic models or languages other than English faces problems in benchmarking against the established Gold Standard.

As a consequence, the best-known stochastic parsers are trained for Penn Treebank English. Yet, to validate these parsers on a broader basis, it has to be evaluated how well these stochastic models carry over to languages with e.g. free word order, intricate long-distance phenomena, pro-drop properties, and agglutinative or clitic languages. Again, this presupposes the availability of annotated corpora and evaluation schemes appropriate to cover a broad range of diverse language types.

3. Towards a New Gold Standard

The current situation in stochastic parsing, as well as prospects for its future development, calls for a new and uniform scheme for parser evaluation which covers both shallow and deep grammars, different syntactic frameworks, and different language types.

What is needed is an annotation scheme bridging structural differences across diverse languages and frameworks. In practice, many researchers have been using their own evaluation metrics which, despite divergences, bear some common ground, namely higher-level syntactic annotations such as grammatical relations, dependencies, or subcategorization frames (Beil et al. (1999), Carroll et al. (2000), Collins et al. (1999), Hockenmaier (2001), etc). Such basic syntactic relations build on crucial, but underlying structural constraints, yet provide more abstract, functional information.

This information is not only an appropriate level of abstraction to bridge structural differences between languages and higher-level syntactic theories, but moreover, provides a basis for evaluation of partial, more shallow analysis systems, at a higher level of representation. For example, if the evaluation is against grammatical relation rather than phrase structure information, partial parsers extracting functional relations can be evaluated within the same setup as full parsers.

Starting from this state of affairs, one of the aims of the workshop will be to provide a forum for researchers in the field to discuss (define and agree on) a new, uniform evaluation metric which provides a basis for comparison between different parsing systems, syntactic frameworks and stochastic models, and how well they extend to languages of different types.

Definition of a new evaluation standard could be restrictive and flexible at the same time: flexible in that training can exploit fine-grained annotations of richer syntactic frameworks; and restrictive in that diverging analyses are then to be mapped to uniform (more coarse-grained) annotations for standardized evaluation.

4. Starting an Initiative

A previous LREC-hosted workshop on parser evaluation in 1998 in Granada brought together a number of people advocating parser evaluation based on dependencies or grammatical relations (Carroll and Briscoe (1998), Lin (1998), Bangalore et al. (1998)). The consensus of the concluding discussion at that workshop was that there is much common ground between these approaches, and that they constitute a viable alternative to the PARSEVAL measures.

In the meantime, as described above, many more corpora are under construction and novel stochastic parsing schemes are being developed, which call for an initiative for establishing a new, agreed-on evaluation standard for parsing which allows for comparison and benchmarking across alternative models and different language types.

The workshop is intended to bring together four parties: researchers in stochastic parsing, builders of annotated corpora, representatives from different syntactic frameworks, and groups with interests in and proposals for parser evaluation. As a kick-off initiative, the workshop should lead to collaborative efforts to work out a new evaluation metric, and to start initiatives for building or deriving sufficiently large evaluation corpora, and possibly, large training corpora according to the new metric.

In conclusion, stochastic parsing has now developed to a stage where new methods are emerging, both in terms of underlying frameworks and languages covered. These need to be brought together by means of a new evaluation metric to prepare the new generation of stochastic parsing.

5. Workshop Programme

The workshop comprises thematic papers focussing on benchmarking of stochastic parsing, parser evaluation, design of annotation schemes covering different languages, and different frameworks, as well as creation of high-quality evaluation corpora.

Intended as a forum for discussion, the workshop programme consists of paper presentations with discussion sessions and a panel, where important results of the workshop are summarized and discussed.

In the final session we intend to wrap-up, and plan a kick-off initiative leading to concrete action plans and the creation of working groups, as well as planning for future coordination. To maintain the momentum of this initiative we will work towards setting up a parsing competition based on new standard evaluation corpora and evaluation metric.

References

Srinivas Bangalore, Anoop Sarkar, Christine Doran, and Beth Ann Hockey. 1998. Grammar and parser evaluation in the xtag project. In *Workshop on the Evaluation of Parsing Systems*, LREC, Granada.

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth. 1999. Inside-outside estimation of a lexicalized PCFG for German. In *Proceedings of ACL'99*, College Park, MD.

Ezra Black. 1992. Meeting of interest group on evaluation of broad-coverage grammars of English. LINGUIST List 3.587, http://www.linguistlist.org/issues/3/3-587.html.

Rens Bod. 2001. What is the minimal set of fragments that achieves maximal parse accuracy? In *Proceedings of ACL-2001*.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of dutch. In *Computational Linguistics in The Netherlands 2000*.

John Carroll and Ted Briscoe. 1998. A survey of parser evaluation methods. In *Workshop on the Evaluation of Parsing Systems*, LREC, Granada.

Eugene Charniak. 1996. Tree-bank grammars. Technical Report CS-96-02, Brown University.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000)*, Seattle, WA.

J. Chen, S. Bangalore, and K. Vijay-Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*.

M. Collins, J. Hajic, L. Ramshaw, and Ch. Tillman. 1999. A Statistical Parser for Czech. In *Proceedings of ACL 99*.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, Santa Cruz, CA.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*, Madrid.

T. Fujisaki, F. Jelinek, J. Cocke, E. Black, and T. Nishino. 1989. A probabilistic method for sentence disambiguation. In *Proceedings of the 1st International Workshop on Parsing Technologies*.

J. Hajic. 1998. Building a syntactically annotated corpus: The prague dependency treebank. Issues of Valency and Meaning. Studies in Honour of Jarmila Panevova.

Julia Hockenmaier. 2001. Statistical parsing for ccg with simple generative models. In *Student Research Workshop of the 39th ACL/10th EACL*.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, MD.

D. Lin. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems*, LREC, Granada.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, Cambridge, MA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

S. Riezler, D. Prescher, J. Kuhn, and M. Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proc. of ACL-2000*.

G. Sampson, R. Haigh, and E. Atwell. 1989. Natural language analysis by stochastic optimization: a progress report on project april. *Journal of Experimental and Theoretical Artificial Intelligence*.

R. Sharman, F. Jelinek, and R. Mercer. 1990. Generating a grammar for statistical training. In *Proceedings of the DARPA Speech and Natural Language Workshop*.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC.

Relational Evaluation Schemes

Ted Briscoe*, **John Carroll**†, **Jonathan Graham***, **Ann Copestake***

*Computer Laboratory
University of Cambridge
{Ted.Briscoe, Ann.Copestake}@cl.cam.ac.uk

†Cognitive and Computing Sciences
University of Sussex
John.Carroll@cogs.susx.ac.uk

Abstract

We describe extensions to a scheme for evaluating parse selection accuracy based on named grammatical relations between lemmatised lexical heads. The scheme is intended to directly reflect the task of recovering grammatical and logical relations, rather than more arbitrary details of tree topology. There is a manually annotated test suite of 500 sentences which has been used by several groups to perform evaluations. We are developing software to create larger test suites automatically from existing treebanks. We are considering alternative relational annotations which draw a clearer distinction between grammatical and logical relations in order to overcome limitations of the current proposal.

1. Introduction

We have developed a scheme for evaluating parse selection accuracy based on named grammatical relations between lemmatised lexical heads. The scheme is intended to directly reflect the task of recovering semantic relations, rather than more arbitrary details of tree topology—as with the PARSEVAL scheme, which has been criticised frequently for the opaque relationship between its measures and such relations (Carroll *et al.*, 1998; Magerman, 1995; Srinivas, 1997). Carroll *et al.* (1998) provide more detailed motivation and comparison with other extant schemes.

Carroll *et al.* (1999, 2002 in press) report the development of a test suite of 500 sentences annotated with grammatical relations, the specification of the relations, and their criteria of application. The set of named relations are organised as a subsumption hierarchy in which, for example, subj(ect) underspecifies n(on)c(lausal)subj(ect). There are a total of 15 fully specified relations, however, many of these can be further subclassified; for example, subj relations have an initial-gr slot used to encode whether the syntactic subject is logical object (as in passive) and for other marked subjects (such as in locative inversion). Thus a fully specified GR might look like (ncsubj marry couple obj) to encode the subj relation in *The couple were married in August*, and the GR annotation of each sentence of the test suite consists of a set of GR *n*-tuples. Figure 1 gives the full set of named relations represented as a subsumption hierarchy. The most generic relation between a head and a dependent is dependent. Where the relationship between the two is known more precisely, relations further down the hierarchy can be used, for example mod(ifier) or arg(ument). Relations mod, arg_mod, aux, clausal, and their descendants have slots filled by a type, a head, and its dependent; arg_mod has an additional fourth slot initial_gr. Descendants of subj, and also dobj have the three slots head, dependent, and initial_gr. Relation conj has a type slot and one or more head slots. The x and c prefixes to relation names differentiate clausal control alternatives.

When the proprietor dies, the establishment should become a corporation until it is either acquired by another proprietor or the government decides to drop it.

```
(ncsubj die proprietor _)
(ncsubj become establishment _)
(xcomp _ become corporation)
(ncsubj acquire it obj)
(arg_mod by acquire proprietor subj)
(ncmod _ acquire either)
(ncsubj decide government _)
(xcomp to decide drop)
(ncsubj drop government _)
(dobj drop it _)
(cmod when become die)
(cmod until become acquire)
(cmod until become decide)
(detmod _ proprietor the)
(detmod _ establishment the)
(detmod _ corporation a)
(detmod _ proprietor another)
(detmod _ government the)
(aux _ become shall)
(aux _ acquire be)
(conj or acquire decide)
```

Figure 2: Grammatical relation sample annotation.

Figure 2 shows the GR encoding of a sentence from the Susanne corpus.

The evaluation metric uses the standard precision and recall and F_α measures over sets of such GRs. Carroll and Briscoe (2001) also make use of weighted recall and precision (as implemented in the PARSEVAL software) to evaluate systems capable of returning *n*-best sets of weighted GRs. The software makes provision for both averaged scores over all relations as well as scores by named relation. It also supports partial scoring in terms of non-leaf named relations which underspecify leaf relations. The current specification of the

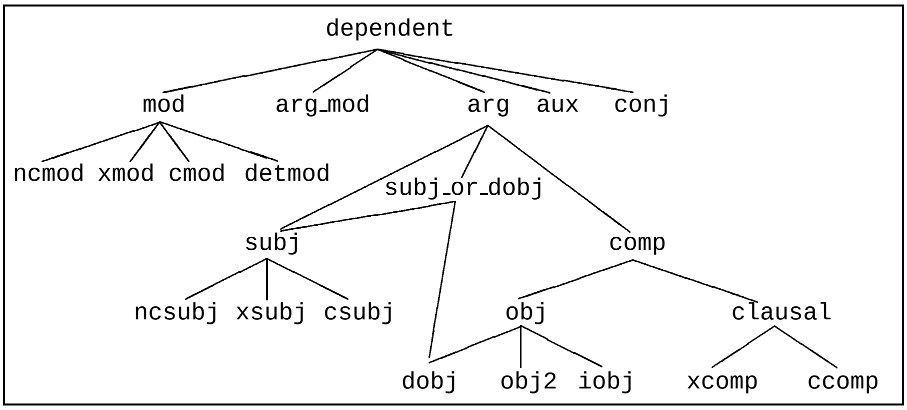


Figure 1: Grammatical relation hierarchy.

scheme along with the test suite and evaluation software (implemented in Common Lisp) is available from http://www.cogs.susx.ac.uk/lab/nlp/carroll/greval.html

Evaluation of stochastic parsers using relational schemes similar to our proposal is becoming more common (e.g. Collins, 1999; Lin, 1998; Srinivas, 2000). However, comparison across such results is hampered by the fact that the set of relations extracted is not standardised across these schemes, and it is clear that some relations (e.g. that between determiners and head nouns) are much easier to extract than others (e.g. control relations in predicative complements), as can be seen, for example, from the separate and divergent precision / recall results by named relation reported by Carroll *et al.* (1999). This makes meaningful comparison of 'headline results' such as mean overall F_1 measures very hard. Our scheme attempts to ameliorate these problems by supporting different levels of granularity within named relations (ncsubj / csub / xsubj $\subset$ subj) and encouraging not only the reporting of overall mean precision / recall scores, but also separate scores for each named relation.

In the rest of this paper we describe ongoing efforts to improve the evaluation scheme and enlarge the annotated test suite(s).

2. Divergent system output representations

There remain several infelicities in the current scheme that are a consequence of the method of factoring information into distinct relations which, in fact, still encode composites of information. For example, a system which clearly separates categorial constituency and functional information, such as one based on LFG, might choose to map F-structure SUBJ relations to subj in our scheme. A more constituency based parser might map NPs immediately dominated by S and preceding a VP to ncsubj, and Ss in the same configuration to csubj. Superficially the latter system is extracting more information because the relation name encodes categorial as well as relational information. The current scoring metric also assigns a penalty to systems that do not recover fully-specified (leaf) relations. However, for either system to score in the evaluation the subj relation most hold between lemmatised heads of the appropriate type, so the distinction between clausal and non-clausal subjects is maintained in both, since clausal subjects have verbal heads. On the other hand a system which systematically returned subj-or-dobj relations, as opposed to a leaf subj or obj one, would clearly be losing significant information pertinent to recovery of underlying logical relations.

There are many other cases of divergent encoding of aspects of categorial and functional information: for example, a LFG system will clearly distinguish clausal and predicative complements at F-structure corresponding directly to the xcomp / ccomp distinction in our relational scheme. However, a parser that represents such complements as clauses (S nodes) with or without an empty (PRO) NP subject, as in the Penn WSJ Treebank, would need to utilise a more complex (non-local) mapping from tree topology and node labels to named relations in order to maintain the xcomp / ccomp distinction. However, in this case, the easier underspecification to comp is genuinely significant since in either case the relation will hold between the same lexical (verbal) heads.

There are, in principle, two ways of dealing with such divergences. The first is to complicate the mapping from system output to named relations so that the specific set of leaf relations identified in the current scheme is recovered, if it is deducible from the total system output. The second is to modify the scoring metric so that informationally insignificant underspecification is not penalised. In some cases, such as the LFG system SUBJ case described above, the latter step will be much easier. In the new version of the specification and evaluation measure, we will attempt to identify such cases and parameterise the evaluation software to compute scores appropriately, as well as provide more specific guidance on mapping of named relations to the output of extant systems. This should improve the validity of cross-system evaluation. However, problems of this type are likely to emerge for each new system representation considered, so this is likely to be an ongoing process requiring judgement on the part of evaluators coupled with explicit description of decisions made alongside reported socres.

Provision of a flexible software system for mapping from parser output representations to factored relational ones may also ameliorate this class of problems (see section 5.). In particular, where a specific choice of system output representation necessitates a more complex mapping to leaf relations in our scheme, it would facilitate fair and feasible cross-system comparison if the evaluation scheme provided software that would recover the named leaf relations from the system output. Once again, each new system representation is likely to throw up new problems of this type, so flexible and easily parameterisable software will be more useful.

3. Surface / logical form divergence

The current annotation scheme attempts to stay close to surface grammatical structure, while also encoding divergence from predicate-argument structure / logical form. Divergence is currently encoded using two distinct mechanisms for different types of cases. Extra slots in named relations are used to indicate surface / underlying logical relation divergences, as with subj discussed in section 1. An additional relation is used for coordination (conj) to indicate how the conjunction scopes over the individual conjuncts.

One conspicuous area where the current scheme is inadequate is with equative and comparative constructions, which occur quite frequently in the 500 sentence test suite. Semantically, it is standard to treat *more* and *as*, etc as generalised quantifiers over propositions so that an example like

> *GR evaluation is more / as attractive than / as PARSEVAL*

is represented (very crudely) as

> more′(is-attr′(GReval′), is-attr′(PARSEVAL′))

This example, however, is annotated by the GRs

> (ncmod _ attractive more)
> (ncmod than attractive PARSEVAL)

However, in general, the GR annotation of such constructions is variable because of the varied surface syntactic location of *more* and *as* and also because of the optionality of and degree of ellipsis in the *than / as* constituent. Furthermore, because of the divergence between surface form and logical form the current annotations give little indication of whether a system would be capable of outputting an appropriate logical form. Replacing the current annotation with one close to the target logical form would undermine the scheme, since most extant stochastic parsers would be unable to generate such a representation.

One alternative is to additionally annotate such constructions with construction-specific named relations. This could be based on the approach to coordination, where the named relation

> (conj conj-type conjunct-heads+)

is used in addition to distributing the conjunct heads over multiple occurrences of the relation over the coordinate construction. For comparatives and equatives, we could add a relation like

> (compequ as/more/... attractive GReval PARSEVAL)

encoding the type of comparison, the predicate of comparison, and the arguments to this predicate.

There are undoubtedly further constructions, beyond coordination and comparatives / equatives that merit some such treatment. The advantage of adding additional construction-specific named relations that encode the same phenomena from different perspectives is that the resulting annotation will support a graded and fine-grained evaluation of the extent to which a specific system can support recovery of underlying logical form / predicate-argument structure in addition to surface grammatical relations. The disadvantage of this approach is that the scheme is likely to become more complex, and thus its recovery from any specific parser representation more time-consuming. In addition, the encoding of the underlying logical relations in the GR scheme has already spawned two divergent mechanisms, and may well require more.

4. MRS-style annotation scheme

A second and more complex but potentially more thorough approach to the issue of surface / logical form divergence is to bleach the current GR scheme of all attempts to represent such mismatches and instead define a factored and underspecified semantic annotation scheme to be used in tandem with GR annotation. The approach to underspecified logical representation developed by Copestake *et al.* (2001) can be extended to allow semantics to be underspecified to a much greater degree. In this extension of minimal recursion semantics (MRS), a Parsons-style notation (Parsons, 1990) is used, with explicit equalities representing variable bindings. For instance, from

> *The couple were married.*

a particular parsing system might return

> (ARGN u1 u2)
> (marry u3)
> (couple u4)

However, the fully specified test suite annotation would be

> (ARG2 e1 x4)
> (marry e2)
> (couple x3)
> e1 = e2
> x3 = x4

where ARG2 is formally a specialisation of ARGN, and the equalities and variable sorts also add information.

Potentially, this would allow us to dispense with complications like init-gr fields in the GR annotation and provide a principled basis for a graded evaluation of the recovery of logical form. The disadvantage over the further extension of the existing scheme is that two stages of extraction from specific system output are now required, the matching operations and scoring metrics become more complex, and the ability to do a graded evaluation of recovery of both grammatical and logical relations may be somewhat undermined.

```
try
  {
   while (dd)
    {
     String s = readWord(W);
     setS += 1;

     if (c==0) dd = false;

     if (s.equals("S"))
      {
       if (domprecedes("S", "NP",
                       "VP", setS))
        { String head = mainverb(setvp);
          String dependent =
            righthead("NP", "N-", setnp);
          String objslot =
            ispassive(setvp);
          System.out.println(
                   "(ncsubj " + head + "
                   " + dependent + "
                   " + objslot + ")");
        }
      }
    }
  }
```

Figure 3: The ncsubj extraction class.

5. Enlarging and improving the test suite(s)

The current test suite of 500 sentences is too small, but was still labour-intensive to create semi-automatically. Consequently, it contains a number of inadequacies: tokenisation of multiwords is somewhat arbitrary, some relations which should be included are systematically omitted (e.g. predicative XP complements of *be* have not been annotated with their controlled subjects), quotation marks have been systematically removed, and so forth. The next release will attempt to remove these inadequacies. However, it is clear that we also need a method for annotating much more data efficiently. To this end we have been developing a generic system, implemented in JAVA, that can be applied to existing treebanks to extract relational information (Graham, 2002). This system can, in principle, extract GRs in the current or related schemes, or even (possibly underspecified) MRSs. It can be parameterised for different extant treebanks, such as Penn Treebank-II or Susanne, and requires a set of declarative rules expressed in terms of tree topology and node labels for each named relation. The system has been designed to process labelled trees looking for relations defined ultimately in terms of (immediate) dominance and (immediate) precedence efficiently. It has been tested on a subset of GRs, concentrating particularly on the subj sub-hierarchy. A fragment of the class for ncsubj encoding relevant constraints is shown in Figure 3, giving a sense of the degree of parameterisation required for different representations. Running a first prototype of the GR extractor on the 30 million word automatically annotated WSJ BLLIP corpus distributed by the LDC results in estimated recovery of 86% of ncsubj and dobj relations with a precision of 84%, taking around 3 hours CPU time on standard hardware.

This system will facilitate rapid automatic construction of relational annotation according to specified input and output scheme(s) up to the limit of what is currently represented in treebanks and system output. Our longer term plan is to make this software, and a number of rule sets implemented in it, available as part of the evaluation scheme. This should facilitate both the construction of test data and the mapping of system output to the required format.

6. Conclusions

Relational schemes for parser evaluation are gaining in popularity over the exclusive use of PARSEVAL or similar tree topology based measures. We hope that the ongoing work reported here will facilitate further cross-system and within-system relational evaluation. To this end, we are developing test suites and software to support flexible mapping from system and treebank output to relational encodings of grammatical and underlying logical relations, and actively seeking feedback from the community on weaknesses of our current encoding scheme and evaluation measures and errors in our current test set.

Acknowledgements

We would like to thank Ron Kaplan for carefully documenting many errors and inconsistencies in our semi-automatic annotation of the 500 word test suite. The GR encoding scheme was heavily influenced by the EAGLES encoding scheme, primarily developed by Antonio Sanfilippo. We would also like to thank Anne Abeillé and Srinivas Bangalore for useful discussions. This work was partially supported by the EPSRC-funded RASP project (grants GR/N36462 and GR/N36493).

References

Carroll, J. and E. Briscoe (2001) ‘High precision extraction of grammatical relations’, *Proceedings of the 7th ACL/SIGPARSE International Workshop on Parsing Technologies (IWPT’01),* Beijing, China, pp. 78–89.

Carroll, J., E. Briscoe and A. Sanfilippo (1998) ‘Parser evaluation: a survey and a new proposal’, *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, pp. 447–454.

Carroll, J., G. Minnen and E. Briscoe (1999) ‘Corpus annotation for parser evaluation’, *Proceedings of the EACL-99 Post-Conference Workshop on Linguistically Interpreted Corpora (LINC’99),* Bergen, Norway, pp. 35–41.

Carroll, J., G. Minnen and E. Briscoe (2002, in press) ‘Parser evaluation using a grammatical relation annotation scheme’ in Abeille, A. (ed.), *Treebanks: Building and Using Syntactically Annotated Corpora,* Dordrecht: Kluwer.

Collins, M. (1999) *Head-driven Statistical Models for Natural Language Parsing,* PhD Dissertation, University of Pennsylvania.

Copestake, A., A. Lascarides and D. Flickinger (2001) ‘An algebra for semantic construction in constraint-based grammars’, *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics,* Toulouse, France, pp. 132–139.

Graham, J. (2002, in preparation) *From Treebank to Lexicon*, DPhil Dissertation, University of Cambridge, Computer Laboratory.

Lin, D. (1998) 'Dependency-based evaluation of MINIPAR', *Proceedings of the The Evaluation of Parsing Systems: Workshop at the 1st International Conference on Language resources and Evaluation*, Granada, Spain.

Magerman, D. (1995) *Natural Language Parsing as Statistical Pattern Recognition,* PhD Dissertation, Stanford University.

Parsons, T. (1990) *Events in the Semantics of English,* MIT Press, Cambridge, MA.

Srinivas, B. (1997) *Complexity of Lexical Descriptions and its Relevance to Partial Parsing,* PhD Dissertation, University of Pennsylvania.

Srinivas, B. (2000) 'A lightweight dependency analyzer', *Natural Language Engineering, vol.6.2,* 113–138.

Towards a Dependency-Oriented Evaluation for Partial Parsing

Sandra Kübler, Heike Telljohann

Seminar für Sprachwissenschaft
Wilhelmstr. 113
D-72074 Tübingen
Germany
{kuebler,hschulz}@sfs.uni-tuebingen.de

Abstract

Quantitative evaluation of parsers has traditionally centered around the PARSEVAL measures of *crossing brackets, (labeled) precision*, and *(labeled) recall*. However, it is well known that these measures do not give an accurate picture of the quality of the parser's output. Furthermore, we will show that they are especially unsuited for partial parsers. In recent years, research has concentrated on dependency-based evaluation measures. We will show in this paper that such a dependency-based evaluation scheme is particularly suitable for partial parsers. TüBa-D, the treebank used here for evaluation, contains all the necessary dependency information so that the conversion of trees into a dependency structure does not have to rely on heuristics. Therefore, the dependency representations are not only reliable, they are also linguistically motivated and can be used for linguistic purposes.

1. Introduction

Quantitative evaluation of parsers has traditionally centered around the PARSEVAL measures of *crossing brackets, (labeled) precision*, and *(labeled) recall* (Black et al., 1991). However, it is well known that these measures do not give an accurate picture of the quality of the parser's output (cf. Manning and Schütze (1999)), e.g. in cases of attachment errors. Additionally, many phenomena like negation or unary branches are ignored in the original measures in order to allow a comparison between parsers that use incompatible grammars. For this reason, research in recent years has concentrated on dependency-based evaluation measures (cf. e.g. Lin (1995), Lin (1998)). We will show in this paper that such a dependency-based evaluation scheme is particularly suitable for partial parsers since it does not lead to disproportionately high losses in precision and recall for partial parses. Furthermore, the dependency representations are not only reliable, they are also linguistically motivated and can be used for linguistic purposes since the treebank used here for evaluation contains all the necessary dependency information.

2. Deficiencies of Constituency-Based Precision and Recall

It is a well known fact that the PARSEVAL measures do not always give an accurate picture of the quality of a parser's output. Carroll and Brisoce (1996), for example, note that the *crossing brackets* measure is too lenient in case of errors involving the disambiguation of arguments and adjuncts, which in some cases are not recognized as errors. The failure to attach a constituent which should be embedded n levels deep leads to n crossing errors, while this constituent may not be very important to the overall structure. Manning and Schütze (1999) show that this behavior is mirrored in *precision* and *recall*: If a constituent is attached very high in a complex right branching structure, but the parser attached it at a lower point in the structure, both precision and recall will be greatly diminished. An example of such a parsing error for the sentence "ich nehme den Zug nach Frankfurt an der Oder" (I will take the train to Frankfort on the Oder) is shown in Figure 1[1]. There the prepositional phrase "an der Oder" is erroneously grouped as an adjunct of the verb instead of being attached as a postmodifier to the noun phrase "nach Frankfurt" (cf. the following section for a description of the annotation scheme). The correct tree is shown in Figure 2. When using the PARSEVAL measures, the output of the parser shown in Figure 1 results in $10/13 = 76.92\%$ recall[2] and $10/12 = 83.33\%$ precision, the only error being the wrong attachment of the last prepositional phrase.

The same behavior can be observed when the parser attaches a constituent very high in a complex right branching structure instead of very low, or if the constituent is not attached at all. The latter is often the case for chunk parsers (Abney, 1991; Abney, 1996) or partial parsers (cf. e.g. Aït-Mokhtar and Chanod (1997)). These parsers generally aim at annotating only partial, reliably discoverable tree structures, i.e. base phrases and clausal structures. Postmodifications are generally not attached since this decision cannot be taken reliably based on very limited local context. TüSBL (Kübler and Hinrichs, 2001a; Kübler and Hinrichs, 2001b), e.g., a similarity-based parser for German, annotates syntactic structures including function-argument structure in a two-level architecture: in the first phase, a deterministic chunk parser (Abney, 1996) is used to anal-

[1] All syntactic trees shown in this paper follow the data format for trees defined by the NEGRA project of the Sonderforschungsbereich 378 at the University of the Saarland, Saarbrücken. They were printed by the NEGRA graphical annotation tool *Annotate* (Brants and Skut, 1998; Plaehn, 1998).

[2] Contrary to the original PARSEVAL measures, we do count the root node as well since there exist different root nodes in the annotation scheme, and there are cases when a sentence in the treebank is annotated with more than one tree (e.g. interjective utterances).

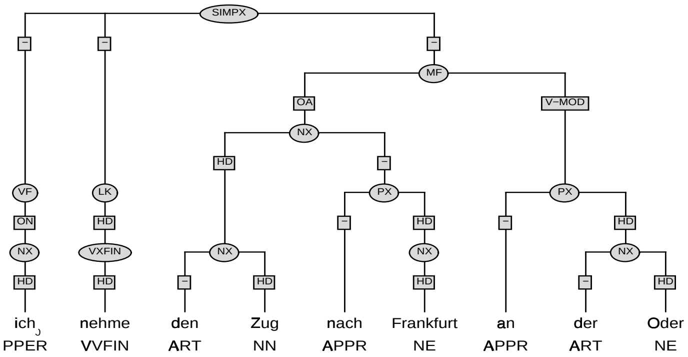


Figure 1: Wrong attachment of the prepositional phrase “an der Oder”.

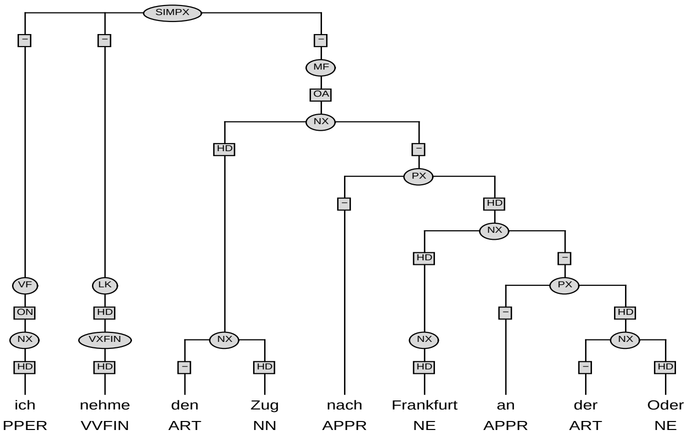


Figure 2: Correct attachment of the prepositional phrase “an der Oder”.

yse major syntactic constituents such as non-recursive base phrases and simplex clauses. As a consequence, dependency relations between individual chunks, such as grammatical functions or modification relations, within a clause remain unspecified. In the second step, the attachment ambiguities are resolved, and the partial annotation of the first step are enriched by dependency information. A typical output of this phase is shown in Figure 3. The second phase of analysis is based on a similarity-based machine learning approach, which uses a similarity metric to retrieve the most similar sentence to the input sentence from the instance base and adapts the respective tree to the input sentence. (For a more detailed description of the algorithm cf. Kübler and Hinrichs (2001a) and Kübler and Hinrichs (2001b).) The parser is designed to prefer partial analyses over uncertain ones. In some cases, this strategy leads to unattached phrases, mostly at the end of sentences, which results in high losses in precision and recall. We therefore propose to use a dependency-based evaluation as described by Lin (1995) and Lin (1998), in which both the gold standard and the parser’s output are transformed into dependencies and then compared on the basis of dependencies rather than on the basis of the constituent structure.

3. The TüBA-D Treebank

The dependency-based evaluation was based on the German corpus TüBa-D (Stegmann et al., 2000; Hinrichs et al., 2000a; Hinrichs et al., 2000b), which consists of approximately 38,000 syntactically annotated sentences. For this treebank, a theory-neutral and surface-oriented annotation scheme has been adopted that is inspired by the notion of topological fields – in the sense of Herling (1821), Erdmann (1886), Drach (1937), Reis (1980), and Höhle (1985) – and enriched by a level of predicate-argument structure, which guides the conversion into dependencies. The linguistic annotations pertain to the levels of morpho-syntax (part-of-speech tagging) (Schiller et al., 1995), syntactic phrase structure, and function-argument structure.

The tree structure contains different types of syntactic information in the following way: As the primary clustering principle the theory of topological fields (Höhle, 1985) is adopted, which captures the fundamental word order regularities of German sentence structure. In verb-second sentences, the finite verb constitutes the left sentence bracket (LK) and the verb complex the right sentence bracket (VC). This sentence bracket divides the sentence into the following topological order of fields: initial field (VF), LK, middle field (MF), VC, final field (NF). This structuring concept in addition favors bracketings that do not rely on crossing branches and traces to describe discontinuous dependencies.

Below this level of annotation, i.e. strictly within the bounds of topological fields, a phrase level of predicate-argument structure is established with its own descriptive

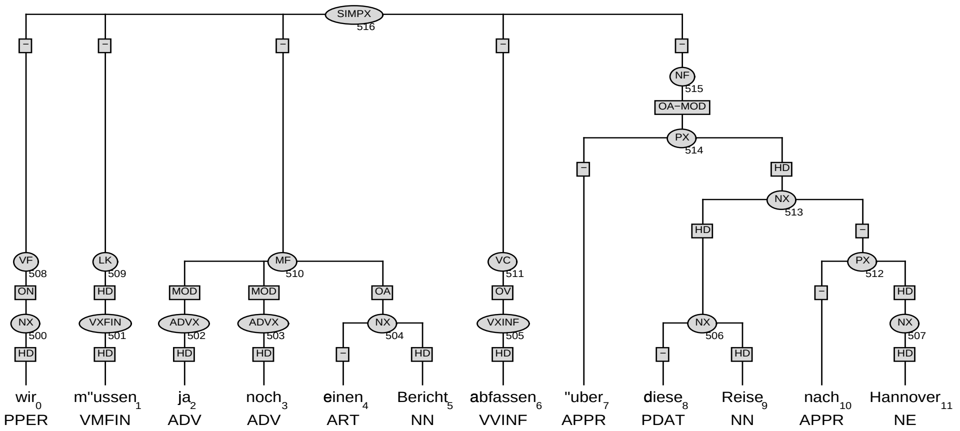


Figure 3: A tree annotated according to the TüBa-D treebank annotation scheme.

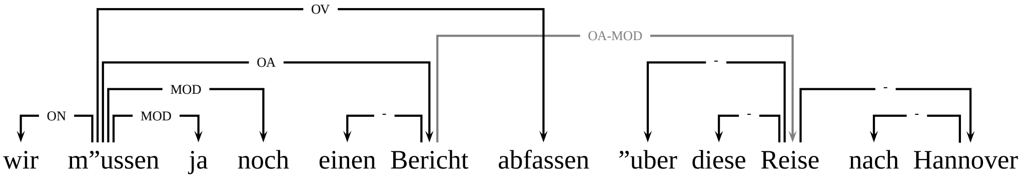


Figure 4: The dependency structure of the tree in Figure 3. The crossing dependency is shown in gray.

inventory based on a minimal set of assumptions concerning constituenthood, phrase attachment, and grammatical functions that have to be captured by any syntactic theory: nodes are labeled with syntactic categories on four different levels of annotation (sentence level, field level, phrase level, and lexical level), edges denote grammatical functions on the phrase level (i.e. immediately below the topological fields) and head/non-head distinctions within phrases. The integrated constituent analysis with its information about grammatical functions ensures that the resulting dependency structures are linguistically motivated and can also be used for linguistic purposes.

An example of such a tree for the sentence “wir müssen ja noch einen Bericht abfassen über diese Reise nach Hannover” (we still need to write a report on this journey to Hanover) is shown in Figure 3 (for more information about the annotation scheme cf. Stegmann et al. (2000)).

Two specific edge labels denote whether a constituent has the function of a head (HD), e.g. a phrase (NX, PX, ADJX, ADVX, VXFIN, VXINF), or a non-head (-), e.g. a determiner or a modifier attached to a phrase. On any annotation level, there is at most one head. The head of a sentence structure (e.g. SIMPX) is always the finite verb, which can be found in the left sentence bracket (LK). If there is no LK, the head is represented by the finite verb in the verb complex (VC). In coordinations, each conjunct depends on the head of the whole construction. Therefore, conjuncts are denoted with the non-head edge label.

The constituents below the topological fields are assigned grammatical functions. A subset of the edge label set consists of labels denoting the grammatical function of complements and modifiers, which depend on the head of the sentence. Another subset consists of labels determining long distance dependencies among these complements or modifiers as well as between conjuncts of split-up coordinations.

In Figure 3, e.g., the first constituent is marked as subject (ON), the finite verb is the head (HD), the two adverbs are modifiers (MOD), and the second noun phrase represents the direct object (OA). The constituent following the verb complex modifies the direct object (OA-MOD). Since the annotation scheme for the TüBa-D treebank facilitates a theory-neutral and surface-oriented representation of syntactic trees, this long distance relation is marked by the label OA-MOD (modifier of the accusative object) which refers to OA (accusative object) in the same tree; instead of using crossing branches and traces. This shows that long distance dependencies, which can even go beyond the border of topological fields, are encoded by special naming conventions for edge labels. Unambiguous edge labels, referring to exactly one non-adjacent constituent in the same tree, are used either for long distance modifications (*X*-MOD) like in the example above or for the rightmost conjunct of split-up coordinations (*X*K) (for an example cf. Figure 5). In both patterns, *X* is a variable for the grammatical function of the constituent to which it refers.

4. Converting TüBa-D into Dependencies

For TüBa-D, the conversion of the constituent structure into dependencies is in general determined by the head/non-head distinction in the tree. The dependency relations are labeled with the functional labels of the governed constituents. Using these strategies, the tree shown in Figure 3 is converted into the dependency structure in Figure 4. Here, the noun phrase “einen Bericht” is converted into one dependency relation, which denotes that the noun “Bericht”

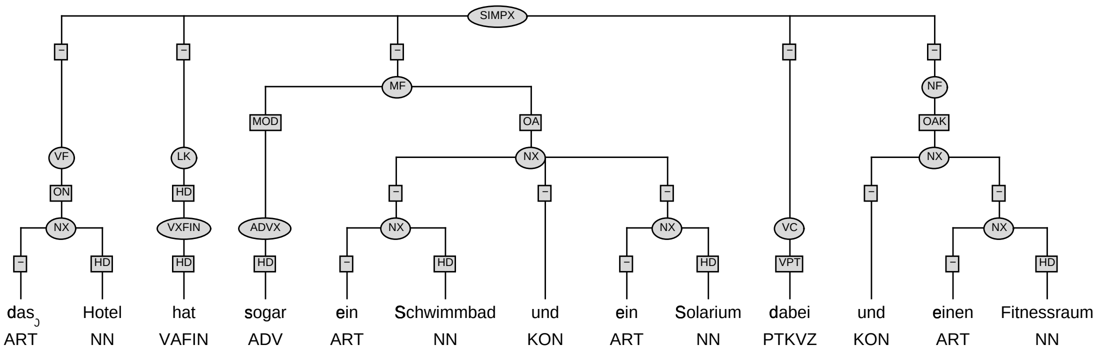


Figure 5: A complex coordination of noun phrases.

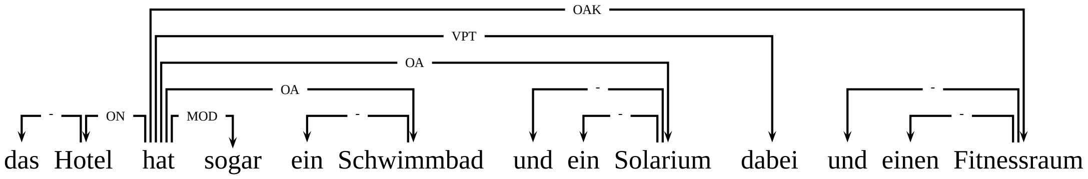

OAK
VPT
OA
OA
ON
MOD
-
das Hotel hat sogar ein Schwimmbad und ein Solarium dabei und einen Fitnessraum

Figure 6: The dependency structure of the tree in Figure 5.

governs the article “den”.

It is evident that the dependency structure contains two different types of dependencies: head/non-head dependencies within phrases (-) and dependencies from the finite verb, i.e. from the head of the clause, to its complements and adjuncts, which are labeled by the grammatical functions of the governed constituents (ON, MOD, OA, OV). This is why e.g. the direct object “einen Bericht” is represented as a dependent of the modal verb “müssen” although it constitutes an argument of the embedded main verb “abfassen”. However, the dependency relations among the finite verb and the (possibly multiple) infinite verbs is explicitly annotated in the syntactic and therefore in the dependency structure. And since information about clausal boundaries is present in the trees, even in this surface-oriented structure, the predicate-argument structure can be recovered.

The long-distance dependency between the direct object and its modifying prepositional phrase was modeled in the syntactic tree by the function label “OA-MOD” instead of by the attachment of the prepositional phrase to the direct object because the latter would have resulted in a *crossing branch*. In the dependency structure, this restriction is suspended, and the dependency is explicitly marked and has now resulted in crossing dependencies. Note that this is the only type of phrase-internal dependency that is not labeled by the head/non-head distinction but by unambiguous labels which denote their specific reference.

Since head information is present on all levels for the majority of constituents, specific decisions for determining dependency have to be taken only in the few cases when dependency relations are not clearly defined in the tree structure, i.e. for the following syntactic phenomena:

1. Conjunctions within coordinations do not depend on the head of the whole construction. Therefore, they are attached to the conjunct on their right hand side. An example of such a coordination is shown in Figure 5, the corresponding dependency structure in Figure 6. Here, the third conjunct is positioned after the verb complex and thus is assigned the label “OAK”.
Similar constructions with a preposition instead of a conjunction like “der achte bis neunte” (the eighth until the ninth) are treated in the same way. In order to stress the identical syntactic status of conjuncts, all conjuncts depend on the head governing the coordination. This analysis is in contrast to Lin (1998), who relies on the *Single Head Assumption* and proposes a dependency relation between the first and the second conjunct.

2. Sentence-initial coordinative particles such as “und” (and) or “oder” (or) in the KOORD-field depend on the head of the sentence.

3. The annotation of prepositional phrases in the syntactic trees is based on the principles of Dependency Grammar (Heringer, 1996); therefore, the noun phrase constitutes the head. For an example of the dependency structure of a prepositional phrase cf. the phrase “nach Hannover” in Figure 4. Circumpositions and postpositions are treated similarly.

4. The single elements of proper names, split cardinal numbers, the spelling of words, and complex conjunctions in the C-field, e.g. “so daß” (so that), are attached on the same level carrying a non-head edge label to indicate that there is no obvious dependency relation between them. Therefore, they are treated like conjuncts in coordinations.

5. A heuristic analysis has to be applied when long distance relations are underspecified – a MOD-MOD la-

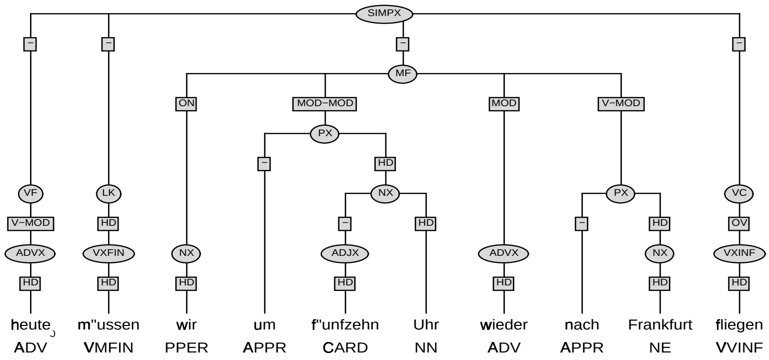


Figure 7: An ambiguous long-distance modifier: MOD-MOD.

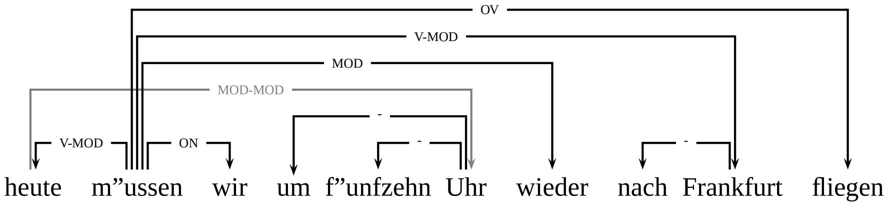


Figure 8: Resolved dependencies for ambiguous long-distance modifiers. The crossing dependency is shown in gray.

bel (modifier of a modifier), e.g., may refer to one of several modifiers in the sentence, such as for the sentence "heute müssen wir um fünfzehn Uhr wieder nach Frankfurt fliegen" (today we need to fly again to Frankfort) in Figure 7. Here, the long-distance modifier MOD-MOD might modify the V-MOD "heute" or the V-MOD "nach Franfurt". A close inspection of such ambiguous sentences in TüBa-D revealed that in a majority of all cases, the MOD-MOD label refers to the first V-MOD in the clause, or the first MOD if there is no V-MOD present. Exceptions to this rule are MOD-MODs in resumptive constructions, which generally refer to the modifier in the VF. Ambiguous OA-MODs generally refer to the closest OA in the clause. By applying these heuristics, the ambiguities are resolved in the dependency structure, as shown in Figure 8 for the syntactic tree in Figure 7.

5. Dependency-Based Parser Evaluation

Lin (1998) proposed a procedure for converting syntactic trees from the gold standard and from the parser into dependency structures. From these structures, precision and recall are calculated.

Another similar evaluation procedure was suggested by Srinivas et al. (1996), they first convert hierarchical phrasal constituents into chunks, and then compute the dependencies between these chunks. This is a valid approach for the Penn treebank annotation style, which assumes a complete flat annotation of complex noun phrases such as noun compounds. Parsers based on manually developed rules tend to assign more internal structure to such noun phrases, which leads to decreased precision. Reducing such phrases to flat chunks alleviates this problem of comparing these different structures. The TüBa-D annotations, however, assign more complex, non-trivial structures to complex noun phrases. Using the method of Srinivas et al. (1996) would therefore lead to a significant loss in information. Additionally, the flattening of phrases into chunks might introduce errors in the data in such cases, in which the conversion into chunks is not obvious, such as for the noun phrase "wichtige Konferenzen und Besprechungen" in the sentence "da haben wir noch wichtige Konferenzen und Besprechungen" (we still have important conferences and business meetings) shown in Figure 9.

Basili et al. (1998) developed a similar approach for the Italian language. But instead of parsing a sentence completely and then reducing this parse to chunks and dependencies between chunks, Basili et al. apply a chunk parser combined with a module that calculates dependencies between these chunks. For this approach, the same restrictions hold as for the evaluation procedure of Srinivas et al. (1996).

The evaluation method presented here is based on Lin's (Lin, 1998) approach. Following Lin's procedure, we first convert both the gold standard tree and the parser's output into dependency structures and compare these by applying (labeled) precision and (labeled) recall to these dependency structures.

TüSBL's analyses depend heavily on the syntactically annotated sentences contained in the instance base. It is therefore difficult to give examples of errors for specific sentences or linguistic phenomena. It is, however, possible to characterize the typical behavior of the parser and give typical examples of errors.

Attachment errors. Attachment errors as described in Section 1. are not very common for TüSBL. Since TüSBL uses the complete sentence as context to retrieve the most similar tree, it either finds the correct spanning analysis or it does not attach all constituents. In the few cases

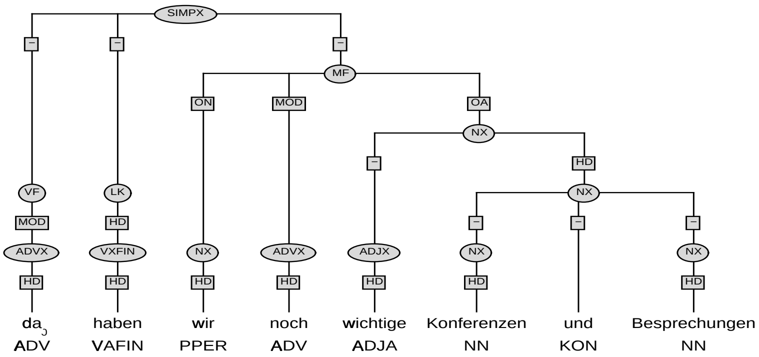


Figure 9: A complex noun phrase in the TüBa-D annotation scheme.

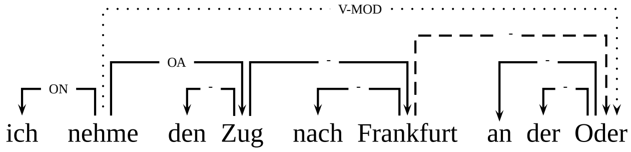


Figure 10: The dependency structure of the trees in Figure 1 and 2. The wrong attachment is shown as a dotted arc whereas the correct attachment is shown as a dashed arc.

where attachment errors are introduced by incorrect adaptations of the retrieved trees or in cases when a wrong tree is found as the most similar one, the parsers evaluation based on constituents suffers from the same problems as decribed in Section 2. above. The parser's output containing the wrong attachment in Figure 1 would result in $10/13 = 76.92\%$ recall and $10/12 = 83.33\%$ precision when using a constituent-based evaluation scheme. The dependency structure of the wrong and the correct attachment is shown in Figure 10. With the dependency-based evaluation, both precision and recall would be calculated as $7/8 = 87.50\%$.

Coordination. Coordination phenomena are in general very difficult to treat with deterministic partial parsers since this type of parsers needs to make the decision on the scope of a coordination early on when there is not enough information available. Two examples of coordination can be found in Figure 11. For both cases, TüSBL would typically retrieve these trees but not be able to attach the conjunction and the second conjunct, as shown in Figure 12 for the second example. For the first example, “am siebten und achten” (on the seventh and the eighth), this would lead to $2/4 = 50.00\%$ recall and $2/3 = 66.67\%$ precision. For the second example, “das wäre Mittwoch der dritte und Donnerstag der vierte August” (that would be Wednesday the third and Thursday the fourth of August), recall would be $9/12 = 75.00\%$ and precision $9/11 = 81.82\%$. If the evaluation is based on dependencies, TüSBL's analysis would deviate from the gold standard by the missing dependencies of the conjunction and the second conjunct. Therefore, recall would be $1/3 = 33.33\%$, for the first example, and $7/9 = 77.78\%$ for the second example. Precision would be $1/1 = 100\%$ for the first example and $7/7 = 100\%$ for the second example.

Another problematic coordination phenomenon constitute split-up coordinations such as in the sentence “das Hotel hat sogar ein Schwimmbad und ein Solarium dabei und einen Fitnessraum” (the hotel even has a swimming pool and a tanning booth – and a fitness room) in Figure 5. A typical error that might occur when parsing such sentences with TüSBL is that the split-up conjunct “und einen Fitnessraum” would not be attached. This would result in $12/14 = 85.71\%$ recall and $12/13 = 92.31\%$ precision. The evaluation based on the dependency structure shown in Figure 6 leads to $11/12 = 91.67\%$ recall and $11/11 = 100\%$ precision.

The comparison shows that dependency-based recall tends to suffer less than constituent-based recall since the unattached part of the coordination does not contribute to errors on higher levels, such as the MF and SIMPX in the second example, which are in principle correct. Dependency-based precision, on the other hand, does not depend on the level of embedding of the coordinations but only on the number of conjuncts that were correctly attached.

Unattached phrases. The failure to attach constituents at the end of an input sentence is the most common error type when evaluating partial parsers. It is generally part of the design decisions to prefer partial analyses which can be gained with a small amount of effort but which will be correct in a majority of cases to complete analyses which involve a high degree of manual labor and a higher error rate for attachment decisions. A typical analysis of TüSBL for the input sentence “wir müssen ja noch einen Bericht abfassen über diese Reise nach Hannover” would be similar to the tree in Figure 3; one possible error might be that the last PX (“nach Hannover”) could not be attached to the NX (“diese Reise”). Thus, the NX node 513 would be missing, and the PX node 514 would then immediately dominate the NX node 506. Using the PARSEVAL measures, this

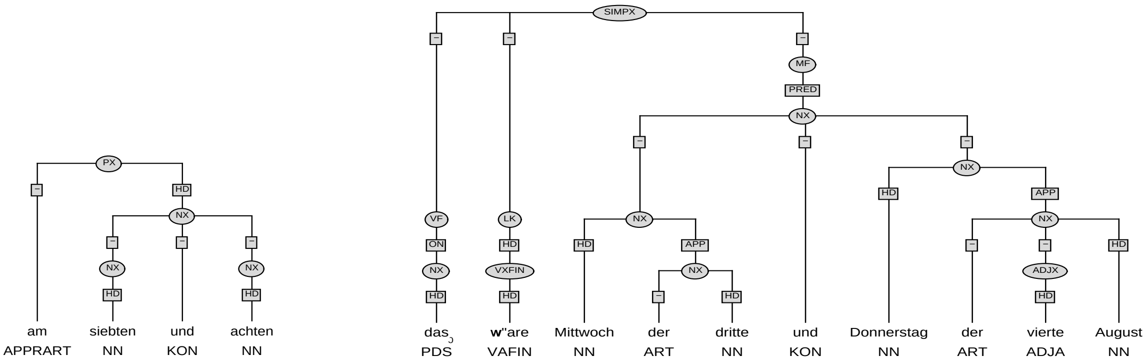


Figure 11: Two trees containing coordination.

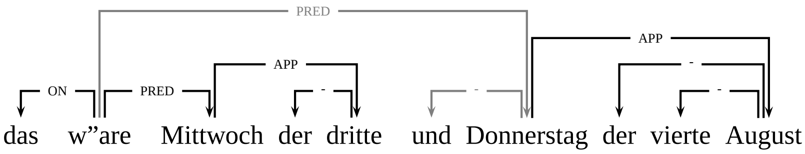


Figure 12: The dependency-based representation of the second example in Figure 11. TüSBL’s analysis is shown in black, the missing dependencies in gray.

error would result in $13/17 = 76.47\%$ labeled recall and $13/16 = 81.25\%$ labeled precision. The evaluation based on the dependency structure would give $10/11 = 90.90\%$ labeled recall and $10/10 = 100\%$ labeled precision. Considering that only the attachment of the final PX is missing and that the analysis of the sentence is otherwise correct and complete, the latter figures give a better picture of the quality of the partial parse.

6. Conclusion

We have shown that the PARSEVAL measures do not allow a suitable evaluation of partial parsers. If the evaluation is based on constituency, missing information in the partial parses leads to precision and recall errors in several constituents, and the losses in both measures are disproportionately high. We therefore proposed a dependency-based evaluation. TüBa-D, the treebank used here, contains all the necessary dependency information so that the conversion of trees into a dependency structure does not have to rely on heuristics. Therefore, the dependency representations are not only reliable, they are also linguistically motivated and can be used for linguistic purposes. Using these structures for evaluation ensures that missing information will not decrease the evaluation measures disproportionately, which allows a more suitable evaluation of partial information.

7. Acknowledgments

The research reported here was supported by the German Research Council (DFG) as part of the Sonderforschungsbereich 441 “Linguistische Datenstrukturen” (Linguistic Data Structures). The authors are grateful to Steven Abney, who made the chunk parser available, and to Prof. Hans Uszkoreit and his colleagues at the Universität des Saarlandes, who kindly provided us with the graphical annotation tool *Annotate*. The authors are also grateful to the anonymous reviewers for their helpful comments.

8. References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Caroll Tenney, editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

Steven Abney. 1996. Partial parsing via finite-state cascades. In John Carroll, editor, *Workshop on Robust Parsing (ESSLLI ’96)*, pages 8 – 15, Prague, Czech Republic.

Salah Aït-Mokhtar and Jean-Pierre Chanod. 1997. Incremental finite-state parsing. In *Proceedings of ANLP’97*, pages 72 – 79, Washington, D.C.

Roberto Basili, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. 1998. Evaluating a robust parser for Italian language. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 1998*, Granada, Spain.

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

Thorsten Brants and Wojciech Skut. 1998. Automation of treebank annotation. In *Proceedings of NeMLaP-3/CoNLL98*, pages 49 – 57, Sydney, Australia.

John Carroll and Ted Brisoce. 1996. Apportioning development effort in a probabilistic LR parsing system through evaluation. In *Proceedings of the ACL/SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 92 – 100, University of Pennsylvania, Philadelphia, PA.

Erich Drach. 1937. *Grundgedanken der Deutschen Satzlehre*. Frankfurt/M.

Oskar Erdmann. 1886. *Grundzüge der deutschen Syntax nach ihrer geschichtlichen Entwicklung dargestellt.* Stuttgart. Erste Abteilung.

Hans Jürgen Heringer. 1996. *Deutsche Syntax Dependentiell*. Stauffenburg-Verlag, Tübingen.

Simon Heinrich Adolf Herling. 1821. Über die Topik der deutschen Sprache. In *Abhandlungen des frankfurterischen Gelehrtenvereins für deutsche Sprache*, pages 296–362, 394. Frankfurt/M. Drittes Stück.

Erhard W. Hinrichs, Julia Bartels, Yasuhiro Kawata, Valia Kordoni, and Heike Telljohann. 2000a. The Tübingen treebanks for spoken German, English, and Japanese. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer, Berlin.

Erhard W. Hinrichs, Julia Bartels, Yasuhiro Kawata, Valia Kordoni, and Heike Telljohann. 2000b. The Verbmobil treebanks. In *5. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2000)*, pages 107 – 112, Ilmenau, Germany.

Tilman Höhle. 1985. Der Begriff “Mittelfeld, Anmerkungen über die Theorie der topologischen Felder. In *Akten des Siebten Internationalen Germanistenkongresses*, pages 329–340, Göttingen.

Sandra Kübler and Erhard W. Hinrichs. 2001a. From chunks to function-argument structure: A similarity-based approach. In *Proceedings of ACL-EACL 2001*, pages 338 – 345, Toulouse, France.

Sandra Kübler and Erhard W. Hinrichs. 2001b. TüSBL: A similarity-based chunk parser for robust syntactic processing. In *Proceedings of the First International Human Language Technology Conference, HLT-2001*, San Diego, CA, March.

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

Dekang Lin. 1998. A depedency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97 – 114.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Oliver Plaehn, 1998. *Annotate Bedienungsanleitung*. Universität des Saarlandes, Sonderforschungsbereich 378, Projekt C3, Saarbrücken, Germany, April.

Marga Reis. 1980. On justifying topological frames: ‘Positional field’ and the order of nonverbal constituents in German. *DRLAV: Revue de Linguistique*, 22/23:59 – 85.

Anne Schiller, Simone Teufel, and Christine Thielen. 1995. Guidelines für das Tagging deutscher Textkorpora mit STTS. Technical report, Universität Stuttgart and Universität Tübingen, September.

B. Srinivas, Christine Doran, Beth Ann Hockey, and Aravind Joshi. 1996. An approach to robust partial parsing and evaluation metrics. In John Carrol, editor, *Eight European Summer School in Language, Logic, and Information*, pages 70 – 82, Prague, Czech Republic.

Rosmary Stegmann, Heike Telljohann, and Erhard W. Hinrichs. 2000. Stylebook for the German Treebank in VERBMOBIL. Technical Report 239, Verbmobil.

LinGO Redwoods

A Rich and Dynamic Treebank for HPSG

**Stephan Oepen, Ezra Callahan, Dan Flickinger,
Christoper D. Manning, Kristina Toutanova**

Center for the Study of Language and Information
Stanford University
Ventura Hall, Stanford, CA 94305 (USA)

{oe|ezra99|dan|manning|kristina}@csli.stanford.edu

Abstract

The LinGO Redwoods initiative is a seed activity in the design and development of a new type of treebank. A treebank is a (typically hand-built) collection of natural language utterances and associated linguistic analyses; typical treebanks—as for example the widely recognized Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993), the Prague Dependency Treebank (Hajic, 1998), or the German TiGer Corpus (Skut, Krenn, Brants, & Uszkoreit, 1997)—assign syntactic phrase structure or tectogrammatical dependency trees over sentences taken from a naturally-occuring source, often newspaper text. Applications of existing treebanks fall into two broad categories: (i) use of an annotated corpus in empirical linguistics as a source of structured language data and distributional patterns and (ii) use of the treebank for the acquisition (e.g. using stochastic or machine learning approaches) and evaluation of parsing systems.

While several medium- to large-scale treebanks exist for English (and some for other major languages), all pre-existing publicly available resources exhibit the following limitations: (i) the depth of linguistic information recorded in these treebanks is comparatively shallow, (ii) the design and format of linguistic representation in the treebank hard-wires a small, predefined range of ways in which information can be extracted from the treebank, and (iii) representations in existing treebanks are static and over the (often year- or decade-long) evolution of a large-scale treebank tend to fall behind theoretical advances in formal linguistics and grammatical representation.

LinGO Redwoods aims at the development of a novel treebanking methodology, (i) *rich* in nature and *dynamic* in both (ii) the ways linguistic data can be retrieved from the treebank in varying granularity and (iii) the constant evolution and regular updating of the treebank itself, synchronized to the development of ideas in syntactic theory. Starting in October 2001, the project is aiming to build the foundations for this new type of treebank, develop a basic set of tools required for treebank construction and maintenance, and construct an initial set of 10,000 annotated trees to be distributed together with the tools under an open-source license. Building a large-scale treebank, disseminating it, and positioning the corpus as a widely-accepted resource is a multi-year effort; the results of this seeding activity will serve as a proof of concept for the novel approach that is expected to enable the LinGO group at CSLI both to disseminate the approach to the wider academic and industrial audience and to secure appropriate funding for the realization and exploitation of a larger treebank. The purpose of publication at this early stage is three-fold: (i) to encourage feedback on the Redwoods approach from a broader academic audience, (ii) to facilitate exchange with related work at other sites, and (iii) to invite additional collaborators to contribute to the construction of the Redwoods treebank or start its exploitation as early-access versions become available.

1. Why Another (Type of) Treebank?

For the past decade or more, symbolic, linguistically oriented methods (like those pursued within the HPSG framework; see below) and statistical or machine learning approaches to NLP have typically been perceived as incompatible or even competing paradigms; the former, more traditional approaches are often referred to as 'deep' NLP, in contrast to the comparatively recent branch of language technology focussing on 'shallow' (text) processing methods. Shallow processing techniques have produced useful results in many classes of applications, but have not met the full range of needs for NLP, particularly where precise interpretation is important, or where the variety of linguistic expression is large relative to the amount of training data available. On the other hand, deep approaches to NLP have only recently achieved broad enough grammatical coverage *and* sufficient processing efficiency to allow the use of HPSG-type systems in certain types of real-world applications. Fully-automated, deep grammatical analysis of unrestricted text remains an unresolved challenge.

In particular, applications of analytical grammars for natural language parsing or generation require the use of sophisticated statistical techniques for resolving ambiguities. We observe general consensus on the necessity for bridging activities, combining symbolic and stochastic approaches to NLP; also, the transfer of HPSG resources into industry has amplified the need for general parse ranking, disambiguation, and robust recovery techniques which all require suitable stochastic models for HPSG processing. While we find promising research in stochastic parsing in an number of frameworks, there is a lack of appropriately rich and dynamic language corpora for HPSG. Likewise, stochastic parsing has so far been focussed on IE-type applications and lacks any depth of semantic interpretation. The Redwoods initiative is designed to fill in this gap.

Most probabilistic parsing research—including, for example, work by by Collins (1997), Charniak (1997), and Manning and Carpenter (2000)—is based on branching process models (Harris, 1963). An important recent advance in this area has been the application of log-linear models (Agresti, 1990) to modeling linguistic systems. These models can deal with the many interacting dependencies and the structural complexity found in constraint-based or unification-based theories of syntax (Johnson, Geman, Canon, Chi, & Riezler, 1999). The availability of even a medium-size treebank would allow us to begin exploring the use of these models for probabilistic disambiguation of HPSG grammars. At the same time, other researchers have started work on stochastic HPSG (or are about to), some pursuing unsupervised approaches, but in many cases using the same grammar or at least the same descriptive formalism and grammar engineering environment. The availability of a reasonably large, hand-disambiguated HPSG treebank is expected to greatly facilitate comparability of results and models obtained by various groups and, eventually, to help define a common evaluation metric.

2. Background

The LinGO Project at CSLI has been conducting research and development in Head-Driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1994) since 1994. In close collaboration with international partners—primarily from Saarbrücken (Germany), Cambridge, Edinburgh, and Sussex (UK), and Tokyo (Japan)—the LinGO Project has developed a broad-coverage, precise HPSG implementation of English (the LinGO English Resource Grammar, ERG; Flickinger, 2000), a framework for semantic composition in large-scale computational grammars (Minimal Recursion Semantics, MRS; Copestake, Lascarides, & Flickinger, 2001), and an advanced grammar development environment (the LKB system; Copestake, 1992, 1999). Through contributions from collaborating partners, a pool of open-source HPSG resources has developed that now includes broad-coverage grammars for several languages, a common profiling and benchmarking environment (Oepen & Callmeier, 2000), and an industrial-strength C^{++} run-time engine for HPSG grammars (Callmeier, 2000). LinGO resources are in use world-wide for teaching, research, and application building. Because of the wide distribution and common acceptance, the HPSG framework and LinGO resources present an excellent anchor point for the Stanford treebanking initiative.

3. A Rich and Dynamic Treebank

The key innovative aspect of the Redwoods approach to treebanking is the anchoring of all linguistic data captured in the treebank to the HPSG framework and a generally-available broad-coverage grammar of English, viz. the LinGO English Resource Grammar, combined with tools for the extraction of various, user-defined representations and a software environment to continuously update the treebank as part of the on-going grammar maintenance and extension. Unlike existing treebanks, there will be no need to define a (new) form of grammatical representation specific to the treebank. Instead, the treebank will record complete syntacto-semantic analyses as defined by the LinGO ERG and provide tools to extract many different types of linguistic information at greatly varying granularity.

In particular, the project centrally draws on the [incr tsdb()] profiling environment (essentially a specialized database recording fine-grained parsing results obtained from diverse HPSG systems; Oepen & Carroll, 2000), constructing the treebank as an extension of the existing data model and tools. In turn building on a pre-existing tree comparison tool in the LKB (similar in kind to the SRI Cambridge TreeBanker; Carter, 1997), the treebanking environment presents annotators, one sentence at a time, with the full set of analyses produced by the grammar. Using the tree comparison tool, annotators can quickly navigate through the parse forest and identify the correct or preferred analysis in the current context (or, in rare cases, reject all analyses proposed by the grammar). The tree selection tools persents users, who need little expert knowledge of the underlying grammar, with a range of properties that distinguish competing analyses and that are relatively easy to judge. Each such property corresponds to the usage of a particular lexical item, semantic relation, or grammar rule applied to a specific substring to form a constituent; unlike the LFG packed f-structure representations discussed by King, Dipper, Frank, Kuhn, and Maxwell (2000), the set of basic discriminating properties reduces the information presented to annotators to the minimal amount of structure required to completely disambiguate a sentence. All disambiguating decisions made by annotators are recorded in the [incr tsdb()] database and thus become available for (i) later dynamic extraction from the annotated profile or (ii) dynamic propagation into a more recent profile obtained from re-running an extended version of the grammar on the same corpus.

Important innovative research aspects pertaining to this approach to treebanking are (i) enabling users of the treebank to extract information of the type they need and to transform the available representation into a form suited for their needs and (ii) updating the treebank for an enhanced version of the grammar underlying the recorded analyses in an automated fashion, viz. by re-applying the disambiguating decisions to an updated version of the corpus.

Depth of Representation and Transformation of Information Internally, the [incr tsdb()] database records analyses in three different formats, viz. (i) as a derivation tree composed of identifiers of lexical items and constructions used to construct the analysis, (ii) as a traditional phrase structure tree labeled with an inventory of some fifty atomic labels (of the type ‘S’, ‘NP’, ‘VP’ et al.), and (iii) as an underspecified MRS meaning representation. While (ii) will in many cases be similar to the representation found in the Penn Treebank, (iii) subsumses the functor – argument (or tectogrammatical) structure as it is advocated in the Prague Dependency Treebank or the German TiGer corpus. Most importantly, however, representation (i) provides all the information required to replay the full HPSG analysis (e.g. using the original HPSG grammar and one of the open-source HPSG processing environments, e.g. the LKB or PET, which already have been interfaced to [incr tsdb()]). Using the latter approach, users of the treebank are enabled to extract information in whatever representation they require, simply by reconstructing the full analysis and adapting the existing mappings (e.g. the inventory of node labels used for phrase structure trees) to their needs. Figure 1 depicts the internal Redwoods encoding and two export representations derived from existing conversion routines. Labeled phrase structure trees result from reconstructing a derivation (using the original grammar) and matching a user-defined set of underspecified feature structure ‘templates’

Table 1: Redwoods development status as of February 2002: four sets of transcribed and hand-segmented VerbMobil dialogues have been annotated. The columns are, from left to right, the total number of sentences (excluding fragments) for which the LinGO grammar has at least one analysis ('♯'), average length ('‖'), lexical and structural ambiguity ('⊙' and '×', respectively), followed by the last four metrics broken down for the following subsets: sentences (i) for which the annotator rejected all analyses (no active trees), (ii) where annotation resulted in exactly one preferred analysis (one active tree), (iii) those where full disambiguation was not accomplished through the first round of annotation (more than one active tree), and (iv) massively ambiguous sentences that have yet to be annotated.

corpus	**total**				**active = 0**				**active = 1**				**active > 1**				**unannotated**			
	♯	‖	⊙	×	♯	‖	⊙	×	♯	‖	⊙	×	♯	‖	⊙	×	♯	‖	⊙	×
VM6	2422	7·7	4·2	32·9	218	8·0	4·4	9·7	1910	7·0	4·0	7·5	80	10·0	4·8	23·8	214	14·9	4·3	287·5
VM13	1984	8·5	4·0	37·9	175	8·5	4·1	9·9	1491	7·2	3·9	7·5	85	9·9	4·5	22·1	233	14·1	4·2	212·0
VM31	1726	6·2	4·5	22·4	164	7·9	4·6	8·0	1360	6·6	4·5	5·9	61	10·1	4·2	14·5	141	13·5	4·7	201·5
VM32	608	7·4	4·3	25·6	46	9·8	4·1	18·3	516	7·5	4·4	9·2	21	10·4	3·9	29·6	25	16·6	4·8	375·4

against the HPSG feature structure at each node in the tree. The elementary dependency graph, on the other hand, is an abstraction from the full MRS meaning representation associated to each full analysis; informally, elementary dependencies correspond to the type of tectogrammatical representations found in the Prague Dependency Treebank or the German TiGer corpus and, likewise, resemble the basic relations suggested for parser evaluation by Carroll, Briscoe, and Sanfilippo (1998). Given a rich body of MRS manipuation and conversion software, it is relatively straightforward to adapt the type and form of elementary dependencies to user needs.

For evaluation purposes, the existing [incr tsdb()] facilities for comparing across competence and performance profiles can be deployed to gauge results of a (stochastic) parse disambiguation system, essentially using the preferences recorded in the treebank as a 'gold standard' target for comparison. While the concept of a meta-treebank of the type proposed here has been explored in earlier research (e.g. the AMALGAM project at Leeds University in the UK; Atwell, 1996), previous approaches to the dynamic mapping of treebank representations have built on a static, finite set of hand-constructed mappings.

Automating Treebank Construction Although a precise HPSG grammar like the LinGO ERG will typically assign a small number of analyses to a given sentence, choosing among a handful or sometimes a few dozens of readings is time-consuming and error-prone. The project will explore two approaches to automating the disambigutation task, viz. (i) seeding lexical selection from a part-of-speech (POS) tagger and (ii) automated inter-annotator comparison and assisted resolution of conflicts. Ranking lexical ambiguity on the basis of tagger-assigned POS probabilities requires research into generalizations over the rather fine-grained hierarchy of HPSG lexical types and identifying many-to-many correspondences in a standard POS tagset. Conversely, detecting mismatches (i.e. conflicts) between disambiguating decisions made for the same input sentence by two independent annotators will facilitate research into the linguistic nature of the discriminating properties used and existing logical relations (inclusion, implication, inconsistency et al.) among subsets of discriminators. To exemplify the nature of these properties, consider the sentence

(1) *Have her report on my desk by Friday!*

which is (correctly) assigned thirty two readings by the HPSG grammar; while human language users (and correspondingly human annotators) will typically not note most of the alternative analyses, one can contextualize the sentence to emphasize either one of the following ambiguities: the causative vs. possessive *have*, the determiner vs. personal pronoun *her*, the noun vs. verb *report*, the temporal vs. locative preposition *by*, and *Friday* as a day of the week vs. as a proper noun (e.g. the name of a bar). Using the tree comparison tool and our notion of elementary discriminators, annotators can reduce the set of analyses quickly (where full disambiguation requires minimally four decisions for this example); yet, a POS tagger will reliably assign high probability to the pairings ⟨*her*, determiner⟩ and ⟨*report*, noun⟩ which could be used to bias the presentation to annotators.

Treebank Maintenance and Evolution Perhaps the most challenging research aspect of the Redwoods initiative is about developing a methodology for automated updates of the treebank to reflect the continuous evolution of the underlying linguistic framework and of the LinGO grammar. Again building on the notion of elementary linguistic discriminators, it is expected to explore the semi-automatic propagation of recorded disambiguating decisions into newer versions of the parsed corpus. While it can be assumed that the basic phrase structure inventory and granularity of lexical distinctions have stabilized to a certain degree, it is not guaranteed that one set of discriminators will always fully disambiguate a more recent set of analyses for the same utterance (as the grammar may introduce additional distinctions), nor that re-playing a history of disambiguating decisions will necessarily identify the correct, preferred analysis for all sentences. Once more, a better understanding into the nature of discriminators and relations holding among them is expected to provide the foundations for an update procedure that, ultimately, should be fully automated or at least require minimal manual inspection.

Scope and Current State of Seeding Initiative The first 10,000 trees to be hand-annotated as part of the kick-off initiative are taken from a domain for which the English Resource Grammar is known to exhibit broad and accurate coverage, viz. transcribed face-to-face dialogues in an appointment scheduling and travel arrangement domain. Corpora of some 50,000 such utterances are readily available from the VerbMobil project (Wahlster, 2000) and have already been studied extensively among researchers worldwide in the field. For the follow-up phase of the project, it is expected to move into a second domain and text genre, presumably more formal, edited text taken from newspaper text or another widely available on-line source. As of April 2002, the seeding initiative is well underway. The integrated treebanking environment, combining [incr tsdb()] and the LKB tree selection tool, has been established and has been deployed in a first iteration of annotating a corpus of 10,000 VerbMobil utterances. For a second-year Stanford undergraduate in linguistics, the approach to parse selection through minimal discriminators turned out to be not at all hard to learn and required less training in specifics of the grammatical analyses delivered by the LinGO grammar than could have been expected.

Table 1 summarizes the current Redwoods development status; while annotation of a residual fraction of highly ambiguous sentences and inter-annotator cross-validation continue, the current development snapshot of the treebank can be made available upon request. We have just started work on stochastic parse selection models for the Redwoods treebank, so far obtaining a parse selection accuracy of around eighty per cent from a combination of existing methods applied to the Redwoods derivation trees and elementary dependency graphs (see Figure 1); details on Redwoods parse selection results will be reported in separate publications.

4. Related Work

To our best knowledge, no prior research has been conducted exploring both the linguistic depth, flexibility in available information, and dynamic nature of treebanks as proposed presently. Earlier work on building corpora of hand-selected analyses relative to an existing broad-coverage grammar was carried out at Xerox PARC, SRI Cambridge, and Microsoft Research; as all these resources are tuned to proprietary grammars and analysis engines, the resulting treebanks are not publicly available, nor have research results reported been reproducible. Yet, especially in the light of the successful LinGO open-source repository, it seems vital that both the treebank and associated processing schemes and stochastic models be made available to the general (academic) public.

An on-going initiative at Rijksuniversiteit Groningen (NL) is developing a treebank of dependency structures (Mullen, Malouf, & Noord, 2001), as they are derived from an HPSG-like grammar of Dutch (Bouma, Noord, & Malouf, 2001). While the general approach resembles the Redwoods initiative (specifically the discriminator-based method used in selecting trees from the set of analyses proposed by the grammar; the LKB tree selection tool was originally developed by Malouf, after all), there are three important differences. Firstly, the Groningen decision to compose the treebank from dependency structures commits the resulting resource to a single stratum of representation, tectogrammatical structure essentially, and thus eliminates some of the flexibility in extracting various types of linguistic structure that the Stanford initiative foresees. Secondly, and in a similar vein, recording dependency structures means that the (stochastic) disambiguation component has to consider two syntactically different analyses equivalent whenever they project identical dependency structures; hence, there is a mismatch of granularity between the disambiguated treebank structures and the primary structures (i.e. derivation trees) constructed by the grammar. Finally, the Groningen initiative is making the assumption that the dependency structures, once they are stored in the treebank, are correct and do not change over time (or as an effect of grammar evolution); from the available publications, at least, there is no evidence that the disambiguating decisions made by annotators are recorded in the treebank or that the project expects to dynamically update the treebank with future revisions of the underlying grammar.

Another closely related approach is the work reported by Dipper (2000), essentially the application of a broad-coverage LFG grammar for German to constructing tectogrammatical structures for the TiGer corpus. While many of the basic assumptions about the value of a systematic, broad-coverage grammar for the treebank construction are shared, the strategy followed by Dipper (2000) exhibits the same limitations as the Groningen initiative: the TiGer target representation, still, is mono-stratal and the approach to hand-disambiguation and subsequent transfer of result structures into the TiGer corpus looses the linkage to the original analyses and basic properties used in the disambiguation, hence the potential for dynamic adaptation of the data or automatic updates.

Acknowledgements

The Redwoods initiative is part of the LinGO Laboratory at CSLI and many people, both at Stanford and at partner sites, have contributed to its design and (given small amounts of resources) relative success so far. Ivan A. Sag, Tom Wasow, Emily M. Bender, Tim Baldwin, John Beavers, and Kathryn Campbell-Kibler all have participated in our regular tree conferences, helping annotators select parses and offering productive critiques on analyses provided by the LinGO grammar. Ann Copestake, John Carroll, Rob Malouf, and Stephan Oepen are the main developers of the LKB and [incr tsdb()] software packages from which the Redwoods treebanking environment has been built and, in various capacities, have influenced the Redwoods approach significantly. During a three-month visit to Stanford, Stuart Shieber has been among the driving forces for applications of the existing development version of the treebank, helping us develop and fine-tune suitable stochastic parse selection models. The Redwoods initiative has been partially funded by an internal opportunity grant from CSLI Stanford and by a donation from YY Technologies.

References

Agresti, A. (1990). *Categorical data analysis.* John Wiley & Sons.

Atwell, E. (1996). Comparative evaluation of grammatical annotation models. In R. Sutcliffe, H.-D. Koch, & A. McElligott (Eds.), *Proceedings of the Workshop on Industrial Parsing of Software Manuals* (pp. 25 – 46). Amsterdam, The Netherlands: Rodopi.

Bouma, G., Noord, G. van, & Malouf, R. (2001). Alpino. Wide-coverage computational analysis of Dutch. In W. Daelemans, K. Sima-an, J. Veenstra, & J. Zavrel (Eds.), *Computational linguistics in the netherlands* (pp. 45 – 59). Amsterdam, The Netherlands: Rodopi.

Callmeier, U. (2000). PET — A platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering, 6 (1) (Special Issue on Efficient Processing with HPSG),* 99 – 108.

Carroll, J., Briscoe, E., & Sanfilippo, A. (1998). Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation* (pp. 447 – 454). Granada, Spain.

Carter, D. (1997). The TreeBanker. A tool for supervised training of parsed corpora. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering.* Madrid, Spain.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence* (pp. 598 – 603). Providence, RI.

Collins, M. J. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics and the 7th Conference of the European Chapter of the ACL* (pp. 16 – 23). Madrid, Spain.

Copestake, A. (1992). The ACQUILEX LKB. Representation issues in semi-automatic acquisition of large lexicons. In *Proceedings of the 3rd ACL Conference on Applied Natural Language Processing* (pp. 88 – 96). Trento, Italy.

Copestake, A. (1999). *The (new) LKB system.* (CSLI, Stanford University: `http://www-csli.stanford.edu/~aac/doc5-2.pdf`)

Copestake, A., Lascarides, A., & Flickinger, D. (2001). An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics.* Toulouse, France.

Dipper, S. (2000). Grammar-based corpus annotation. In *Workshop on linguistically interpreted corpora linc-2000* (pp. 56 – 64). Luxembourg.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering, 6 (1) (Special Issue on Efficient Processing with HPSG),* 15 – 28.

Hajic, J. (1998). Building a syntactically annotated corpus. the Prague dependency treebank. In *Issues of valency and meaning* (pp. 106 – 132). Prague, Czech Republic: Karolinum.

Harris, T. E. (1963). *The theory of branching processes.* Berlin, Germany: Springer.

Johnson, M., Geman, S., Canon, S., Chi, Z., & Riezler, S. (1999). Estimators for stochastic ‘unification-based’ grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics* (pp. 535 – 541). College Park, MD.

King, T. H., Dipper, S., Frank, A., Kuhn, J., & Maxwell, J. (2000). Ambiguity management in grammar writing. In *Workshop on linguistic theory and grammar implementation* (pp. 5 – 19). Birmingham, UK.

Manning, C. D., & Carpenter, B. (2000). Probabilistic parsing using left corner language models. In H. Bunt & A. Nijholt (Eds.), *Advances in probabilistic and other parsing technologies* (pp. 105 – 124). Kluwer Academic Publishers.

Marcus, M. P., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics, 19,* 313 – 330.

Mullen, T., Malouf, R., & Noord, G. van. (2001). Statistical parsing of Dutch using Maximum Entropy models with feature merging. In *Proceedings of the Natural Language Processing Pacific Rim Symposium.* Tokyo, Japan.

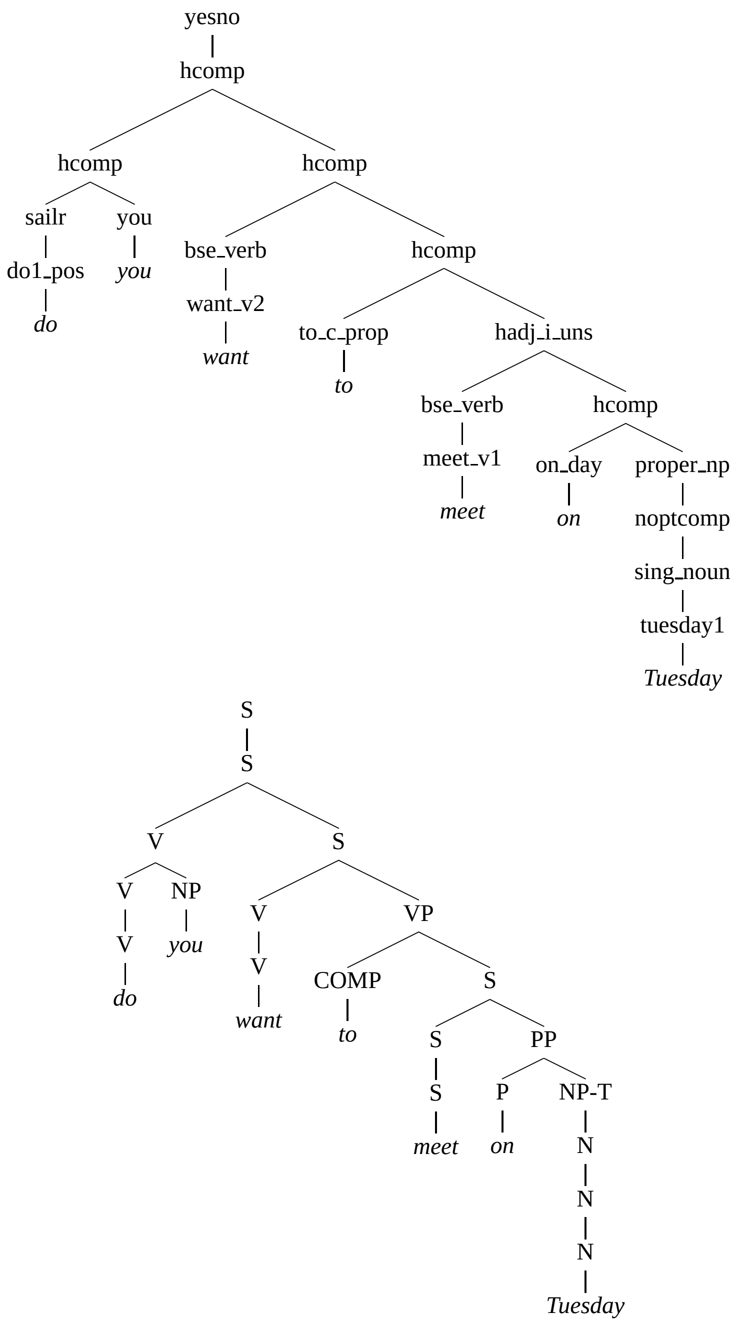
Oepen, S., & Callmeier, U. (2000). Measure for measure: Parser cross-fertilization. Towards increased component comparability and exchange. In *Proceedings of the 6th International Workshop on Parsing Technologies* (pp. 183 – 194). Trento, Italy.

Oepen, S., & Carroll, J. (2000). Performance profiling for parser engineering. *Natural Language Engineering, 6 (1) (Special Issue on Efficient Processing with HPSG),* 81 – 97.

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar.* Chicago, IL and Stanford, CA: The Univeristy of Chicago Press and CSLI Publications.

Skut, W., Krenn, B., Brants, T., & Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing.* Washington, DC.

Wahlster, W. (Ed.). (2000). *Verbmobil. Foundations of speech-to-speech translation.* Berlin, Germany: Springer.



```
_4:{
   _4:int_rel[SOA e2:_want2_rel]
  e2:_want2_rel[ARG1 x4:pron_rel, ARG4 _2:hypo_rel]
  _1:def_rel[BV x4:pron_rel]
  _2:hypo_rel[SOA e18:_meet_v_rel]
  e18:_meet_v_rel[ARG1 x4:pron_rel]
  e19:_on_temp_rel[ARG e18:_meet_v_rel, ARG3 x21:dofw_rel]
  x21:dofw_rel[NAMED :tue]
  _3:def_np_rel[BV x21:dofw_rel]
 }
```

Figure 1: Native and derived Redwoods representations for the sentence *Do you want to meet on Tuesday?* — (a) derivation tree using unique rule and lexical item identifiers of the source grammar (top), (b) phrase structure tree labelled with user-defined, parameterizable category abbreviations (center), and (c) elementary dependency graph extracted from MRS meaning representation (bottom).

A Test of the Leaf-Ancestor Metric for Parse Accuracy

Geoffrey Sampson and Anna Babarczy

School of Cognitive and Computing Sciences
University of Sussex
Falmer, Brighton BN1 9QH, England
{geoffs, annab}@cogs.susx.ac.uk

Abstract

The GEIG metric for quantifying accuracy of parsing became influential through the Parseval programme, but many researchers have seen it as unsatisfactory. The LA metric, first developed in the 1980s, arguably comes closer to formalizing our intuitive concept of relative parse accuracy. We support this claim via an experiment which contrasts the performance of alternative metrics on the same body of automatically-parsed examples. The LA metric has the further virtue of providing straightforward indications of the location of parsing errors.

1. Introduction

One of us (Sampson 2000) has argued that what we call the Òleaf-ancestorÓ (LA) metric is better than the Grammar Evaluation Interest Group (GEIG) metric used in the Parseval competition series (e.g. Black et al. 1991) as a way of quantifying the accuracy of automatic parses, in a context where gold-standard parses using a known scheme of node labelling are available. This paper presents an experiment comparing the performance of the two metrics on a sample of automatic-parser output.

The GEIG metric, which counts the numbers of tagmas (multi-word grammatical units) correctly and incorrectly identified, from our point of view lays excessive weight on locating the exact boundaries of constructions.

As originally defined by Black et al. and as it is often applied, the GEIG metric takes no account of node labels at all: it only considers the location of brackets. And in consequence, this metric includes no concept of approximate correctness in identifying tagmas: a pair of brackets either enclose a sequence of words (or other terminal elements) exactly corresponding to a sequence bracketed off in the gold-standard parse, or not. The result is that Òit is unclear as to how the score on [the GEIG] metric relates to success in parsingÓ (Bangalore et al. 1998).

More recently (Magerman 1995, Collins 1997) a refined variant of the GEIG metric has been used which does check label identity as well as wordspan identity in matching tagmas between gold-standard and candidate parses. We shall argue that even this variant of the GEIG metric is inferior to the LA metric. We shall refer to the Black et al. (1991) and Collins (1997) variants of the GEIG metric as GEIG/unlabelled and GEIG/labelled respectively.

We think of ÒparsingÓ as determining what kind of larger elements are constituted by the small elements of a string that are open to direct observation. Identifying the exact boundaries of the larger elements is a part, but only one part, of that task. If, for instance, in the gold standard, words 5 to 14 are identified as a noun phrase, then a candidate parse which identifies a noun phrase as beginning at word 5 but ending at word 13, or word 15, should in our view be given substantial though not full credit; under the GEIG metric it is given no credit. The LA metric quantifies accuracy of parsing in this sense.

Incidentally, we believe that the LA metric was the earliest parse-assessment metric in the field, having been used, and briefly described in print, in the 1980s (Sampson, Haigh, & Atwell 1989: 278), though it was later eclipsed by the influential Parseval programme.

2. The Essence of Leaf-Ancestor Assessment

The LA metric evaluates the parsing of an individual terminal element in terms of the similarity of the ÒlineagesÓ of that element in candidate and gold-standard parse trees, where a lineage is essentially the sequence of node-labels for nodes on the unique path between the terminal element and the root node. The LA value for the parsing of an entire sentence or other many-word unit is simply the average of the values for the individual words. Apart from (we claim) yielding figures for parsing accuracy of complete sentences which succeed better than the GEIG metric in quantifying our intuitions about parse accuracy, the LA metric has the further practical virtue of identifying the location of parsing errors in a straightforward way.

We illustrate the general concept of LA assessment using one of the shortest sentences in our experimental data-set. (The nature of that data-set, and the gold-standard parsing scheme, will be discussed below.) The sentence runs *two tax revision bills were passed*. (Certain typographic details, including capitalization, inverted commas, and sentence-final punctuation marks, have been eliminated from the examples.) The gold-standard analysis, and the candidate analysis produced by an automatic parser, are respectively:

1G [S [N1 *two* [N1 *tax revision*] *bills*] *were passed*]
1C [S [NP *two tax revision bills*] *were passed*]

(Here and below, Ò*n*GÓ and Ò*n* CÓ label gold-standard and candidate analyses for an example *n*.)

The automatic parser has failed to identify *tax revision* as a unit within the tagma headed by *bills*, and it has labelled that tagma NP rather than N1. Lineages for these tree structures are as follows, where for each terminal element the gold-standard lineage is shown to the left and the candidate lineage to the right of the colon, and within each of the paired lineages the Leaf end is to the Left and the Root end to the Right:

two	N1 [S : NP [S
tax	[N1 N1 S : NP S
revision	N1] N1 S : NP S
bills	N1] S : NP] S
were	S : S
passed	S] : S]

The only aspect of the relationship between this notation and the tree structures which is not self-explanatory is the inclusion of boundary markers (left and right square brackets) in many of the lineages. These are included in accordance with the following rules:

- a left-boundary symbol is inserted in the lineage of a terminal element immediately before the label of the highest nonterminal beginning with that element, if there is such a nonterminal

- a right-boundary symbol is inserted in the lineage of a terminal element immediately after the label of the highest nonterminal ending with that element, if there is such a nonterminal

The reason for including these elements in lineages is that, without them, a set of lineages would not always uniquely determine a tree structure; for instance the structures ÒP [Q *a b*] [Q *c*]]Óand ÒP [Q *a b c*]]Ó would not be distinguishable, since the lineage for each terminal element in both cases would consist of the sequence Q P. A set of lineages in which boundary markers have been inserted by these rules uniquely determines the tree structure from which it is derived.

Thus, in the example above, the LA metric equates the accuracy with which the word *two* has been parsed with the degree of similarity between the two strings NP˚[S and N1 [S, it equates the parse-accuracy for *tax* with the degree of similarity between NP S and [˚N1˚N1˚S, and so forth; for the last two words the lineages are identical, so the metric says that they have been parsed perfectly. We postpone discussion of our method for calculating string similarity until after we have discussed our experimental material.

3. The Experimental Material

Our experiment used a set of sentences from genre sections A and G of the SUSANNE Treebank (Sampson 1992) parsed by an automatic treebanker developed at the Universities of Cambridge and Sussex by Ted Briscoe and John Carroll; of those sentences for which the treebanker was able to produce a structure, a set of 500 was randomly chosen. For the purposes of illustrating the performance of the LA metric and comparing it with the GEIG metric, we wanted material parsed by a system that used a simple parsing scheme with a smallish vocabulary of nonterminal labels, and which made plenty of mistakes in applying the scheme to real-life data; there is no suggestion that the parses in our experimental data-set represent the Òstate of the artÓfor automatic parsing.

The parsing scheme which the automatic parser was intended to apply used seven nonterminal labels, which we gloss with our own rather than Briscoe and CarrollÕs labels:

S	finite clause
VP	nonfinite clause
NP	noun phrase containing specifier
N1	noun phrase without specifier
PP	prepositional phrase
AP	adjectival or adverbial phrase
T	Òtextual constituentÓ, defined by Briscoe and Carroll as a tagma enclosing Òa sequence of sub-constituents whose relationship is syntactically indeterminate due to the presence of intervening, delimiting punctuationÓ

The use of these seven categories is defined in greater detail in documentation supplied to us, but for present purposes it is unnecessary to burden the reader with this material. The automatic-parser output occasionally included node-labels not on the above list (e.g. V, N2), but these were always regarded by the developers of the parser as mistakes.

Briscoe and CarrollÕs original data included GEIG/unlabelled precision and recall scores for each automatic parse, assessed against the SUSANNE bracketing as gold standard. For the purposes of this experiment, the Evalb program (Sekine & Collins 1997) was used to produce GEIG/labelled precision and recall figures for the same data. In order to be able to compare our LA scores with single GEIG/labelled and GEIG/unlabelled scores for each sentence, we converted pairs of precision (P) and recall (R) figures to F-scores (van Rijsbergen 1979) by the formula $F = 2PR/(P + R)$, there being no reason to include a weighting factor to make precision accuracy count more than recall accuracy or *vice versa*.

One of us (Babarczy) constructed a set of gold-standard trees that could be compared with the trees output by the automatic parser, by manually adding labels from the seven-element Briscoe and Carroll label vocabulary to the SUSANNE bracketing, in conformity with the documentation on that seven-element vocabulary. Because the parsing scheme which the automatic parser was intended to apply was very different from the SUSANNE scheme, to a degree this was an artificial exercise. In some cases, none of the seven labels was genuinely suitable for a particular SUSANNE tagma; but one of them was assigned anyway, and such assignments were made as consistently as possible across the 500-sentence data-set. The admitted artificiality of this procedure did not seem unduly harmful, in the context of an investigation into the performance of a metric (as opposed to an investigation into the substantive problem of automatic parsing).

4. Calculation of Lineage Similarity

Leaf-ancestor assessment depends on quantifying the similarity between pairs of strings, which is done in terms of a variant of Levenshtein distance (Levenshtein 1966), also called edit distance. The Levenshtein distance between two strings is the minimum cost for a set of insert, delete, and replace operations to transform one string into the other, where each individual operation has a cost of one. For instance, the Levenshtein distance between A B C B D and A D C B is two: the latter string can obtained from the former by replacing the second character with D and deleting the last character.

We define similarity between candidate and gold-standard lineages in terms of a variant of Levenshtein distance, in which the cost of a replace operation is not fixed but varies over the interval (0, 2) depending on an application-defined concept of similarity between the two symbols involved in a replacement. In the present experiment, replacement of a symbol by an unrelated symbol costs 2; replacement of a symbol by a different symbol sharing the same first character (e.g. NP for N1 or *vice versa*) costs 1.5. The intuition here is that if two grammatical categories are entirely dissimilar, then for a parser to mistake one for the other amounts to two separate errors of failing to recognize the existence of a tagma of one kind, and falsely positing the existence of another type of tagma (a delete and an insert); but partial credit ought to be given for mistaking, say, a noun phrase for an N-bar. (When LA assessment is deployed in practice, the symbol-replacement cost function should be chosen by reference to the nature of the particular scheme of parsing categories, and to the goals of the application for which parsing is needed.)

If len(*s*) is the length of a string *s*, and ML(*s*, *t*) is the modified Levenshtein distance (under some chosen symbol-replacement cost function) between string *s* and string *t*, then the similarity between candidate and gold-standard lineages *c*, *g* for a given terminal element is computed as 1 — ML(*c*, *g*)/(len(*c*) + len(*g*)), which for any *c*, *g* must fall on the interval (0, 1). The accuracy of a candidate parse is defined as the mean similarities of the lineage-pairs for the various words or other terminal elements of the string.

Applied to our short example sentence, this metric gives the scores for successive terminal elements shown in the left-hand column below:

0.917	*two*	`N1 [ S : NP [ S`
0.583	*tax*	`[ N1 N1 S : NP S`
0.583	*revision*	`N1 ] N1 S : NP S`
0.917	*bills*	`N1 ] S : NP ] S`
1.000	*were*	`S : S`
1.000	*passed*	`S ] : S ]`

The average for the whole sentence is 0.833. For comparison, the GEIG/unlabelled and GEIG/labelled *F*-scores are 0.800 and 0.400.

5. Are the Metrics Equivalent?

The figure for the LA metric just quoted happens to be very similar to one of the two GEIG figures. An obvious initial question about the performance of the metrics over the data-set as a whole is whether, although the metrics are calculated differently, they perhaps turn out to impose much the same ranking on the candidate parses.

To this the answer is a clear no. An extreme case is (2):

2G [S *it is not* [NP *a mess* [S *you can make sense of*]]]
G12:0520.27
2C [S *it is not* [NP *a mess* [S *you can make sense*]] *of*]
(0.333, 0.333, 0.952)

(Here and below, gold-standard analyses are followed by the SUSANNE location code of the first word — omitting the last three characters, this is the same as the Brown Corpus text and line number. The location for example (1) was A02:0790.51. Candidate parses are followed in brackets by their GEIG/unlabelled, GEIG/labelled, and LA scores, in that order.)

The LA and GEIG numerical scores are very different; but more important than the raw scores are the rankings assigned by the respective metrics, relative to the range of 500 examples. For both GEIG/labelled and GEIG/unlabelled, the parse of (2) is in the tenth (i.e. the lowest) decile; for LA, it is in the second decile. (The Ò*n*th decileÓ for any of the metrics, refers to the set of examples occuping ranks 50(*n* — 1) + 1 to 50*n*, when the 500 candidate parses are ordered from best to worst in terms of score on that metric; for instance the second decile is the set of examples ranked 51 to 100. Where a group of examples share identical scores on a metric, for purposes of dividing the examples into deciles an arbitrary order was imposed on members of the group.)

The difference between the low GEIG scores and the high LA score for example (2) is a classic illustration of one of the standard objections to the GEIG metric. The parser has correctly discovered that the sentence consists of an S within NP within S structure, and almost every word has been given its proper place within that structure; just one word has been attached at the wrong level, but because this leads to the right-hand boundary of two of the three tagmas being slightly misplaced, the GEIG score is very low. We believe that most linguistsÕ intuitive assessment of example (2) would treat it as a largely-correct parse with one smallish mistake, not as one of the worst parses in the data-set — that is, the intuition would agree much better with the LA metric than with the GEIG metrics in this case.

An extreme case in the opposite direction (LA score lower than GEIG score) is (3):

3G [S *yes* , [S *for they deal* [PP *with distress*]]]
G12:1340.42
3C [T *yes* , [PP *for they deal* [PP *with distress*]]] (1.0, 0.333, 0.262)

The GEIG/unlabelled metric gives 3C a perfect mark — all brackets are in the right place; but its LA score is very low, because two of the three tagmas are wrongly labelled — 0.262 is in fact by a clear margin the lowest LA score for any of the 500 examples. One might of course debate about whether, in terms of the Briscoe/Carroll labelling scheme, the root tagma *should* be labelled S rather than T, but that is not to the point here. The relevant point is that, *if* the target is the gold-standard parse shown, then a metric which gives a poor score to 3C is performing better than a metric which gives it a perfect score.

For example (3), GEIG/labelled performs much better than GEIG/unlabelled. Where parsing errors relate wholly or mainly to labelling rather than to structure, that will be so. But we have seen in the case of (2), and shall see again, that there are other kinds of parsing error where GEIG/labelled is no more, or little more, satisfactory than GEIG/unlabelled.

6. Performance Systematically Compared

In order systematically to contrast the performance of the metrics, we need to focus on examples for which the ranking of the candidate parse is very different under the different metrics, which implies checking cases whose parses are among the lowest-ranked by one of the metrics. It would be little use to check the highest-ranked parses by either metric. Many candidates are given perfect marks by the LA metric, because they are completely accurate, in which case they will also receive perfect GEIG marks. Some candidates receive perfect GEIG/unlabelled marks but lower LA (and GEIG/labelled) marks, however this merely reflects the fact that GEIG/unlabelled ignores labelling errors.

We have checked how many examples from the lowest GEIG/unlabelled and GEIG/labelled deciles fall into the various LA deciles, and how many examples from the lowest LA decile fall into the various GEIG/unlabelled and GEIG/labelled deciles. These are the results:

LA deciles for GEIG/unlabelled 10th decile:

1st	0
2nd	1
3rd	3
4th	2
5th	2
6th	4
7th	8
8th	7
9th	11
10th	12

LA deciles for GEIG/labelled 10th decile:

1st	0
2nd	1
3rd	1
4th	0
5th	0
6th	1
7th	7
8th	8
9th	13
10th	19

GEIG/unlabelled deciles for LA 10th decile:

1st	1
2nd	0
3rd	4
4th	9
5th	3
6th	8
7th	4
8th	5
9th	4
10th	12

GEIG/labelled deciles for LA 10th decile:

1st	0
2nd	0
3rd	1
4th	1
5th	3
6th	8
7th	5
8th	5
9th	8
10th	19

Clearly there is a tendency for parses assigned poor LA scores also to be assigned poor GEIG scores, and *vice versa*. If there were not, at least one of the metrics could never have been taken seriously by anyone. But there are many exceptions.

The GEIG/unlabelled and GEIG/labelled 10th-decile, LA 2nd-decile example is (2), already discussed above. The GEIG/labelled 10th-decile, LA 3rd-decile example (and this is also one of the GEIG/unlabelled 10th-decile, LA 3rd-decile examples) is (4):

4G [S *then he began* [VP *to speak* [PP *about* [NP *the tension* [PP *in art*] [PP *between* [NP *the mess* [N1 *and form*]]]]]]] G12:0610.27
4C [S *then he began* [VP *to speak* [PP *about* [NP *the tension*]] [PP *in* [N1 *art* [PP *between* [NP *the mess*]] [N1 *and form*]]]]] (0.353, 0.353, 0.921)

The two further GEIG/unlabelled 10th-decile, LA 3rd-decile cases are:

5G [S *Alusik then moved Cooke across* [PP *with* [NP *a line drive* [PP *to left*]]]] A13:0150.30
5C [S *Alusik then moved Cooke across* [PP *with* [NP *a line drive*]] [PP *to left*]] (0.500, 0.500, 0.942)

6G [S [NP *their heads*] *were* [PP *in* [NP *the air*]] *sniffing*] G04:0030.18
6C [S [NP *their heads*] *were* [PP *in* [NP *the air sniffing*]]] (0.500, 0.500, 0.932)

Examples (5) and (6) are essentially similar to (2) above, since all three concern errors about the level at which a sentence-final element should be attached. The LA scores are marginally lower than for (2), because the misattached elements comprise a higher proportion of total words in the respective examples. In (5) a two-word phrase rather than a single word is misattached, and in (6) the misattached element is a single word but the sentence as a whole is only seven words long while example (2) contains ten words. Intuitively it is surely appropriate for a misattachment error involving a higher proportion of total words to be given a lower mark, but for these candidates nevertheless to be treated as largely correct. (Notice that the candidate parse for (5) verges on being a plausible alternative interpretation of the sentence, i.e. not mistaken at all. It is only the absence of *the* before *left* which, to a human analyst, makes it rather certain that our gold-standard parse is the structure corresponding to the writerÕs intended meaning.)

The candidate parse for (4) contains a greater variety of errors, and we would not claim that in this case it is so intuitively clear that the candidate should be ranked among the above-average parses. Notice, though, that although several words and tagmas have been misattached, nothing has been identified as a quite different kind of tagma from what it really is (as *for they deal with distress* in (3) above was identified as a prepositional phrase rather than subordinate clause). Therefore our intuitions do not clearly suggest that the candidate should be ranked as worse than average, either; our intuitions are rather indecisive in this case. In other cases, where we have clear intuitions, the LA ranking agrees with them much better than the GEIG ranking.

Turning to cases where LA gives a much lower ranking than GEIG: the most extreme case is (3), already discussed. The LA 10th-decile, GEIG/labelled 3rd decile case (which is also one of the GEIG/unlabelled 3rd-decile cases) is (7):

7G [S [NP *its ribs*] *showed* , [S *it was* [NP *a yellow nondescript color*]] , [S *it suffered* [PP *from* [NP *a variety* [PP *of sores*]]]] , [S *hair had scabbed* [PP *off* [NP *its body*]] [PP *in patches*]]] G04:1030.15

7C [T [S [NP *its ribs*] *showed*] , [S *it was* [NP *a yellow nondescript color*]] , [S *it suffered* [PP *from* [NP *a variety* [PP *of sores*]]]] , [S *hair had scabbed off* [NP *its body*] [PP *in patches*]]] (0.917, 0.833, 0.589)

There are large conflicts between candidate and gold-structure parses here: the candidate treats the successive clauses as sisters below a T node, whereas under the SUSANNE analytic scheme the gold-standard treats the second and subsequent clauses as subordinate to the first, with no T node above it; and the candidate fails to recognize *off* as introducing a PP. The presence v. absence of the T node, because it occurs at the very top of the tree, affects the lineages of all words and hence has a particularly large impact on LA score (Sampson 2000: 66 discussed this property of the LA metric).

The three other LA 10th-decile, GEIG/unlabelled 3rd-decile cases are:

8G [S [S *when* [NP *the crowd*] *was asked* [S *whether it wanted* [VP *to wait* [NP *one more term*] [VP *to make* [NP *the race*]]]]] , *it voted no* —[S *and there were* [NP *no dissents*]]] A01:0980.06

8C [T [S [PP *when* [NP *the crowd*] *was asked* [PP *whether it wanted* [VP *to wait* [NP *one more term*] [VP *to make* [NP *the race*]]]]]] , *it voted no*] — [S *and there were* [NP *no dissents*]]] (0.952, 0.667, 0.543)

9G [S [S *we wo +n’t know* [NP *the full amount*] [S *until we get* [NP *a full report*]]] , *Wagner said*] A09:0520.12

9C [T [S *we wo +nÕt know* [NP *the full amount*] [PP *until we get* [NP *a full report*]]] , [S *Wagner said*]] (0.909, 0.545, 0.531)

10G [S [S [NP *her husband*] *was lying* [PP *on* [NP *the kitchen floor*]]] , *police said*] A19:1270.48

10C [T [S [NP *her husband*] *was lying* [PP *on* [NP *the kitchen floor*]]] , [S *police said*]] (0.909, 0.727, 0.627)

Each of these involves the same problems as (7) of presence v. absence of a root T node and co-ordinate v. subordinate relationships between successive clauses. Example (8) includes further large discrepancies: *when* and *whether* are both treated as initiating prepositional phrases rather than clauses (though neither word has a standard prepositional use). Example (9) has a similar error involving *until* (this is more understandable, since *until* can be a preposition). Intuitively, to the present authors, the LA metric seems correct in characterizing (8) and (9) as two of the cases where the candidate parse deviates furthest from the gold standard, rather than as two of the best parses. The case of (10) is admittedly less clearcut.

Some readers may find this section unduly concerned with analystsÕintuitions as opposed to objective facts. But, if the question is which of two metrics better quantifies parsing success, the only basis for comparison is peopleÕs intuitive concept of what ought to count as good or bad parsing. Both metrics give perfect scores to perfect matches between candidate and gold-standard structure; which departures from perfect matching ought to be penalized heavily can only be decided in terms of Òeducated intuitionÓ that is intuition supported by knowledge and discussion of the issues. It would not be appropriate to lend such intuitions the appearance of objectivity and theory-independence by Òcounting votesÓ from a large number of subjects. (Since the GEIG metric is widely known, raw numbers from an exercise like that could have as much to do with the extent to which individual informants were aware of that metric as with pre-theoretical responses to parse errors.) Deciding such an issue in terms of averaged subject responses would be as inappropriate as choosing between alternative scientific theories by democratic voting. Rather, the discussion should proceed as we have conducted it here, by appealing to readersÕ and writersÕ individual intuitions with discussion of particular examples.

7. Local Error Information

Different accuracy rankings assigned to complete parses are not the only reason for preferring the LA to the GEIG metric. Another important difference is the ability of the LA metric to give detailed information about the location and nature of parse errors.

Consider, for instance, example (11), whose gold-standard and candidate analyses are:

11G [S *however* , [NP *the jury*] *said* [S *it believes* [S [NP *these two offices*] *should be combined* [VP *to achieve* [N1 *greater efficiency*] [VP *and reduce* [NP *the cost* [PP *of administration*]]]]]]] A01:0210.15

11C [S *however* , [NP *the jury*] *said* [S *it believes* [NP *these two*] [S *offices should be combined* [VP *to* [VP *achieve* [N1 *greater efficiency*] [VP *and reduce* [NP *the cost* [PP *of administration*]]]]]]]] (0.762, 0.667, 0.889)

The score of 0.889 assigned by the LA metric to this candidate analysis, like any LA score, is the mean of scores for the individual words. For this example, those are as follows:

1.000	*however*	[S : [S
1.000	*,*	S : S
1.000	*the*	[NP S : [NP S
1.000	*jury*	NP] S : NP] S
1.000	*said*	S : S
1.000	*it*	[S S : [S S
1.000	*believes*	S S : S S
0.667	*these*	NP [S S S : [NP S S
0.750	*two*	NP S S S : NP] S S
0.667	*offices*	NP] S S S : [S S S
1.000	*should*	S S S : S S S
1.000	*be*	S S S : S S S
1.000	*combined*	S S S : S S S
1.000	*to*	[VP S S S : [VP S S S
0.800	*achieve*	VP S S S : [VP VP S S S
0.923	*greater*	[N1 VP S S S : [N1 VP VP S S S
0.923	*efficiency*	N1] VP S S S : N1] VP VP S S S
0.769	*and*	[S VP S S S : [VP VP VP
0.727	*reduce*	S VP S S S : VP VP VP S S S
0.800	*the*	[NP S VP S S S : [NP VP VP VP S S S
0.769	*cost*	NP S VP S S S : NP VP VP VP S S S
0.824	*of*	[PP NP S VP S S S : [PP NP VP VP VP S S S
0.824	*administration*	PP NP S VP S S S] : PP NP VP VP VP S S S]

The display shows that, according to the LA metric, the early part of the sentence is parsed perfectly, and that the worst-parsed part is *these two offices*. That seems exactly right; in the correct analysis, these three words are a NP, subject of the clause which forms the object of *believe*, but the automatic parser has interpreted *believe* as a ditransitive verb with *these two* as indirect object, and has treated *offices* as grammatically unrelated to *these two*. Intuitively, these are gross mistakes. The next stretch of erroneous parsing is from *achieve* to the end, where each wordÕs mark is pulled down by the error of taking *achieve* to open a subordinate VP within the VP initiated by the preceding *to*. Relative to the SUSANNE scheme used here as gold standard, this also seems a bad error. It is an understandable consequence of the fact that the Briscoe/Carroll automatic parser was based on a parsing scheme that made different theoretical assumptions about grammar, but in the present target scheme no English construction ought to be analysed as ÒVP *to* [VP ...Ó

We would not pretend that our intuitions are so refined that they positively justify a score for *reduce* which is marginally lower than those for the surrounding words, or positively justify the small difference between 0.727 as the lowest score in the second bad stretch and 0.667 in the earlier stretch; the present authorsÕ intuitions are vaguer than that. But notice that, as it stands, the GEIG metric offers no comparable technique for identifying the locations of bad parsing performance within parsed units; it deals in global scores, not local scores. True, one can envisage ways in which the GEIG metric might be developed to yield similar data; it remains to be seen how well that would work. (Likewise, the GEIG/labelled metric could be further developed to incorporate the LA concept of partial matching between label pairs. On the other hand, there is no way that GEIG could be adapted to avoid the unsatisfactory performance exemplified by (2) above.)

Using the LA metric, researchers developing an automatic parser in a situation where large quantities of gold-standard analyses are available for testing should easily be able to identify configurations (e.g. particular grammatical words, or particular structures) which are regularly associated with low scores, in order to focus parser development on areas where further work is most needed. If the parsing scheme used a larger variety of nonterminal labels, one would expect that individual nonterminals might regularly be associated with low scores, though with the very austere nonterminal vocabulary of the Briscoe/Carroll scheme that is perhaps less likely to be so. Even with a small-vocabulary scheme like this, though, one might expect to find generalizations such as Òwhen a subordinate S begins with a multi-word NP, the parser tends to failÓ Note that we are not saying that this is a true generalization about the particular automatic parser used to generate the data under discussion; one such mistake occurs in the example above, but we do not know whether this is a frequent error or a one-off. Any automatic parser is likely to make errors in *some* recurring patterns, though; the LA metric is potentially an efficient tool for identifying these, whatever they happen to be.

8. Conclusion

From these results, we would argue that the leaf-ancestor metric comes much closer than the GEIG metric to operationalizing our intuitive concepts of accurate and inaccurate parsing.

Someone looking for reasons to reject the LA metric might complain, correctly, that the algorithm for calculating it is much more computation-intensive than that for the GEIG metric. But intensive computation is not a problem in modern circumstances.

More important: there is no virtue in a metric that is easy to calculate, if it measures the wrong thing.

9. Acknowledgment

The research reported here was supported in part by the Economic and Social Research Council (UK) under grant R000 23 8746. We are very grateful to John Carroll for supplying us with the experimental material, generating the GEIG/labelled data, and discussing the material with us.

10. References

Bangalore, S., et al. (1998) ÒGrammar and parser evaluation in the XTAG projectÓ In J.A. Carroll et al., eds., *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation, Granada, Spain, 26 May 1998*.

Black, E., et al. (1991) ÒA procedure for quantitatively comparing the syntactic coverage of English grammarsÓ In *Proceedings of the Speech and Natural Language Workshop, DARPA, February 1991, Pacific Grove, Calif.*, pp. 306-11. Morgan Kaufmann.

Collins, M. (1997) ÒThree generative, lexicalised models for statistical parsingÓ In *Proceedings of the 35th Annual Meeting of the ACL and 8th Conference of the*

European Chapter of the ACL, 7-12 July 1997, Madrid. Morgan Kaufmann.

Levenshtein, V.I. (1966) ÒBinary codes capable of correcting deletions, insertions, and reversalsÓ *Soviet Physics — Doklady* 10.707-10 (translation of Russian original published in 1965).

Magerman, D.M. (1995) ÒStatistical decision-tree models for parsingÓ In *Proceedings of the 33rd Annual Meeting of the ACL, 26-30 June 1995, Cambridge, Mass.*, pp. 276-83. Morgan Kaufmann.

van Rijsbergen, C.J. (1979) *Information Retrieval*, 2nd ed. Butterworths (London).

Sampson, G.R. (1992) The SUSANNE Treebank. Up-to-date release available from `www.grsampson.net` Òdownloadable research resourcesÓ

Sampson, G.R. (2000) ÒA proposal for improving the measurement of parse accuracyÓ *International Journal of Corpus Linguistics* 5.53-68.

Sampson, G.R., R. Haigh, & E.S. Atwell (1989) ÒNatural language analysis by stochastic optimizationÓ *Journal of Experimental and Theoretical Artificial Intelligence* 1.271-87.

Sekine, S. & M. Collins (1997) Evalb software at `www.cs.nyu.edu/cs/projects/proteus/evalb/`.

Evaluating parser accuracy using edit distance

Brian Roark

AT&T Shannon Labs
180 Park Avenue
Building 103, Room E145
Florham Park, NJ 07932-0971
email: `roark@research.att.com`

Abstract

This paper examines parser evaluation in terms of edit distance, rather than precision and recall. We demonstrate the potential efficacy of edit distance as an evaluation metric with an example, and compare several edit distance scores to labeled precision and recall scores for several statistical parsers from the literature. We suggest a simple schema that has some nice properties. Finally, we discuss the applicability of edit distance metrics for comparing heterogeneous output.

1. Introduction

The dominant metrics in parsing accuracy evaluation are precision and recall, which might suggest that parsing is a classification task, since these metrics are intended to measure classification accuracy. Another way to view parsing is as a recognition task, more analogous to something like speech recognition – recovering hidden structure over an unlabeled, temporally sequenced signal – than classification. From this perspective, we propose adopting the evaluation metric most widely used for speech recognition, namely string edit distance (Wagner and Fischer, 1974). While there is a literature on tree edit distance (Tai, 1979; Zhang and Shasha, 1989), which involves expanding and contracting labeled edges in a tree structure, we will not approach parser evaluation from this perspective. Rather, we will use the simpler string edit distance between string representations of parses. String edit distance as a general framework is very flexible, which will allow us to score parses on more than simply constituent label and span, and to tailor the edit costs to particular needs. We will use it to evaluate how well a parse matches to gold standard, in terms of (i) a sequence of context-free rules corresponding to the tree; (ii) a sequence of constituents with span information; or (iii) the labeled bracketing itself. We will show that the specific distance metrics we will present have some nice properties: they make similar distinctions between parsers to those of standard labeled precision and recall, when the new measures are applied to statistical parsers trained to produce the same kind of annotation; and they provide a conceptually clean way to encode both partial matches and matches across labeling schemas. This paper is not intended to present a specific approach to evaluating specific parsers; rather a general approach that, in our opinion, has the best chance to provide a unified evaluation framework for the parsing community.

Before continuing further, let us define the terms and metrics that we are going to be using in the course of this paper. A parse can be represented as a labeled bracketing[1], e.g.

```
(S  (NP (DT The) (NN dog) )
    (VP (VBD barked) ) )
```

Each left parenthesis, which we will also refer to as an open bracket, is associated with a label and a right parenthesis, which we will also call a close bracket. Each open and close bracket pair denotes a constituent in the parse tree. Each constituent has a span, which is the part of the string that is at the leaves of the sub-tree rooted at the constituent. For example, the NP constituent in the example spans the words “The” and “dog”. Each local tree in the structure, i.e. the constituent label and the labels of its children, constitutes a context-free rule instance. For example, the highest local tree in the above labeled bracketing is an instance of the rule S → NP VP.

A parser returns a labeled bracketing of a string, which is evaluated with respect to a hand-labeled “gold standard” annotation. The most widely used metrics to evaluate the accuracy of the parser come from the PARSEVAL recommendations (Black et al., 1991); among these, labeled precision and recall are principal. The labeled measures do not count part-of-speech (POS, also known as pre-terminal) labels, which are those non-terminals with one and only one terminal child. In the above example, the accuracy of the DT, NN, and VBD labels would not be included in the labeled precision and recall scores. Of the other constituents in the labeled bracketing (S, NP, and VP), each would be compared with the constituents in the gold standard parse. If they match the label and the span of an otherwise unmatched constituent in the gold standard parse, then they are counted as correct, otherwise incorrect. Labeled precision (LP) is the number of correct constituents divided by the total number of constituents in the parser’s labeled bracketing (excluding, of course, POS). Labeled recall (LR) is the number of correct constituents divided by the total number of constituents in the gold standard labeled bracketing. Often these are combined into a single measure, called the F-measure, according to the following formula:

$$2(\mathrm{LR})(\mathrm{LP})/(\mathrm{LR} + \mathrm{LP}) \qquad (1)$$

[1] The labeled bracketing convention that will be used for examples in this paper are those from the Penn Treebank (Marcus et al., 1993), but that is not intended to restrict the scope of this paper to these annotations.

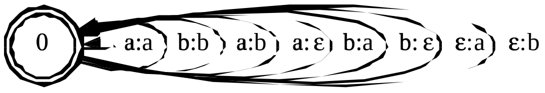


Figure 1: Example edit distance transducer.

The E-measure is 100 minus the F-measure. This will come in handy when comparing precision and recall based measures with error rates based on edit distance. There are other measures in the PARSEVAL recommendations, such as the crossing brackets scores, but for the purpose of this paper, we will focus on labeled precision and recall.

We will present string edit distance as weighted finite-state transduction[2]. Let Σ_0 be the alphabet of the input string $S_0 \in \Sigma_0*$, and Σ_1 be the alphabet of the output string $S_1 \in \Sigma_1*$. Let T be a finite state transducer that maps (i) all $a \in \Sigma_0 \bigcap \Sigma_1$ to themselves at no cost; (ii) all $a \in \Sigma_0$ to $b \in \Sigma_1$ at cost 1 (substitution); (iii) all $a \in \Sigma_0$ to the empty string at cost 1 (deletion); and (iv) the empty string to all $b \in \Sigma_1$ at cost 1 (insertion). Then the edit distance between S_0 and S_1 is the least cost path in $S_0 \circ T \circ S_1$.

Figure 1 presents a transducer for (a+b)*, which, when used according to the previous paragraph, returns the edit distance between two strings. Each arc is labeled with an input symbol i, and output symbol o and a cost c, as follows: i:o/c. The empty string has label ϵ. Only those arcs which map characters to themselves are cost free. All other arcs – substitution (a:b), deletion (a:ϵ), and insertion (ϵ:b) – have cost 1. Thus, for example, the edit distance between an input string "abab" and an output string "aabb" is two[3].

Of course, it is up to us what the alphabets and strings are. In the case of speech recognition, where edit distance is used to calculate the word error rate (WER), the strings are strings of words in the lexicon, and the transduction maps words to words or the empty string. For the case of parser evaluation, we might consider strings of rules, strings of constituents with span, or strings consisting of pieces of the labeled bracketing. It is also up to us what cost to put on the arcs. In the above figure, all edit costs are one, but there may be cases where the cost of an edit might be judged to be less than one.

The rest of the paper will be structured as follows. First we will motivate the use of edit distance with an example where precision and recall give scores that fail to correspond to intuitive notions of similarity, which an edit distance score is closer to. Then we will present three edit distance measures, and how they perform on four parser implementations from the statistical parsing literature. Finally, we will discuss how this approach can handle more complicated schemas that might address some of the cross-system issues.

2. Motivation

One perceived failure of precision and recall measures was presented in Bangalore et al. (1998), namely that shallow parses that do not make strong attachment choices are penalized less than parses that are quite close to the gold standard, but that make some wrong attachment decisions. Their example is repeated here, with node labels on the bracketing.

```
1. (S (NP (PRN She))
      (VP (VBD bought)
          (NP (NP (DT an) (JJ incredib ly)
                 (JJ expensive ) (NN coat))
            (PP (IN with)
              (NP (NP (JJ gold) (NNS buttons))
                  (CC and)
                  (NP (NN fur) (NN lining)))) )
          (PP (IN at)
              (NP (DT the) (NN store)))))
2. (S (NP (PRN She))
      (VP (VBD bought)
          (NP (NP (DT an) (JJ incredib ly)
                 (JJ expensive ) (NN coat))
            (PP (IN with)
              (NP (NP (JJ gold) (NNS buttons))
                  (CC and)
                  (NP (NP (NN fur) (NN lining) )
                      (PP (IN at)
                         (NP (DT the)
                            (NN store)))) )))))
3. (S (NP (PRN She))
      (VP (VBD bought)
          (NP (DT an) (JJ incredibly)
              (JJ expensive ) (NN coat))
          (IN with)
          (NP (JJ gold) (NNS buttons) )
          (CC and)
          (NP (NN fur) (NN lining))
          (PP (IN at)
              (NP (DT the) (NN store)))))
```

The gold standard parse (1) has the final prepositional phrase modifying the verb phrase, while parse 2 mistakenly attaches the PP to the lowest noun phrase. Other than this, though, parse 2 is quite close to the gold standard. Parse 3 is a much shallower parse, with much of the hierarchy in the gold standard left out. Using standard labeled precision and recall against the gold standard parse (1), parse 2 achieves 72.73 labeled recall and 66.67 labeled precision, or 69.56 F-measure. In contrast, parse 3, which is much farther from the gold standard, achieves 72.73 labeled recall and 100.0 labeled precision, or 84.21 F-measure. The clear reason for this is that judging correctness based on span ignores dominance relationships, so that parse 3 identifies many correct constituents, despite getting the immediate dominance relationships quite wrong (according to the gold standard).

One way to include the dominance relationships is to encode the parse as a sequence of rules (top-down, left-most). This uniquely identifies the tree, and serves as a string for input into the simple edit distance approach outlined in the previous section. We will uniformly give a cost of 1 for each insertion, deletion, and substitution, without giving credit for substitutions between more or less similar

[2] For an introduction to finite-state transducers, see, e.g. Jurafsky and Martin (2000).

[3] Note that there is more than one derivation here with the same cost, e.g. a:a b:a a:b b:b (two substitutions) and a:a b:ϵ a:a b:b ϵ:b (one deletion and one insertion).

Gold parse rules	parse (2) rules	(2) edits	edit type	parse (3) rules	(3) edits	edit type
S → NP VP	S→NP VP			S→NP VP		
→ PRN	→ PRN			→ PRN		
→ VBD NP PP	→ VBD NP	1	substitution	→ VBD NP IN NP CC NP PP	1	substitution
→ NP PP	→ NP PP				1	deletion
→ DT JJ JJ NN	→ DT JJ JJ NN			→ DT JJ JJ NN		
→ IN NP	→ IN NP				1	deletion
→ NP CC NP	→ NP CC NP				1	deletion
→ JJ NNS	→ JJ NNS			→ JJ NNS		
	→ NP PP	1	insertion			
→ NN NN	→ NN NN			→ NN NN		
→ IN NP	→ IN NP			→ IN NP		
→ DT NN	→ DT NN			→ DT NN		
Total:		2			4	

Table 1: Edits from gold standard for aligned rule sequences.

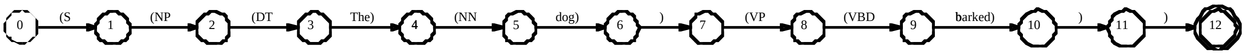


Figure 2: Labeled bracketing encoded as a string

rules. This may appear somewhat coarse, but it is merely a starting point. If it is felt that certain rules are, in some sense, closer than others, then this can be represented in the amount of cost that is associated with that arc in the transducer. The approach is general enough to allow for different costs. To begin, we will count each deletion, substitution, and insertion as one edit, and present the error rate, i.e. the total edit cost per 100 rules of gold standard. This is the same measure that is used for accuracy in the speech recognition community (word error rate).

One note about the encoding: because every left-hand side category (except S at the root) also occurs on a right-hand side, we omit the left-hand side category (except the initial S), so as to avoid doubling errors. Because of the rule ordering, there is no ambiguity with respect to the left-hand sides.

Table 1 shows the aligned sequences of rules, and the edit cost accumulated by both parse 2 and parse 3 under the standard cost. Recall that the symbols that are being compared are entire rules, not individual non-terminals within the rules. Hence each candidate parse has a single substitution for the verb phase expansion, despite the fact that the parse 2 VP expansion is in some sense closer to the original than that of parse 3. In this sense, as was stated above, the metric is somewhat coarse. Nevertheless, we can see that parse 2 accrues one substitution and one insertion, for an error rate of 18.2 (2 edits for 11 rules), whereas parse 3 accrues one substitution and three deletions for a 36.4 error rate. These scores better correspond to our intuition, and the one stated in Bangalore et al. (1998), that parse 2 is closer to the gold standard than parse 3.

What this example demonstrates is that there are circumstances where precision and recall can unduly penalize parses, and others where they do not penalize parses enough. Edit distance between sequences of rules does the right thing, at least in these cases. Note, however, that edit distance does not need to be over sequences of rules. In fact, we can calculate edit distance over constituent spans, by replacing the rules in the table above with the left-hand side and word span positions. In the next section, we will present a third edit distance approach, which treats the labeled bracketing itself as a string. This will be followed by some parser evaluations using all three of our proposed edit distance approaches.

3. Labeled bracketing edit distance

Before presenting our method for calculating edit distance between labeled bracketings, let us first discuss how to encode them as strings, i.e. what are the tokens? We will adopt what was used in Roark (2001b) for storing automatically generated treebanks. Every open bracket has a label, so an open bracket plus its label is one token. Terminal items (i.e. lexical items) are always followed by a close bracket (at least in Penn Treebank annotation), so terminal plus close bracket is one token. Finally, close brackets for non-POS constituents, which do not have an associated terminal item, are each tokens. In this way, the labeled bracketing can be encoded as a string. Figure 2 shows our original labeled bracketing encoded as a string with this tokenization.

To calculate the edit distance between labeled bracketings of the same string, we are going to exploit some characteristics of the bracketing. First, we know that the terminal items are the same in the input and the gold standard. Hence there is no need to provide any arcs for terminal items other than those mapping them to themselves. Next, we assume that both the input and the gold standard are balanced, i.e. there are the same number of right and left parentheses. What this means is that, for every successful mapping between one parse to the other, every deleted open bracket must be matched with either an inserted open bracket or a deleted close bracket, to maintain balanced bracketing. Indeed, any bracket insertion or deletion must be paired with another edit.

	Open Bracket	Close Bracket
Delete	(X:ϵ/0.5	):ϵ/0.5
Insert	ϵ:(X/0.5	ϵ:)/0.5

Table 2: Labeled bracketing edit cost table

If we used a standard edit cost of 1, then we would get a double count for every deleted or inserted constituent. If, however, we make the cost of deleting or inserting open (labeled) brackets or close (non-terminal) brackets one half, we can rely on the balanced brackets property to ensure that we ultimately get a cost of one for the error. Table 2 presents this cost schema. One nice side effect of this is that substitution can be considered simply as a deletion plus an insertion, and hence automatically given an edit cost of 1, without having to explicitly put substitution arcs into the transducer, since deleting a labeled bracket and inserting an alternatively labeled bracket has cost 1 under this schema. We can, however, if we wish, allow substitution at no cost. For example, if we do not wish to include POS tags in the evaluation, we can allow any POS tag to rewrite as any other POS tag with no cost. Since the close brackets for POS tags are terminals, which cannot be inserted or deleted, and since the yield for the two parses are identical, this free substitution means that no POS tagger error will result in any cost for the least cost transduction. Let us refer to as the *basic labeled bracketing edit transducer* that which maps (i) all symbols to themselves at no cost, (ii) all non-POS open bracket non-terminals to epsilon and vice versa at cost 0.5, (iii) the close bracket symbol to epsilon and vice versa at cost 0.5, and (iv) all POS open bracket non-terminals to all other POS open bracket non-terminals at no cost.

There does remain one serious problem with this approach, however. Referring back to our motivating example in the previous section, one can see by inspection that the least cost path under this approach between the gold standard (parse 1) and parse 2 will involve deleting one NP constituent (one open and one close bracket) and inserting and deleting three close brackets. In other words, the misattachment of the PP is costing just as much as it would under precision and recall evaluation. A better solution is to allow certain kinds of constituent movement to count as just one edit. Deleting and inserting consecutive close brackets in effect allows constituents to be moved higher or lower in the hierarchy. Thus, in parse 2, the deletion of the three close brackets from their current location and their insertion into the correct location – how the movement of the constituent is effected – should count as a single edit. We can do this with a simple additional pass over the least cost path through the composition with our basic labeled bracketing edit transducer.

This basic transducer cannot simply be changed to give less cost to deletion and insertion of multiple consecutive close brackets, because those brackets may be paired with open brackets or non-consecutive close brackets. We only want to give reduced cost to those consecutive brackets that are matched with other consecutive brackets. We could, in advance of composition with the basic labeled bracketing edit transducer, pass the input through an arbitrary number of transducers that, for any given k, map the input to an output by deleting k consecutive close brackets and inserting k consecutive close brackets, at a cost of 1. Figure 3 shows one such transducer, for two consecutive brackets. Because an arbitrary number of these would need to apply in order to cover all possible movements, this is not a particularly useful approach.

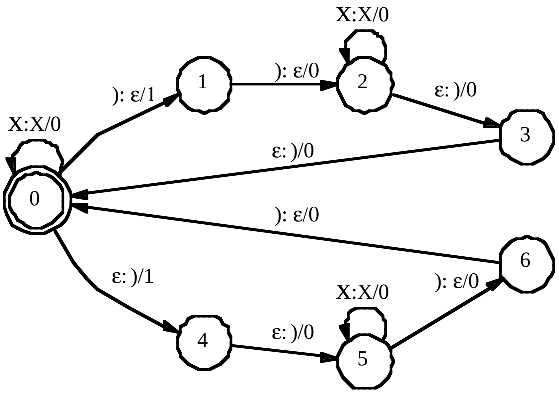


Figure 3: Example transducer giving cost 1 to multiple consecutive bracket insertion and deletion. ‘X’ is intended to range over all symbols.

Luckily, we can find the exact cost that corresponds to this approach, without having to actually perform these transductions, very quickly in the following manner:

1. Find the least cost path p through the composition with our basic labeled bracketing edit transducer
2. Calculate the total cost c of the edits in p
3. Find the length of the longest consecutive string of deleted close brackets m and the length of the longest consecutive string of inserted close brackets n in p
4. r = min(m,n)
5. If $r \geq 2$
   a) remove r consecutive deleted and r consecutive inserted close brackets from p
   b) $c = c - (r - 1)$
   c) return to 3.
5. return c

This will yield the same edit cost that would have resulted from applying the arbitrary transductions described in the previous paragraph. To see this, one must recall that all consecutive close bracket symbols come immediately after terminal items. Because the terminal items cannot be inserted or deleted, this means that the only way to reposition close brackets into the correct location is to delete them and re-insert. The deletion or insertion of any other symbols will not change this. Hence the deletion and insertion of consecutive close brackets will be part of the least cost path through the basic transducer. If there exist consecutive deleted brackets and consecutive inserted brackets of the same length, then there would have been a better path between the input and gold standard parses that involved pairing these consecutive brackets, and hence ac-

Parser	F-measure	Label edit	Span edit	Rule edit
Charniak (2000)	89.74	87.95	86.5	85.6
Collins (2000)	89.71	87.94	86.4	85.5
Collins (1997)	88.24	86.45	84.5	84.0
Roark (2001)	86.71	85.04	82.9	83.3

Table 3: Comparison of four parsers with four measures of accuracy

cruing a single edit. The above algorithm will find all such instances, and adjust the cost accordingly.

We will call the score that results from this method of calculating the edit distance between two labeled bracketings, which counts movements of constituents as a single edit, "Label edit distance" to contrast it with the other edit distance scores that we have discussed, "Rule edit distance", which is based on the edit distance between ordered sequences of rules, and "Span edit distance", which is based on the edit distance between ordered constituents with span and label.

In the next section, we will present scores using different measures for four treebank-style parsers from the literature. This will be followed by a discussion of how these approaches generalize.

4. Evaluation

To evaluate these edit distance measures, we will measure performance of the output from four different parsers. Each of the four parsers was trained on sections 2-21 of the Penn Wall St. Journal Treebank, and tested on section 23. We can thus compare the parsers with each other using different measures, and compare the measures by looking at the distinctions that are made by the measures between the variously performing parsers.

The parsing outputs are those taken from the latest version of the parser presented in Charniak (2000), and the output from Collins (2000), Collins (1997), and Roark (2001a)[4]. For each of the edit distance measures – label, rule, and span – we can define the error rate, which is the number of edits per 100 events (i.e. rules or constituents) in the gold standard parse. The accuracy is then defined as 100 minus the error rate. The accuracy measures that we used were: (i) F-measure; (ii) Label accuracy (based on label edit distance); (iii) Span accuracy (based on span edit distance); and (iv) Rule accuracy (based on rule edit distance). To avoid penalizing the parsers for POS tagging errors, the POS tags in rule instances used for the rule edit distance were replaced with the terminals. Table 3 shows the four different scores of the four parsers.

All three of our edit distance metrics are harder on the parsers than the F-measure score, with rule edit distance giving the lowest scores overall. This is perhaps not surprising given the particular test domain, since the Penn Treebank annotation is known for having many flat constituents, most notably the base NP constituents. In this case, then, we can have structures with no rules correct, but some constituent span predictions correct.

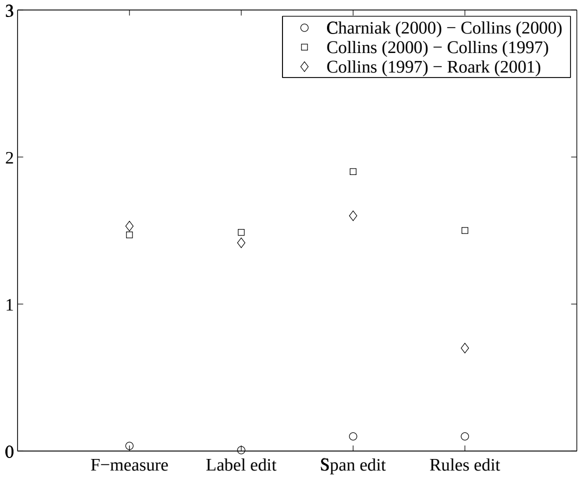


Figure 4: The difference between accuracy scores with F-measure and the three edit distances, for the four parsers: Charniak(2000) minus Collins (2000); Collins (2000) minus Collins (1997); and Collins (1997) minus Roark (2001)

It is interesting to note that, despite the differences in scores – with F-measure giving the highest scores, and rule accuracy generally the lowest scores – there is always the same ordering among parsers. To see how close the distinctions between the parsers are under the four different measures, in figure 4 we plotted the differences in score between parsers ranked n and (n+1), for each of the four measures. As one can see, the distinctions that are being made between parsers are remarkably consistent between the measures. The only exception to this is that the Roark (2001) parser does relatively better in rule accuracy – in fact half a point better than span accuracy, while all the other parsers had worse rule accuracy than span accuracy. This may be due to the fact that the Roark parser is a left-to-right incremental parser, making predictions about children of a left-hand side given the context, in contrast to the others, which also make predictions about heads of constituents given governing heads. Thus it may tend to do a better job choosing children of the left-hand side than it does choosing specific constituents. Alternately, it might be that this parser is making more grave attachment errors than the others, and hence benefitting from the rule-oriented evaluation, which has been shown to be more lenient on these errors. If this were true, however, we would expect the labeled bracketing accuracy to be similarly affected, which it is not.

This is all well-and-good for overall scores, but how do these measures differ on a sentence by sentence basis. Figure 5 plots each sentence from Charniak (2000), with its labeled bracketing error rate versus its E-measure, i.e. 100 minus F-measure. Along the line the scores are identical[5].

[4] The F-measure accuracy for the Charniak parser is 0.2 better than the published result. In addition to these four parsers, we also attempted to obtain the output from one other high accuracy parser, but were unable to do so.

[5] The labeled bracketing error rate goes above 100, because there can be more edits required than there are non-terminals in the parse, e.g. if there is nothing right.

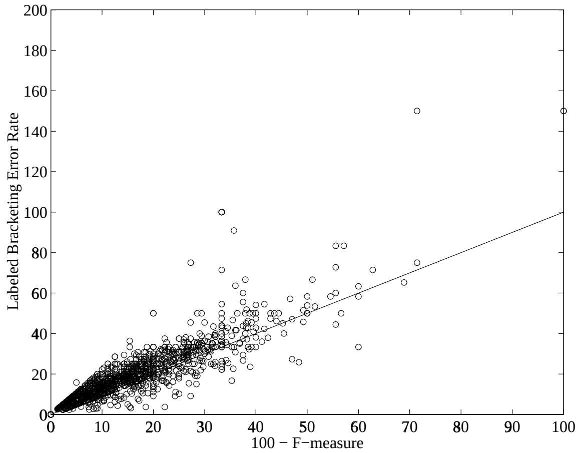


Figure 5: 100 - F-measure (E-measure) versus labeled bracketing edit rate for each sentence in section 23 from Charniak (2000)

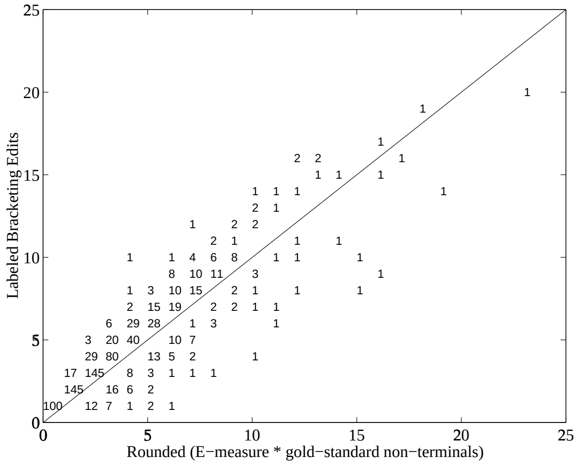


Figure 6: Number of sentences at (x,y), where x is E-measure * non-terminals rounded to the nearest integer, and y is labeled bracketing edits.

Most of the parses are close, but there are some outliers. A brief inspection of some of the outliers provides no surprises. Those out along the x-axis result from constituent attachment decisions that make many constituents incorrect with respect to their span. Those closer to the y-axis result from large differences in precision and recall – i.e. perfect precision and poor recall or vice versa – which tend to give higher F-measure scores than if we simply count the number of edits required to match the gold standard parse.

Most of the parses that have very divergent scores are quite small, so that a small number of required edits accounts for a large percentage of the non-terminals in the parse. These do not account for much when it comes to overall performance, because the raw numbers that they contribute are small. To try to get a better sense of where the bulk of the differences are, we plotted in figure 6 the raw numbers that fall into particular buckets. For the edit distance score, we have the number of edits. To approximate a number of errors in the precision and recall case, we took the E-measure times the number of non-terminals in the gold standard parse, and rounded it to the nearest integer. At each point (x,y), where x is the rounded approximation to errors from the E-measure, and y is the number of edits required for the labeled brackets, we plotted the number of sentences out of the test set of 2416 sentences that fall in that bucket. We omitted those sentences where x = y. As one can see, the bulk of the sentences that differ do so by one or two. Those that are greatly different are relatively rare.

To summarize this section, we have presented three evaluation measures based on edit distance. All three give lower scores than labeled precision and recall to the parsers evaluated, yet they maintain similar distinctions between the parsers. That is, these measures are just as useful as precision and recall in discriminating between parsers in the domain where precision and recall has been used most widely and most successfully. The labeled bracketing and rules edit rate scores give better scores than precision and recall for certain kinds of errors, although these parsers do not seem to be making these errors with much frequency.

In the next section, we will discuss how edit distance scoring allows for very natural generalizations for such things as heterogeneous labeling schemas.

5. Discussion

For the case of labeled bracketing, an edit distance approach provides a very natural way to extend evaluation to include such things as partial matches or equivalence classes. For example, suppose the gold standard parse includes things such as function tags, e.g. not simply NP, but NP-SBJ for subject NPs, or PP-TMP for temporal PPs. One may want to impose some cost for failure to label these tags onto the non-terminal node. This cost may be less than that imposed for a complete mismatch. In order to do this, all that one needs to do is include a transition in our transducer to map from one to the other with the desired cost. Similarly, one may want to include POS tags in the evaluation, but impose less of a cost for mis-tagging something that should be NN as, say, NNP, than one would for mis-tagging it as VBD. Errors in POS tagging as a whole could be assigned less cost than errors in non-POS labels.

Alternately, one might be more interested in whether or not the parser finds the categories of a shallow parse accurately. In this case, one might not impose any cost for deletion, just for insertion, i.e. more structure is okay. Some might consider this “dumbing down” the parser, but in fact getting very flat constituents high in the tree correct can be more difficult than making finer distinctions lower in the tree. For purposes such as information extraction, for example, a lot of hierarchical structure may be less important, which would lead one to an evaluation of this sort. Making deletion cost free is very straightforward.

Perhaps most importantly, edit distance generalizes to any sequence of parser decisions, including those of, say, a dependency parser. Given the left-to-right ordering of the words, we can stipulate an ordering among dependency relations to give an overall ordering of the dependencies. Edit distance can then straightforwardly apply in the same way as it did to the sequence of rules. This approach was taken in evaluating a natural language generation system in Bangalore et al. (2000).

One benefit of using edit distance on a simple sequence of rules, constituents with span, or dependency relationships, is that existing error rate software, such as those used for speech recognition evaluation, can be used as is on the output. This is actually how the evaluation from the previous section was carried out for the rule edit distance and the constituent span edit distance[6]. This is beneficial, not only because one does not have to write new code to evaluate the output, but also because these evaluation routines often can output interesting diagnostic information, including frequent substitutions, merges, or splits.

One comment that is obvious, but deserves to be made nonetheless, is that the evaluation should fit the task. Why should the degree to which a parser can identify dependencies be a better evaluation than the degree to which it can identify constituents? Only insofar as some particular task requires lexical dependencies, not constituents. Parsing, unlike speech recognition, is not generally viewed as a task in and of itself. Rather, it is taken to be in service of something else, e.g. some kind of semantic processing or language modeling. This can also be true of the output of a speech recognizer – the recognized words might be used to attempt to classify the utterance, for example. In that case, it may be useful to measure WER, but ultimately the efficacy of the particular recognizer will be found in the classification accuracy. If an improvement can be had in recognizer accuracy that makes no difference in classification accuracy, then why bother. The same is true in parsing. The danger with embedding a parser in another system for evaluation is that it becomes more difficult to control for the other parts of the system. It thus is beneficial to be able to evaluate the output of the parser independently – but presumably with respect to its ability to recognize the hidden structures that are used by the system. In the absence of such a parser-independent criterion, objective evaluation is difficult. In its presence, error rate and accuracy based on edit distance is a good alternative for evaluation.

In this paper, we have presented three edit distance measures that can be used to evaluate parsers. The intent is not to exhaust the possibilities of the approach, but rather to show that, as a general framework, edit distance can provide the flexibility to meet the varied demands of parser evaluation. These measures can make similar distinctions between parsers as is made by precision and recall, yet repair some long-standing weaknesses of them. Edit distance can be applied to a variety of string representations of trees, including an ordered list of constituents with span, an ordered list of context-free rules, or even the labeled bracketing itself.

6. Acknowledgements

Thanks to Eugene Charniak and Michael Collins for sharing the outputs of their respective parsers. Thanks also to Mehryar Mohri and Srinivas Bangalore for useful discussion on this topic.

7. References

Srinivas Bangalore, Anoop Sarkar, Christine Doran, and Beth Ann Hockey. 1998. Grammar and parser evaluation in the xtag project. In *Workshop on the Evaluation of Parsing Systems*.

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *International Conference on Natural Language Generation*.

Ezra Black, Steven Abney, Dan Flickenger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith 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 *DARPA Speech and Natural Language Workshop*, pages 306–311.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

Michael J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23.

Michael J. Collins. 2000. Discriminative reranking for natural language parsing. In *The Proceedings of the 17th International Conference on Machine Learning*.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing*. Prentice-Hall, New Jersey.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Brian Roark. 2001a. *Robust Probabilistic Predictive Syntactic Processing*. Ph.D. thesis, Brown University. http://arXiv.org/abs/cs/0105019.

Brian Roark. 2001b. Storing automatically generated treebanks in lattices of derivations. In *Proceedings of the IRCS Workshop on Linguistic Databases*, pages 210–218.

Kuo-Chung Tai. 1979. The tree-to-tree correction problem. *Journal of the ACM*, 26(3):422–433.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262.

[6] Since the labeled bracketing edit distance required a special transducer, it was evaluated differently.

Evaluating Syllabification: One Category Shared by Many Grammars

Karin Müller

Department of Computational Linguistics
University of Saarland
Saarbrücken, Germany
kmueller@coli.uni-sb.de

Abstract

We apply a series of context-free grammars to syllabification by using a supervised training method. In our experiments, we investigate various phonological grammars, which strongly differ in structure. A simple evaluation metric "word accuracy" supports grammar development by denoting an increasing performance for grammars enriched with linguistic structure. This evaluation, judging one single category shared by all grammars, is in strong contrast to PARSEVAL, which is designed for a single grammar evaluating (almost) all categories. Using a toy-treebank, we show that the PARSEVAL measures are hard to interpret, since the results are inconsistent with one another. It turns out that evaluating only a limited number of categories (here only one single category) is a harder evaluation measure than measuring the precision of all occurring substructures of a grammar.

1. Introduction

In computational linguistics, the PARSEVAL measures suggested by Black et al. (1991) are now the standard measure for evaluation of context-free grammars (CFGs). The measures quantify precision and recall of common parenthesis based on a treebank. The metrics focus on the precision of all substructures that are specified by a CFG. However, the PARSEVAL metrics are not suitable for all problems that can be described with probabilistic context-free grammars (PCFGs) especially in cases when partial structures are more interesting. In comparison to the field of parsing, there are no phonological treebanks of transcribed words. However, large pronunciation dictionaries are available which can be exploited for evaluation and training. We develop a series of grammars in our supervised experiments and evaluate them on partial structures. There are two main reasons why we chose an evaluation procedure different from PARSEVAL. Firstly, if we had chosen the PARSEVAL evaluation metrics, a separate evaluation suite would have to be constructed for each grammar type. This would be very time-consuming. Secondly, we chose alternative evaluation metrics because syllabification tasks are usually evaluated either by syllable accuracy or even word accuracy, i.e. partial structures of a phonological tree are evaluated. Syllable accuracy means that each syllable is compared with an annotated syllabified corpus. If the system predicts the syllable boundary correctly, syllable accuracy increases. A stricter variant of measuring the capability of a system is to determine word accuracy, which means that each syllable boundary has to be predicted correctly within a word. We try to solve the evaluation problem by annotating and evaluating those structures which are usually referred to in the literature linked to syllabification, and which are shared by all grammars.

The paper is organized as follows: in Section 2, we introduce the syllabification task, as well as a series of phonological grammars describing German syllable structure. Section 3 discusses our evaluation measure in comparison to PARSEVAL. In Section 4, we conclude.

2. Syllabification

In text-to-speech (TTS) systems, like those described in Sproat (1998), the correct pronunciation of unknown and novel words is a crucial problem. Thus, TTS systems usually use large pronunciation dictionaries, however there are in all languages productive word formation processes which generate words that are new to the system. The correct pronunciation of a new word is not only dependent on the correct identification of phonemes but also on the correct assignment of syllables. Van Santen et al. (1997) showed that location in the syllable influences the duration of a phone. Furthermore, identifying syllables boundaries is essential for the application of phonological rules (Kahn (1976), Blevins (1995)), which is certainly the case e.g. for German syllable-final devoicing. Thus, we are interested in developing models that predict syllable boundaries of unknown words as well as possible. This means for context-free grammars, that we need a single category e.g. "SYL" (i) which spans a whole syllable, (ii) occurs in all grammars and (iii) which can be evaluated easily.

In our approach, we developed several context-free grammars and trained them on large automatically transcribed corpora extracted from newspaper corpora by looking up the words and their transcriptions in the pronunciation dictionary CELEX (Baayen et al. (1993)). The different grammars describe the internal structure of words and can be used to predict syllable boundaries after a training procedure. In our experiments, we use a supervised training method which is a combination of treebank and bracketed corpora training (Müller (2001)) exploiting the syllabification information of a pronunciation dictionary and the frequency information of a training corpus consisting of 182 000 words.

We investigate six different grammars to predict syllable boundaries by introducing new categories for each new grammar.

Treebank grammar. The first grammar describes a word as a sequence of syllables consisting of one or n phonemes. The analysis at the top of Figure 1 dis-

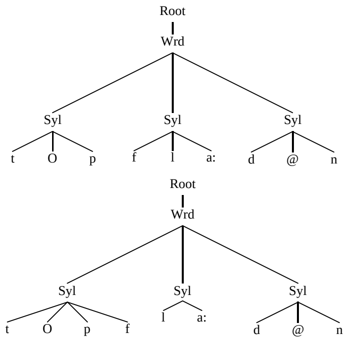


Figure 1: Treebank grammar

plays a possible syllabification of the German word "Topfladen" ([tOpfla:d@n], which can be either translated by *top chapatti* (Top-fladen), or *pot shop* (Topf-laden) (of course there are additional possible syllabifications)

Phoneme grammar. The second grammar introduces an abstract level between the phonemes and the syllables. Each phoneme is labeled by an abstract phoneme label. The grammar learns information about the complexity of a syllable. Figure 2 shows two possible analyses of the phoneme string [tOpfla:d@n] according to the phoneme grammar.

Consonant-vowel grammar. The third grammar distinguishes between consonants and vowels by labeling all phonemes either by a C or a V label. The grammar also demands a vowel inside of a syllable. The structure of the grammar is exemplified by Figure 3 displaying two possible syllabifications of the phoneme string [tOpfla:d@n].

Syllable structure grammar. The fourth grammar specifies syllable structure in more detail. Syllables are split into onset, nucleus and coda. The probability of a consonant depends on its occurrence in the onset or the coda. Two example trees of the phoneme string [tOpfla:d@n] are shown in Figure 4.

Positional syllable structure grammar. The fifth grammar further describes a phoneme depending on the position of the syllable within the word, and depending on the position of the phoneme within the syllable by enumerating the consonants. There are four possible positions of the syllable: word-initial, word-medial, word-final, and monosyllabic words. The consonants of the phonemes are enumerated according to their position inside of the syllable onset, or coda. The structure of the grammar is examplified in Figure 5

Advanced positional syllable structure grammar. An additional feature, cluster size is added to the last grammar. Thus, the consonants depend on their position within the cluster, and the size of the cluster. Figure 6 displays two examples.

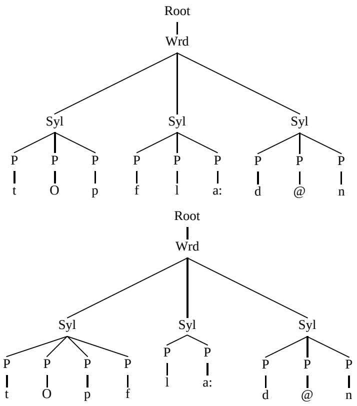


Figure 2: Phoneme grammar

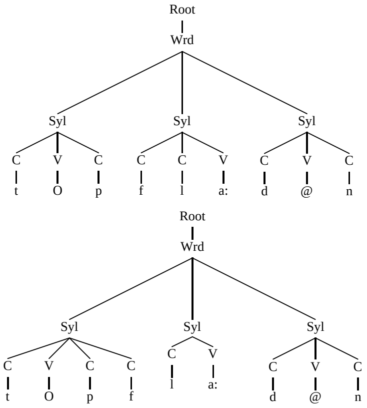


Figure 3: Consonant-vowel grammar

In a next step, the grammars are trained using a novel algorithm consisting of a combination of bracketed corpora and treebank training (see Müller (2001)). However, in contrast to older experiments (where we trained on a series of training corpora ranging from 4 500 to 2.1 million words), we use for our new experiments a fixed training corpus consisting of 182 000 words. Training on this corpus provides a probabilistic version for each of the six phonological grammars. Since all grammars have in common that they are written to predict syllable boundaries, they share the category "SYL" which spans a whole syllable. After training, we can use the most probable parse of a word, the so-called Viterbi parse, to read off the syllables of this word: all phonemes under a syllable node "SYL" belong to one syllable. In Section 3, we describe the performance for the trained grammars on a syllabification task using a huge evaluation corpus of about 240 000 words. Moreover, we try to relate these results to an evaluation using PARSEVAL measures for a toy-treebank.

3. Evaluation

As already presented in Section 2, our system is designed to predict syllable boundaries using probabilistic phonological grammars. For parsing, we used the implementation of Schmid (2000). Our evaluation corpus consists of about 242 000 correctly syllabified words. For syllabification of these words, we used the CELEX dictio-

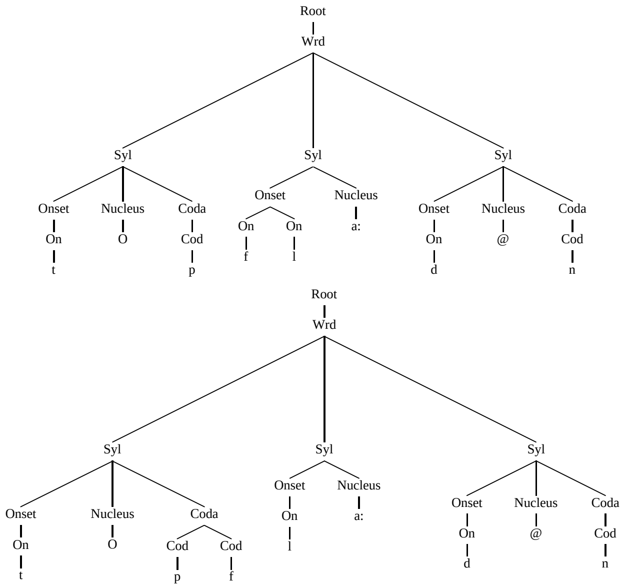


Figure 4: Syllable structure grammar

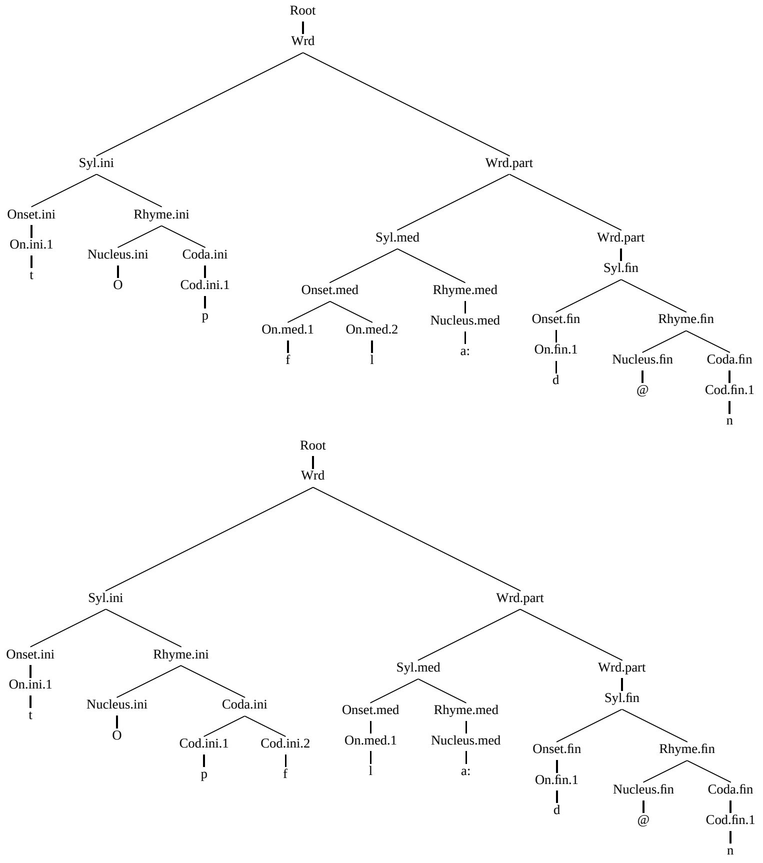


Figure 5: Positional syllable structure grammar

nary. For accuracy measurement, the raw phoneme strings of each word of the evaluation corpus are parsed with our various PCFGs, and the Viterbi parses are taken to extract the syllables of the phoneme strings. Then, the result is compared with the annotated variant in the evaluation corpus. Word accuracy is computed by counting the number of correctly syllabified words, and by dividing this number by the size of the evaluation corpus.

3.1. Evaluation Results

Figure 1 shows our evaluation results. Column 1 displays the series of grammars we investigated, and Column 2 displays the corresponding accuracy values.

The evaluation shows that the grammar with the richest

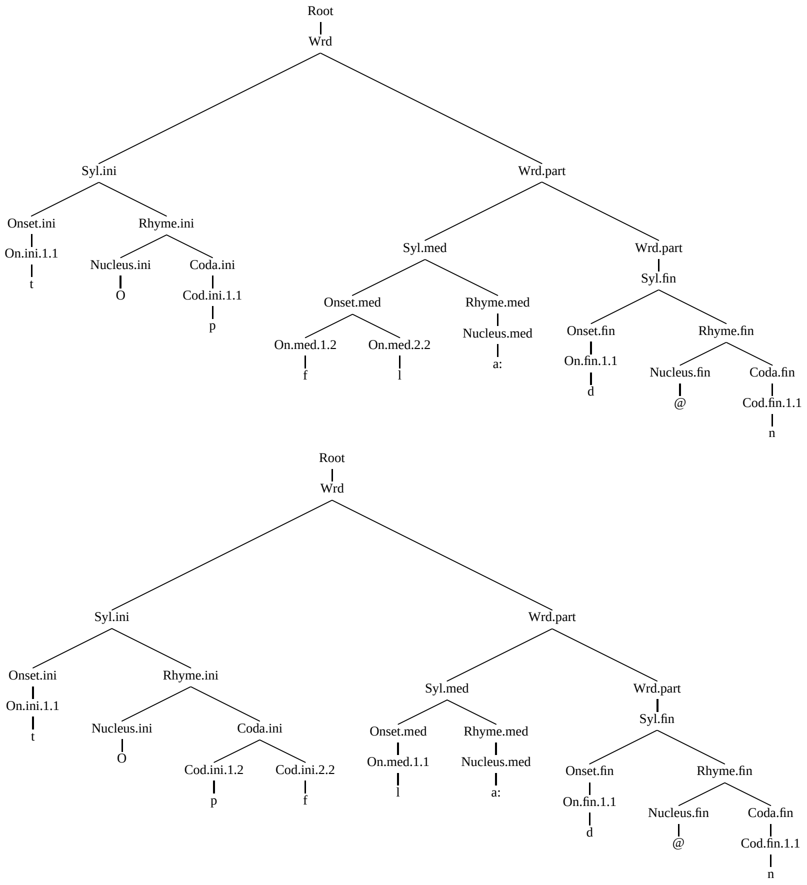


Figure 6: Advanced positional syllable structure grammar

grammar	word accuracy
phoneme grammar	62.37
treebank grammar	71.01
consonant-vowel grammar	93.31
syllable structure grammar	94.12
positional syllable structure grammar	96.42
advanced pos. syllable structure grammar	96.48

Table 1: Word accuracy of the probabilistic phonological grammars trained on a corpus of 182 000 words, and evaluated on a corpus of 242 000 words.

structure, the advanced positional syllable structure grammar, reaches the highest performance of 96.48% word accuracy for a training corpus size of 182 000 words. In general, the more linguistic knowledge is added to the grammar, the higher the accuracy of the grammar is. In contrast to the linguistic grammars, the results of the treebank grammar strongly depend on the size of the training corpus as reported in Müller (2001). They showed that even the simplest grammar, the phoneme grammar, was better than the treebank grammar until the treebank grammar was trained with a corpus size of 77 800. Of course, the low accuracy rates of the treebank grammar (trained on small corpora) were due to the high number of syllables that have not been seen in the training corpus.

3.2. Comparison to PARSEVAL

In this section, we want to exemplify that the problem of using PARSEVAL measures for this series of grammars is that an increase (or decrease) of the PARSEVAL measures can hardly be interpreted in terms of syllabification. In more detail, we show that it is simply unclear what it means for syllabification if two structurally varying grammars yield different values for "labeled precision".

The following example clarifies the problem why we choose an evaluation measure different from PARSEVAL. Let us suppose that

(i) the evaluation corpus consists of one single word, namely the above mentioned example word "Topfladen",

(ii) all six trained grammars predict the (wrong) syllable structure, Top-fladen ([tOp][fla:][d@n]) shown at the top of Figures 1-6,

(iii) the correct syllabification of Topfladen is annotated as Topf-laden ([tOpf][la:][d@n]) coded in six different treebanks shown at the bottom of Figures 1-6,

(iv) we evaluate our series of phonological grammars with the PARSEVAL measure "labeled precision".

Under these assumptions, all grammars fail in solving the syllabification task: all grammars yield a word accuracy of 0%, and a syllable accuracy of 33%. However, if the

grammars	analyses	PARSEVAL without preterminals	labeled precision
treebank grammar (Figure 1)	tree at the top tree at the bottom	**Wrd(0:9)** **Wrd(0:9)**	1/1 = 100%

phoneme grammar
(Figure 2)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl(0:3), Syl(3:6), **Syl(6:9)**
		Wrd(0:9), Syl(0:4), Syl(4:6), **Syl(6:9)**
			2/4 = 50%

consonant-vowel grammar
(Figure 3)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl(0:3), Syl(3:6), **Syl(6:9)**
		Wrd(0:9), Syl(0:4), Syl(4:6), **Syl(6:9)**
			2/4 = 50%

syllable structure grammar (Figure 4)	tree at the top tree at the bottom	**Wrd(0:9)**, Syl(0:3), Syl(3:6), **Syl(6:9)**, **Onset(0:1)**, Coda(2:3), Onset(3:5), **Onset(6:7)**, **Coda(8:9)** **Wrd(0:9)**, Syl(0:4), Syl(4:6), **Syl(6:9)**, **Onset(0:1)**, Coda(2:4), Onset(4:5), **Onset(6:7)**, **Coda(8:9)**,	5/9 = 55.5%

positional syllable structure grammar
(Figure 5)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl.ini(0:3), **Onset.ini(0:1)**, Rhyme.ini(1:3), Coda.ini(2:3), Wrd.part(3:9), Syl.med(3:6), Onset.med(3:5),**Rhyme.med(5:6)**, **Wrd.part(6:9)**, **Syl.fin(6:9)**, **Onset.fin(6:7)**, **Rhyme.fin(7:9)**, **Coda.fin(8:9)**
		Wrd(0:9), Syl.ini(0:4), **Onset.ini(0:1)**, Rhyme.ini(1:4), Coda.ini(2:4), Wrd.part(4:9), Syl.med(4:6), Onset.med(4:5), **Rhyme.med(5:6)**, **Wrd.part(6:9)**, **Syl.fin(6:9)**, **Onset.fin(6:7)**, **Rhyme.fin(7:9)**, **Coda.fin(8:9)**
			8/14 = 57.1%

advanced positional syllable structure grammar
(Figure 6)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl.ini(0:3), **Onset.ini(0:1)**, Rhyme.ini(1:3), Coda.ini(2:3), Wrd.part(3:9), Syl.med(3:6), Onset.med(3:5),**Rhyme.med(5:6)**, **Wrd.part(6:9)**, **Syl.fin(6:9)**, **Onset.fin(6:7)**, **Rhyme.fin(7:9)**, **Coda.fin(8:9)**
		Wrd(0:9), Syl.ini(0:4), **Onset.ini(0:1)**, Rhyme.ini(1:4), Coda.ini(2:4), Wrd.part(4:9), Syl.med(4:6), Onset.med(4:5), **Rhyme.med(5:6)**, **Wrd.part(6:9)**, **Syl.fin(6:9)**, **Onset.fin(6:7)**, **Rhyme.fin(7:9)**, **Coda.fin(8:9)**
			8/14 = 57.1%

Table 2: PARSEVAL measure "labeled precision" (omitting preterminals) calculated on the basis of the examples shown in Figures 1-6

grammars	analyses	PARSEVAL with preterminals	labeled precision

treebank grammar
(Figure 1)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl(0:3), Syl(3:6), **Syl(6:9)**
		Wrd(0:9), Syl(0:4), Syl(4:6), **Syl(6:9)**
			2/4 = 50%

phoneme grammar
(Figure 2)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl(0:3), Syl(3:6), **Syl(6:9)**,**P(0:1)**, **P(1:2)**, **P(2:3)**, **P(3:4)** **P(4:5)**, **P(5:6)**, **P(6:7)**, **P(7:8)**, **P(8:9)**
		Wrd(0:9), Syl(0:4), Syl(4:6), **Syl(6:9) P(0:1)**, **P(1:2)**, **P(2:3)**, **P(3:4)** **P(4:5)**, **P(5:6)**, **P(6:7)**, **P(7:8)**, **P(8:9)**
			11/13 = 84.6%

consonant-vowel grammar
(Figure 3)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl(0:3), Syl(3:6), **Syl(6:9)**,**C(0:1)**, **V(1:2)**, **C(2:3)**, **C(3:4)**, **C(4:5)**, **V(5:6)**, **C(6:7)**, **V(7:8)**, **C(8:9)**
		Wrd(0:9), Syl(0:4), Syl(4:6), **Syl(6:9)**,**C(0:1)**, **V(1:2)**, **C(2:3)**, **C(3:4)** **C(4:5)**, **V(5:6)**, **C(6:7)**, **V(7:8)**, **C(8:9)**
			11/13 = 84.6%

syllable structure grammar
(Figure 4)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl(0:3), Syl(3:6), **Syl(6:9)**, **Onset(0:1)**, **On(0:1)**, **Nucleus(1:2)** Coda(2:3), **Cod(2:3)**, Onset(3:5), On(3:4), **On(4:5)**, **Nucleus(5:6)**, **Onset(6:7)** **On(6:7)**, **Nucleus(7:8)**, **Coda(8:9)**, **Cod(8:9)**
		Wrd(0:9), Syl(0:4), Syl(4:6), **Syl(6:9)**, **Onset(0:1)**, **On(0:1)**, **Nucleus(1:2)** Coda(2:4), **Cod(2:3)**, Cod(3:4), Onset(4:5), **On(4:5)**, **Nucleus(5:6)**, **Onset(6:7)** **On(6:7)**, **Nucleus(7:8)**, **Coda(8:9)**, **Cod(8:9)**
			13/18 = 72.2%

positional syllable structure grammar
(Figure 5)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl.ini(0:3), **Onset.ini(0:1)**, **On.ini.1(0:1)**, Rhyme.ini(1:3), **Nucleus.ini(1:2)** Coda.ini(2:3), **Cod.ini.1(2:3)**, Wrd.part(3:9), Syl.med(3:6), Onset.med(3:5) On.med.1(3:4), On.med.2(4:5), **Rhyme.med(5:6)**, **Nucleus.med(5:6)**, **Wrd.part(6:9)** **Syl.fin(6:9)**, **Onset.fin(6:7)**, **On.fin.1(6:7)**, **Rhyme.fin(7:9)**, **Nucleus.fin(7:8)** **Coda.fin(8:9) Cod.fin.1(8:9)**
		Wrd(0:9), Syl.ini(0:4), **Onset.ini(0:1)**, **On.ini.1(0:1)**, Rhyme.ini(1:4), **Nucleus.ini(1:2)** Coda.ini(2:4), **Cod.ini.1(2:3)**, Cod.ini.2(3:4), Wrd.part(4:9), Syl.med(4:6), Onset.med(4:5) On.med.1(4:5), **Rhyme.med(5:6)**, **Nucleus.med(5:6)**, **Wrd.part(6:9)**, **Syl.fin(6:9)** **Onset.fin(6:7)**, **On.fin.1(6:7)**, **Rhyme.fin(7:9)**, **Nucleus.fin(7:8)** **Coda.fin(8:9) Cod.fin.1(8:9)**
			15/23 = 65.2%

advanced positional syllable structure grammar
(Figure 6)
	tree at the top
	tree at the bottom
		Wrd(0:9), Syl.ini(0:3), **Onset.ini(0:1)**, **On.ini.1.1(0:1)**, Rhyme.ini(1:3) **Nucleus.ini(1:2)**, Coda.ini(2:3), Cod.ini.1.1(2:3), Wrd.part(3:9), Syl.med(3:6), Onset.med(3:5), On.med.1.2(3:4), On.med.2.2(4:5), **Rhyme.med(5:6)**, **Nucleus.med(5:6)** **Wrd.part(6:9)**, **Syl.fin(6:9)**, **Onset.fin(6:7)**, **On.fin.1.1(6:7)**, **Rhyme.fin(7:9)** **Nucleus.fin(7:8)**,**Coda.fin(8:9) Cod.fin.1.1(8:9)**
		Wrd(0:9), Syl.ini(0:4), **Onset.ini(0:1)**, **On.ini.1.1(0:1)**, Rhyme.ini(1:4), **Nucleus.ini(1:2)**, Coda.ini(2:4), Cod.ini.1.2(2:3), Cod.ini.2.2(3:4), Wrd.part(4:9) Syl.med(4:6), Onset.med(4:5), On.med.1.1(4:5), **Rhyme.med(5:6)**, **Nucleus.med(5:6)**, **Wrd.part(6:9)**, **Syl.fin(6:9)**, **Onset.fin(6:7)**, **On.fin.1.1(6:7)**, **Rhyme.fin(7:9)**, **Nucleus.fin(7:8)**, **Coda.fin(8:9) Cod.fin.1.1(8:9)**
			14/23 = 60.9%

Table 3: PARSEVAL measure "labeled precision" (including preterminals) calculated on the basis of the examples shown in Figures 1-6

PARSEVAL measure "labeled precision" is expected to be useful for the syllabification task, then "labeled precision" should express that all grammars perform equally good (or bad) in our toy-setting.

Table 2 displays the results of "labeled precision". The matching brackets are shown in bold. Following the suggestion of Manning and Schütze (1999), the root node "Root" is not taken into account. Moreover, we omitted comparisons of pre-terminal nodes, since Manning and Schütze (1999) suggest to evaluate tagging and parsing separateley. In this evaluation, the simplest grammar, the treebank grammar, achieves the highest value for labeled precision (100%), since only the word-node is taken into account. The phoneme and the consonant-vowel grammar achieve the lowest values for labeled precision (50%). Table 3 also displays the results of "labeled precision", but here, we include the comparison of pre-terminals, due to the fact that we never applied our grammars using a seperate tagger. Here, the treebank grammar achieves the lowest value for labeled precision (50%). The phoneme grammar, and the consonant-vowel grammar achieve the highest values for labeled precision (84.6%).

Thus, the results of both evaluations are hard to interpret, since they are inconsistent with one another. Furthermore, neither the first evaluation (omitting pre-terminals), nor the second evaluation (including pre-terminals) correspond to syllable accuracy, or word accuracy.

For these reasons, we doubt that the PARSEVAL measures are useful for evaluation of phonological grammars, at least for our grammars, which we developed for the syllabification task in mind. In contrast, we focus on evaluation of partial structures, namely on the category "SYL", and measure how good the grammars detect this single category on the word level. Interestingly, it seems that evaluating only a limited number of categories (here only a single category) is a harder evaluation measure than measuring the precision of all occurring substructures of a grammar.

3.3. Grammar Transformation: An Attempt to Map Word Accuracy to PARSEVAL

In this section, we discuss a grammar transformation enabling the measurement of word accuracy via PARSEVAL measures. In more detail, it could be suggested that the output of the phonological parser can be transformed to a tree, where all categories are removed except for the categories "Root", "SYL", and the terminals. However, if we follow this suggestion, there appear some problems. For the transformed grammar,

(i) the remaining category "SYL" is a pre-terminal node, which is usually NOT evaluated according to PARSEVAL; at least, if we follow Manning and Schütze (1999), who suggest to treat the tagging and parsing problem separateley.

(ii) all phonological information about syllable structure is lost, i.e., the syllabification problem is transformed to a tagging problem. However, we proved in recent work (Müller (2001), Müller (2002)) that it is advantageous to regard syllabification as a parsing problem.

(iii) although syllabification is a kind of segmentation, i.e., a one-dimensional process on a sequence of phonemes, we experienced, the more linguistic knowledge is added to the grammar, the higher the word accuracy of the grammar is. Thus, in our point of view, it is more adequate to model syllabification as a higher-dimensional process.

For these reasons, we prefer to use phonological enriched context-free grammars for stochastic inference and an evaluation focusing on partial structures most important for the particular task.

4. Conclusion

We presented an approach to supervised learning and automatic detection of syllable boundaries. In our experiments, we used a variety of grammars, which strongly differ in structure.

An evaluation using the standard measure for syllabification "word accuracy" shows that the grammar with the richest structures, the advanced positional syllable structure grammar, reaches the highest performance of 96.48% word accuracy for a training corpus size of 182 000 words. In general, the more linguistic knowledge is added to the grammar, the higher the word accuracy of the grammar is.

This evaluation, judging one single category of many grammars, is in strong contrast to PARSEVAL, which is designed for a single grammar evaluating (almost) all categories.

In a second evaluation using the original PARSEVAL measures on a toy-treebank, and a simple variant of the PARSEVAL measures, the results of both evaluations are hard to interpret, since they are inconsistent with one another. Furthermore, we found that neither the first evaluation (omitting pre-terminals), nor the second evaluation (including pre-terminals) correspond to syllable accuracy, or word accuracy.

Moreover, it turns out that evaluating only a limited number of categories (here only a single category) is a harder evaluation measure than measuring the precision of all occurring substructures of a grammar.

Lastly, we discussed a grammar transformation enabling the measurement of word accuracy via PARSEVAL measures. Here, it was necessary to reduce the syllabification problem to a tagging problem. However, we believe that it is advantageous to regard syllabification as a parsing problem.

For these reasons, future work will still use phonological enriched context-free grammars for stochastic inference and evaluations focusing on partial structures most important for the particular phonological task.

REFERENCES

Harald R. Baayen, Richard Piepenbrock, and H. van Rijn. 1993. The CELEX lexical database—Dutch, English, German. (Release 1)[CD-ROM]. Philadelphia, PA: Linguistic Data Consortium, Univ. Pennsylvania.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A Procedure for Qualitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of Fourth DARPA Speech and Natural Language Workshop*.

Juliette Blevins. 1995. The Syllable in Phonological Theory. In John A. Goldsmith, editor, *Handbook of Phonological Theory*, pages 206–244, Blackwell, Cambridge MA.

Daniel Kahn. 1976. *Syllable-based Generalizations in English Phonology*. Ph.D. thesis, Massachusetts Institute of Technology, MIT.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Karin Müller. 2001. Automatic Detection of Syllable Boundaries Combining the Advantages of Treebank and Bracketed Corpora Training. In *Proc. 39th Annual Meeting of the ACL*, Toulouse, France.

Karin Müller. 2002. Probabilistic Context-Free Grammars for Phonology. Submitted.

Helmut Schmid. 2000. LoPar. Design and Implementation. [http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/LoPar-en.html].

Richard Sproat, editor. 1998. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Kluwer Academic, Dordrecht.

Jan P.H. Van Santen, Chilin Shih, Bernd Möbius, Evelyne Tzoukermann, and Michael Tanenblatt. 1997. Multilingual duration modeling. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, volume 5, pages 2651–2654, Rhodos, Greece.

Towards Comparing Parsers from Different Linguistic Frameworks An Information Theoretic Approach

Gabriele Musillo and Khalil Sima'an

Language and Inference Technology Group (LIT)
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands
musillo@science.uva.nl; khalil.simaan@hum.uva.nl

Abstract

Various efforts have been undertaken for developing methods for parser evaluation (Black et al., 1991; Lin, 1995; Carroll et al., 1996; Lin, 1998; Carroll et al., 1998; Carroll et al., 1999). These efforts concentrated on developing measures of parser performance, e.g. PARSEVAL (Black et al., 1991) labeled recall/precision for phrase-structure annotations. Different problems have been identified in the existing evaluation methods, but one of these problems strikes us as particularly challenging. The current benchmark for parser evaluation, Penn Wall Street Journal (WSJ) tree-bank (Marcus et al., 1993), cannot be used for the evaluation of parsers that are based on linguistic theories or annotation schemes that differ (*essentially*) from the annotation scheme found in this tree-bank. This problem can be restated as follows: how can parsers from different linguistic frameworks be *compared* in a quantitative and thorough manner? In this paper, we address this problem and suggest a new methodology for comparing parsers. The new methodology integrates Information Theoretic measures together with the PARSEVAL measures in a way that allows direct comparison of parsers that originate from different linguistic frameworks.

1. Introduction

In the last decade or so, a relatively large body of work in Computational Linguistics has been directed at the development and application of different parsing models for natural language processing e.g. (Brill, 1993; Magerman, 1993; Bod, 1995; Charniak, 1996; Eisner, 1996; Ratnaparkhi, 1997; Collins, 1997; Carroll et al., 1998; Lin, 1998; Chiang, 2000; Sima'an, 2000). Much work has been concentrating on how to extract stochastic models from existing tree-banks. One tree-bank in particular, the Penn Wall Street Journal tree-bank (Marcus et al., 1993) has received much attention in these efforts. Regardless of the reasons for this situation, this tree-bank has become a kind of bench mark for the evaluation and comparison of parsers. However, parsers that do not abide by the same linguistic framework as the WSJ tree-bank, or parsers for other languages than American English, are hard to compare to the mainstream which has been tested on the WSJ tree-bank. In this paper we address the issue of parser comparison across different linguistic frameworks. This problem is interesting as it touches on linguistic issues, but also on the question "how to view the role of language structure". In this preliminary report on our research into this question, we assume that the parsers that are being compared are built for the same language. We also assume the existence of a corpus of utterances, i.e. that parser comparison takes place on specific domains of language use. Furthermore, in our evaluation here we do not take into consideration the important aspect of parser efficiency.

The structure of this paper is as follows. Section 2 discusses the linguistic aspects of how to compare parsers that originate from different linguistic frameworks. Section 3 argues that parser comparison needs more thought than parser evaluation, exactly because the scores need to be compared somehow. Section 4 we extend PARSEVAL with an Information Theoretic methodology, which allows parser comparison across different linguistic frameworks. Finally, section 5 concludes this paper.

2. How to compare parsers?

Remarkably, current parser evaluation practice seems to have been limited to a single benchmark tree-bank (Penn WSJ tree-bank). The evaluation of newly developed parsers proceeds by testing on a held-out portion of this tree-bank. When the parsers are acquired from the tree-bank itself, evaluation (and comparison) of the parsers is based on the PARSEVAL measures (Black et al., 1991). However, when a new parser is devised and this parser employs a different linguistic framework than the annotation scheme of the Penn WSJ tree-bank (e.g. dependency grammar), a serious problem arises. The problem lies in how to relate the different syntactic schemes to one another. There seem to be two related ideas on how to address this problem: (1) devising a general, syntactic scheme, a kind of "common ground", which serves as an "interlingua", or (2) devising mappings between each pair of syntactic schemes. In this section we argue that both suggestions seem not be workable in practice. We review the more popular among the two, i.e. the methods of devising mappings from the WSJ annotation scheme to other schemes. Subsequently we suggest that it is more expedient to develop different tree-banks of the same corpus of utterances, each in a different syntactic annotation scheme.

2.1. Common syntactic scheme?

At first glance, the problem of specifying the syntactic "common ground" seems to rely on the choice of some general linguistic framework to which both parsers' outputs can be mapped in order to compare them. However, selecting a common linguistic framework seems a hopeless task:

the problem lies in anticipating the kinds of linguistic information that a new linguistic theory might be interested in. Take for example the *Minipar* parser (Lin, 1998) or the *Link parser* (Sleator D, 1991), the first outputs one type of dependencies, while the second outputs so called “links”. Both parsers do not exactly coincide with the traditional grammatical relations used in other frameworks (or with the WSJ annotation). How this kind of syntactic information would be anticipated by a common framework is not completely clear. We believe that a common framework will always be contested as being more favorable to some parser than another. Moreover, it seems to us that the goal of devising such a framework coincides with the ultimate goal, which has evaded the syntactic linguistic work for so many years now. This might be inherent to deciding on the so called “borders” of syntax, which seems to overlap with morphology on the one side, and with semantics and pragmatics on the other. It is highly doubtful that such a framework can be developed.

2.2. Mappings between syntactic schemes

Various researchers (Lin, 1995; Carroll et al., 1998) have developed methods that attempt at transforming the the Penn WSJ format into the different output formats of their parsers. However, there is in these efforts a hidden assumption: a complete mapping can be constructed, which maps a WSJ parse-tree into a parse-tree in any of these schemes. Apart from the linguistic arguments that exist against the “relatively shallow” WSJ style of annotation, there is a serious problem in assuming the existence of a complete, possibly deterministic mapping.

The arguments against this practice emerge from linguistic frameworks (e.g. dependency grammar) that differ to a large extent from the framework employed in the WSJ tree-bank. The main arguments address the risks that accompany mapping parse-trees from one framework to another. When a parse-tree is mapped from one framework to another, one might

- risk losing linguistic information (e.g. some dependencies not found in the WSJ),
- face ambiguity, since the categories provided by the parser might map onto different WSJ categories (possibly dependent on context).

These problems suggest that the comparison of two parsers can not rely on mapping the output onto some pre-selected linguistic framework, since different linguistic frameworks address different syntactic aspects. Nevertheless, we find in the literature various empirical efforts aimed at devising such mappings, notably (Lin, 1995; Carroll et al., 1998). Next we first address what it takes to devise a mapping from Phrase-Structure to Dependency grammar and vice versa. Then we shortly discuss some of the problems that exist in the mappings devised by (Lin, 1995; Carroll et al., 1998).

2.3. What is necessary for a mapping?

We concentrate on two popular linguistic frameworks of syntax: dependency and phrase-structure. Although this could be an illusive task, we give here a simplified description of the two, each in a single line. Phrase-structure syntax allows describing the syntactic structure of utterances in terms of the phrases which constitute them, using a hierarchical and recursive set of concepts. In contrast, dependency syntax aims at making explicit the dependencies between the pairs of words in the utterance. Both approaches aim at facilitating the discovery of argument-structure, which is often assumed by subsequent semantic processing.

It has been suggested by (Hudson, 1984; Covington, 1992; Covington, 1994) that, according to a phrase-structure grammar, constituency is basic and dependency (or government) is derived, whereas according to dependency grammar, dependency is basic and constituency is derived. If this claim is correct, then a transformation procedure and its inverse that map phrase-structure parses to dependency parses could be defined; in that case, phrase-structure and dependency grammars are to a large extent isomorphic.

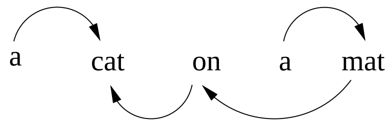


Figure 1: A dependency parse

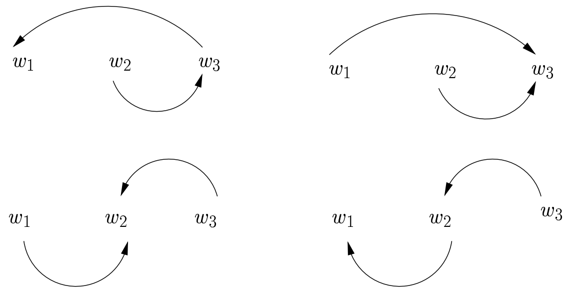


Figure 3: Different possible dependency trees

The problem lies, of course, in the kind of concepts that each framework presupposes: the types of dependencies and the constituent types must refer to the same abstract syntactic concepts, which is usually not the case. In order to shed some light on the problematic aspects of such a mapping, consider the expression *a cat on a mat* and the plausible dependency parse for it shown in figure 1; at least three distinct phrase-structure parses may be projected from it as shown in figure 2. Clearly, there is here a problem of how to decide on the single correct phrase-structure parse, given the dependency structure. The reverse mapping can also be problematic: consider the following unlabeled bracketing $(w1(w2w3))$ of an expression $w1w2w3$. At least four dependency parses can be generated from it as shown in figure 3. Again, a principled choice of the single correct dependency parse is not easy and demands a procedure for recognizing the head word of each phrasal category. The problem, however, in devising head word

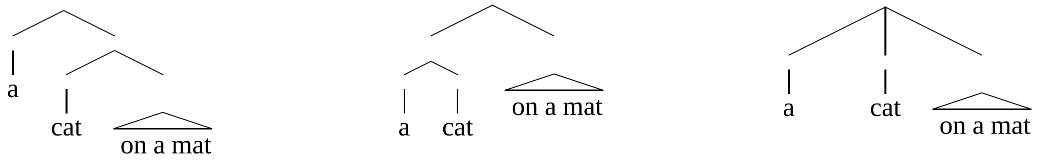


Figure 2: Three different phrase-structure trees for same dependency structure

recognition procedures for an existing tree-bank, has been exemplified by the various versions of the head recognition procedure developed by for the WSJ tree-bank (Magerman, 1993; Collins, 1997; Buchholz et al., 1999; Eisner, 2001). In any case, it seems that the problems that accompany these mappings can be summarized in two elements (1) a common set of concepts that underlie the types in each of the two frameworks, and (2) a clear and well founded definition of a head recognition procedure. Let us consider two attempts at developing such mappings.

2.4. Lin's proposal

In (Lin, 1995), a dependency parse of a sentence is defined as a set of tuples. Such dependency tuples consist of 5 components: a *dependent*, a *PoS*, a *position*, a *head* and a *type*. This last component is optional. Lin defines the values that can be assigned to these components as follows: a word in a sentence to be parsed is assigned to the *dependent* variable, *PoS* represents its lexical category, the head word on which the value of *dependent* depends is assigned to the *head* variable, the *position* takes a value in the set $\{<, >, <<, >>, <<<, ..., *, ?\}$. Remarkably, no well-defined set of values is defined for the optional component *type*. Furthermore, Lin gives no hints at how to label *head-dependent* relations.

Lin presents an algorithm to transform a constituency tree into a dependency tree. His transformation procedure exploits suggestions made in (Magerman, 1993) for determining *lexical representatives* of phrases. Whether the notion of *lexical representative* coincides with the notion of *head* as used in dependency grammar, is not clear. Suppose, for the sake of the argument, that *lexical representatives* are *heads* and consider the *wh*-interrogative, parsed according to the bracketing guidelines for the Penn Tree Bank as shown at the left side of figure 4. Let us apply Magerman's rules to it. According to these rules, the head-word of *SBARQ* is the head-word of *SQ*, that is *propose*. Therefore, the head-word of *which measures* is a dependent of *propose*. However, consider the *wh*-interrogative on the right hand side of figure 4. The head-word of *SBARQ* is *think*. Therefore, the *WHNP which measures* is not any more a dependent of *proposed*, it is a dependent of the head word *think*. This implies that transforming standard phrase structure analysis into some dependency representation in this way results in loss of information. Such information represented by the position of a trace is of course relevant to semantic interpretation. Our examples clearly show that such a transformation procedure fails to detect a dependency that relates a dependent to a "lower head" (one that does not percolate across the constituent boundaries).

2.5. Carroll et. al's proposal

Carroll et al. (Carroll et al., 1998) has proposed some relational evaluation measures that exhibit some resemblance to Lin's. They describe a corpus annotation scheme that encodes grammatical relations between heads and dependents. We believe that Carroll's proposal is somehow superior to Lin's in a few aspects. Firstly, the set of dependency types or grammatical relations is well-defined and constitutes a hierarchy. This allows robust and shallow evaluation. Secondly, grammatical relations are strictly speaking not dependency relations; the external argument of a "subject control verb" is grammatically related to the control verb and to the controlled verb (e.g. *I promise to come*, where *I* is related to both *promise* and *come*). Finally, grammatical relations are specified even for moved phrases that do not occur in a canonical position. This addresses a problem in Lin's proposal, mentioned above.

From the experience described in (Carroll et al., 1998), it might seem that dependency types (or grammatical relations) are easy to specify and extract from phrase-structure. Nevertheless, this is true only because (Carroll et al., 1998) assumes that the phrase-structure grammar is an explicit, determinate set of rules. As Carroll et al. recognize, extracting grammatical relations from an implicit grammar, induced automatically from a tree-bank, is much harder to do consistently. In addition, the grammatical relations in (Carroll et al., 1998) do not capture some relevant information. For instance, topicalized constituents of the Penn Tree Bank (bearing the TPC tag) are ignored, because they are allegedly difficult to specify under which conditions a constituent is topicalized.

Clearly, from these examples we observe that developing deterministic, complete mappings between phrase-structure and dependency grammars is a tedious and risky task. For a nice review of the problems that arise in relating Dependency to Phrase-Structure syntax see (Schneider, 1998). We believe that the development of different tree-banks, each in another linguistic annotation scheme, for the same corpus of utterances might provide a more fruitful path to proceed. When a pair of parallel tree-banks exits, it is possible to explore automatic means for learning complex, stochastic mappings between the two. More importantly, a pair of parallel tree-banks for the same corpus of utterances may serve as a suitable infrastructure for the comparison of parsers from different linguistic frameworks as we describe in the rest of this paper.

3. Comparison: more than evaluation

In line with current practice, we believe that empirical parser evaluation requires a manually constructed, gold-

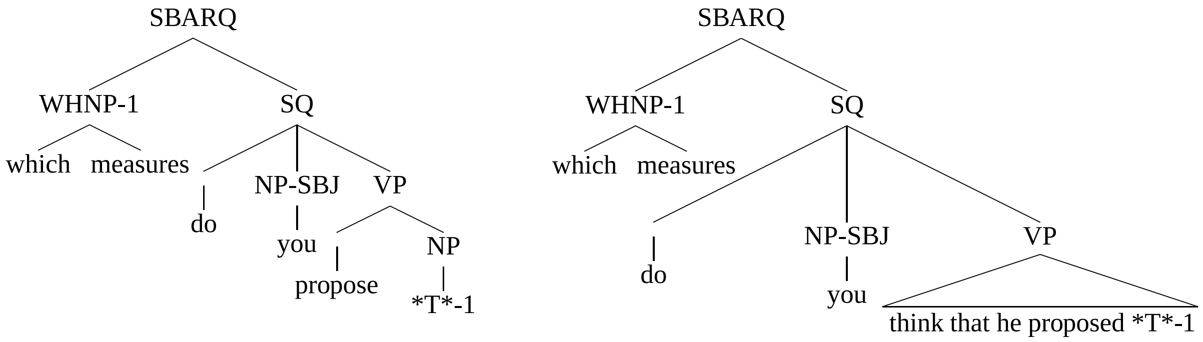


Figure 4: Two Penn WSJ style parse-trees

standard tree-bank and suitable measures of the "similarity" between the analyses output by the parser and the corresponding analyses that are found in the tree-bank. Usually, the measure of "analyses similarity" consists of different figures pertaining to coverage (or recall) and accuracy (or precision). In contrast, the task of *comparing* parsers can be more complex than the evaluation of a single parser (or the comparison of parsers that share the same output scheme). When two parsers are being compared, another major issue, beside evaluation, must be addressed: how to compare the similarity measures across different kinds of parser outputs (possibly originating from different linguistic frameworks)? We believe that the latter question is of theoretical importance and we address it in this section. First, however, we need to discuss the multiple possibilities for parser comparison and provide the argumentation that underlies the specific choices that we make.

3.1. Task-oriented comparisons

Initially, we distinguish between two goals of parser comparison: (1) the suitability of the parser to a given task, and (2) the suitability of the parser as a model of syntactic language processing. Although (ideally) the two goals are strongly related, in practice they might imply different comparison methodologies. The comparison of parser's suitability for a specific task is usually guided by some detailed specification of the requirements which the parser must meet. For example, the simplified Question-Answering task requires the parser to output (as fast as possible) the main predicate-argument structure of the input. In contrast, more complex tasks, such as the task of Machine Translation, will possibly require a much more detailed syntactic analysis of the input. Task-oriented comparison is interesting and useful, but is strongly specific to the task at hand, which means that it does not constitute a general comparison methodology.

3.2. Qualitative comparisons

When the parser comparison is not tied to a specific task[1], parser comparison is aimed at investigating the utility of the different models underlying the parsers. Clearly, a *qualitative* comparison, based on theoretical considerations of coverage of language phenomena (e.g. (Carroll and Weir, 1997)), could be illuminating. Issues such as "what phenomena the parser can be expected to cover" and "what quality of the output is provided by the parser" are important for advancing the state of the art. However, qualitative comparisons become more powerful when they are supplemented with quantitative comparisons that are based on actual empirical evidence (weighted according to expected frequency of occurrence). This is because the theoretical investigations might pay too much attention to relatively infrequent phenomena and less attention to frequent (yet seemingly irrelevant) phenomena. Empirical, quantitative parser comparison aims at providing an answer to the question: what quality of output is provided by the parser and how does it compare to other parsers? Before we address this question, however, we address a related idea which is currently being floated as an alternative for (full) parser comparison: partial evaluation.

3.3. Partial evaluation based comparisons

Because of the problematic mappings between the different linguistic frameworks, it seems suitable to consider only some of the issues upon which these frameworks agree. For example, one could conduct comparisons on the main predicate-argument structure of the input, or on recall/precision for the set of predicate-argument structures for the verbs in the input utterance. Similar suggestions have been made in order to contrast the so-called "shallow" parsers to existing "full" parsers, by e.g. listing the recall/precision on each phrasal category (or kind of dependency) separately (Tjong Kim Sang and Déjean, 2001). These suggestions for *partial evaluation* usually provide a detailed and informative listing of various aspects of the parser's behavior. We think that these should be taken more seriously in *practical parser evaluation*. It is important to have detailed lists of scores of a parser on different tasks. However, partial evaluation has its limitations, even for practical comparisons. It is very hard to predict what elements in the output of a given parser could be important. For example, predicate-argument structures, which take only verbal predicates into consideration, are useless for some applications where prepositional phrases are important (e.g. domains of travel information, or money transactions etc). Another weakness of partial evaluation is that, by definition, it does not answer the need for a "bottom line figure" which summarizes the behavior of the parser, and allows direct comparison to other parsers.

There is, moreover, a more urgent matter, which par-

[1] This is currently the case in parser evaluations on the Wall Street Journal corpus, for example.

tial evaluation does not address, and which is of theoretical importance. This concerns the question: how much information[2] does the parser return, and what is it's quality? Answering this theoretical question is important for advancing the state of the art in natural language processing. In the light of the current divergence in parser output formats (e.g. shallow vs. deep parsers) and given the differences between the linguistic frameworks, it becomes important to be able to measure differences in the informativeness of parsers, even when their outputs are not directly comparable.

3.4. Comparison on parallel tree-banks

As argued in Section 2., for comparing parsers that come from different linguistic frameworks (or different depths of analysis) it is necessary to maintain some kind of mapping between the outputs of the parsers. The mapping might be realized in one of two manners:

Explicit: a tree-bank exists in one annotation scheme (according to some linguistic framework) together with a sound, complete and correct mapping which translates every analysis in the tree-bank into the corresponding analysis in the other linguistic framework,

Implicit: two parallel tree-banks[3] of the same corpus of utterances, each annotated according to one of the linguistic frameworks.

We already showed in Section 2., that developing an explicit deterministic mapping seems a hard task. Therefore, we advocate the use of implicit mappings that are embodied in parallel tree-banks of the same corpus of utterances. It is evident that given such pairs of tree-banks, different automatic methods can be explored for learning complex, stochastic mappings between the two tree-banks. How to acquire these mappings is an interesting subject of research but is beyond the scope of this paper.

4. An Information Theoretic proposal for parser comparison

Suppose we are given two parsers P_1 and P_2 which have different output schemes, respectively, L_1 and L_2. Suppose also we are given a corpus of utterances C, and tree-banks TB_1 and TB_2 that are annotated versions of C according to schemes L_1 and L_2, respectively. In order to ground the discussion, the reader might want to imagine that L_1 is dependency grammar (Mel'$\check{c}$uk, 1988) and L_2 is phrase structure grammar (Manning and Schutze, 1999); or alternatively, L_1 could be the scheme output by a shallow parser and L_2 a "deeper" linguistic scheme. The question is, how do we compare P_1 and P_2 in a way that takes into consideration not only the coverage/accuracy but also the informativeness of their output? Below we discuss the two issues of coverage/accuracy and informativeness separately. Subsequently we propose combined measures which allow comparison.

4.1. Coverage/accuracy: generalizing PARSEVAL

The PARSEVAL measures of labeled constituent recall and precision (Black et al., 1991) have been central in current efforts at parser evaluation on the current American English language benchmark (Penn WSJ tree-bank). It has often been claimed that these measures are not suitable for evaluating e.g. dependency syntax. Indeed, when taken literally, constituency can be meaningless when evaluating dependency syntax. However, the PARSEVAL measures can easily be generalized to deal with whatever kind of parses as long as they can be represented as sets of relations. For example, a labeled constituent $\langle i, j, XP\rangle$ is a relation (where i and j are the positions of the left-most and right-most words respectively and XP is the label of the constituent); a labeled dependency $\langle h, d, L\rangle$ is a relation (where h and d are the positions of the head-word and the dependent, and L is the label of the dependency). A parse-tree, whether in dependency syntax or in phrase-structure, can be represented as a set of such relations (cf. (Goodman, 1998)). Recall and precision, as a direct generalization of the notation used in PARSEVAL, are defined as measures over sets of such relations. If a given parser outputs parse T for sentence U for which the gold standard parse is G, Goodman defines[4]:

$$Recall(T,G) = \frac{|G \cap T|}{|G|} \qquad Precision(T,G) = \frac{|G \cap T|}{|T|}$$

We believe that the PARSEVAL measures can be generalized further to stricter recall/precision measures, where a parse-tree is viewed as a set of relations that range over different aspects of syntax, e.g. relations in which the labeled constituent is head-lexicalized, and possibly supplemented with the set of subcategorization frames of its head word. The notions of recall/precision over sets of relations are general enough to accommodate a wide range of aspects of parse-trees, including suggestions for partial evaluation, e.g. on the basis of predicate-argument structures of verbs.

Hence, the parsers P_1 and P_2, assumed earlier, can be evaluated on their own tree-banks TB_1 and TB_2 using the PARSEVAL recall/precision measures. However, the PARSEVEAL measures of recall/precision do not address the problem of comparison across different output/annotation schemes. To arrive at a suitable comparison methodology we first need to define measures of the informativeness of the output of a given parser.

4.2. Informativeness of a parser

What makes a parser informative? To answer this we turn to the Information Theoretic concept of compression. Suppose we are given two parsers. The one parser outputs only unlabeled bracketed parse-trees, while the other labels the same parse-trees with different syntactic categories. The output of the second parser can be described as more informative. As it turns out, the kind of concepts, e.g. phrasal categories or dependency types, which the parser includes in its output determine its informativeness. For example, a parser that marks the difference between singular/plural noun and verb phrases could be more informative than another that does not do so (all else being equal, of

[2] This is opposed to the practical question: how much information can the parser provide for this or that task?

[3] We will keep the term tree-bank when we refer to a bag of utterance-analysis pairs, where the analyses are syntactic structures according to some linguistic framework, e.g. dependency grammar or phrase-structure grammar.

[4] If ($|T| == 0$) then $Precision(T, G) = 0$ by definition.

course).

In general, a *linguistic concept* is viewed as a set of word sequences, e.g. the noun-phrase concept consists of a sequence of all noun constituents. Here, we take a slightly different perspective on this notion: a concept is a probability distribution over word sequences (Manning and Schutze, 1999). In a generative grammar, a finite set of concepts is specified hierarchically (and recursively). If the concepts are "stricter" or sharper they will tend to be more informative, provided that the strictness captures regularities in the language. This sense of an "informative annotation scheme" is strongly related to the notion of a "compression code" in the communication over a noisy channel view in Information Theory. The more the annotation scheme allows to compress a large corpus of utterances from the language, the more informative this annotation scheme is.

4.3. Cross entropy of an annotation scheme

Technically speaking, in language modeling techniques that originate from the speech community, the "goodness" of a model is captured through the notion of Perplexity of the model on a corpus of utterances. The strongly related notion of *Cross Entropy* is also known from the statistical parsing literature, e.g. (Manning and Schutze, 1999); it captures the average amount of surprise that the model encounters when parsing the utterances in the corpus. A model that captures the regularities in the corpus in a better way, through more adequate syntactic constructs and concepts, will encounter less surprises. How do we apply this idea to parser comparison where we want to measure the informativeness of an annotation scheme (the output of the parser)?

We stress that we would like to compare the outputs of the parsers, rather than their ambiguity resolution capacity[5]. To do so, we suggest a method for measuring a kind of "cross entropy" between each of the tree-banks TB_i and the corpus of utterances C. If the tree-bank parse-trees capture the regularities in the corpus utterances in a better way, the cross entropy will be smaller. But, how do we define this "cross entropy between a tree-bank (rather than a model) and a corpus"?

Although it is not a trivial task, we believe that every tree-bank annotation scheme, whether phrase-structure, head-lexicalized phrase-structure or dependency structure, allows the extraction of a probabilistic model which we will call the "basic generative model". The "basic generative model" must fulfill the following requirements:

1. the rewrite rules of this model must coincide with the atomic units assumed by the linguistic framework, and

2. only the information that exactly coincides with *the logical constraints on the composition] operators*[6] *that originate from the linguistic framework should be available as conditioning context for the model parameters*.

For example, for (context-free) phrase-structure grammars, the logical constraint on the substitution operator is *category substitutability*, which implies that the conditioning context in context-free rule probabilities consists of the label of the left-hand side of the rule, as in standard Probabilistic Context-Free Grammars. Accordingly, it is not suitable to condition the probabilities of the extracted basic model on e.g. the label of the parent node[7]. In effect, beyond the necessary conditions, the basic generative model assumes independence between the different basic rewrite units $r_1 \cdots, r_n$ that generate a parse-tree T, i.e. $P(T) = \prod_{i=1}^{n} P(r_i|\phi(r_i))$, where $P(r_i|\phi(r_i))$ is the relative frequency of r_i in the tree-bank, conditioned on the necessary information only.

Let us consider a few example frameworks from the literature. For phrase-structure annotations, as we just said, the basic model is a Probabilistic Context Free Grammar (PCFG) (Jelinek et al., 1990) (the so-called Tree-bank Grammar (Charniak, 1996)); for a dependency syntax tree-bank, this is a probabilistic generative model in which the dependency probabilities are conditioned on the head-words (see e.g. model 3 of (Eisner, 1996) without conditioning on the preceding child, i.e. 0^{th}-order Markov model for generation of dependents); for a head-lexicalized tree-bank, where head words augment the phrasal nonterminals, the basic model is similar to model 1 of (Collins, 1996) (simplified to exclude "distance measures").

Having extracted a basic generative model from the tree-bank, it becomes easy to specify how the measure of Cross Entropy between the model and the corpus can be estimated. Let model μ be a probabilistic generative model and let C be a corpus of utterances. The cross-entropy is defined by

$$H_\mu(C) = \lim_{|C|\to\infty} \frac{1}{|C|} \sum_{U\in C} \log P(U|\mu)$$

where $|C|$ is the number of utterances U in C. When C is a large corpus, it is possible to estimate this by dropping the limit:

$$\hat{H}_\mu(C) = \frac{1}{|C|} \sum_{U\in C} \log P(U|\mu)$$

Note that for our goal of comparison, it is enough to estimate only roughly the cross entropy on a reasonably large corpus (clearly, the larger the better).

4.4. Per bit recall/precision

For integrating the measures of disambiguation and informativeness of a parser, we will argue for the notion of "per bit disambiguation capacity". This new notion addresses the question: how good is the quality of the output of the parser given its informativeness? This notion is obtained by integrating the PARSEVAL Precision/Recall measures of disambiguation with the Cross Entropy of the annotation scheme (which is the scheme used in the output

[5] The latter issue has been addressed in the recall/precision aspect of the evaluation methodology suggested here.

[6] The composition operators that are used for the construction of parse-trees from the basic rewrite rules.

[7] Note that we say this is not suitable only for the goal of measuring the informativeness of the output of the parser, not for the ability of the parser to disambiguate.

of the parser):

$$Per\ bit\ Recall = \frac{Recall}{CrossEntropy}$$
$$Per\ bit\ Precision = \frac{Precision}{CrossEntropy}$$

The intuition underlying these notions is that frameworks that leave the parses more ambiguous, will demand less effort during parsing and so recall/precision must be discounted accordingly. We claim that the new measures of per bit recall and precision for different parsers can be compared directly, even when the parsers originate from different linguistic frameworks.

4.5. Discussion

We note that there exist various notions that are strongly related to Cross Entropy as a measure of model goodness. One of these notions is the *description length* (or the theoretical Kolmogorov complexity), another is the *message length* (Rissanen, 1983). However, the Cross Entropy measure is the most directly applicable among these closely related notions because of its direct interpretation in terms of smoothed relative frequency.

We expect two issues to constitute the critical points in the application of the methodology proposed in this paper: (1) the development of parallel tree-banks for the same corpus of utterances, (2) the benchmarking of methods for the extraction of basic probabilistic models from these tree-banks. The first issue is critical because it is labor intensive; the second because it demands further specification of what constitutes a basic model that can be extracted from a newly developed tree-bank, especially when this concerns simplified frameworks such as those used by shallow parsers. Despite of these possible difficulties, we believe that our proposal could provide a theoretical departure point towards workable approximations.

5. Conclusions

We presented a preliminary, informal discussion of what it takes to develop a methodology for parser comparison across different linguistic frameworks. We have presented a simple Information Theoretic approach to avoid the problems that arise in mapping between different linguistic frameworks. This approach takes into account the specific concepts of the framework, circumventing the problem of information loss.

As a positive side to this proposal, we envision that for a given domain of language use there will be different tree-banks, each according to a different linguistic framework. This will allow the development of automatic approximations for domain-specific mappings between the different frameworks (e.g. using Machine Learning techniques). Furthermore, this also enables the exploration of complementary aspects of the different existing linguistic frameworks, possibly leading to better stochastic parsers.

Future work in this direction might concentrate on evaluating existing parsers from dependency and phrase-structure grammar, and comparing them using the present approach. Another line of work might concentrate on the application of Machine Learning or stochastic methods to the induction of approximate mappings between these different frameworks. Finally, we are intrigued by the possibility of empirical studies that combine aspects from different frameworks based on parallel tree-banks.

Acknowledgments: We thank Maarten de Rijke and Gabriel Infante-Lopez for discussion and comments on an earlier version. This research was supported by a grant from the Netherlands Organization for Scientific Research (NWO), under project number 220-80-001.

6. References

E. Black et al. 1991. A procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Proceedings of the February 1991 DARPA Speech and Natural Language Workshop*.

R. Bod. 1995. *Enriching Linguistics with Statistics: Performance models of Natural Language*. PhD thesis, ILLC-dissertation series 1995-14, University of Amsterdam.

E. Brill. 1993. *Transformation-Based Learning*. Phd Thesis , University of Pennsylvania.

S. Buchholz, J. Veenstra, and W. Daelemans. 1999. Cascaded grammatical relation assignment. In *In Proceedings of EMNLP/VLC-99*, pages 239–246.

J. Carroll and D. Weir. 1997. Encoding frequency information in lexicalized grammars. In *In ACL/SIGPARSE workshop on Parsing Technologies (IWPT), MIT, Cambridge. Also to appear in Data Oriented Parsing, R. Bod, R. Scha and K. Sima'an (editors), CSLI publications, 2002.*

J. Carroll, T. Briscoe, N. Calzolari, S. Federici, S. Montemagni, V. Pirrelli, G. Grefenstette, A. Sanfilippo, G. Carroll, and M. Rooth. 1996. Sparkle poject. http://www.ilc.pi.cnr.it/sparkle/wp1-prefinal/node10.html.

J. Carroll, T. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain.

J. Carroll, G. Minnen, and T. Briscoe. 1999. Corpus annotation for parser evaluation. In *Poroceedings of the EACL-99 Post-conference Workshop on Linguistically Interpreted Corpora*, pages 35–41, Bergen, Norway.

E. Charniak. 1996. Tree-bank Grammars. In *Proceedings AAAI'96*, Portland, Oregon.

D. Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of the* 38^{th} *Annual Meeting of the Association for Computational Linguistics (ACL'00)*, pages 456–463, Hong Kong, China.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 184–191.

M. Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the EACL*, pages 16–23, Madrid, Spain.

Michael A. Covington. 1992. GB Theory as Dependency Grammar. Technical Report AI-1992-03, Athens, GA.

Michael A. Covington. 1994. An empirically motivated reinterpretation of dependency grammar. Technical Report AI-1994-01, Athens, GA.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*, pages 340–245, Copenhagen, Denmark.

J. Eisner. 2001. Smoothing a probabilistic lexicon via syntactic transformations. PhD thesis, Department of Computer Science, UPenn.

J.T. Goodman. 1998. *Parsing Inside-Out*. PhD thesis, Departement of Computer Science, Harvard University, Cambridge, Massachusetts, May.

R. Hudson. 1984. *Word Grammar*. Basil Blackwell.

F. Jelinek, J.D. Lafferty, and R.L. Mercer. 1990. *Basic Methods of Probabilistic Context Free Grammars, Technical Report IBM RC 16374 (#72684)*. Yorktown Heights.

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

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain, May.

D. Magerman. 1993. Parsing as statistical pattern recognition. PhD Thesis, Stanford University.

C. Manning and H. Schutze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

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

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proceedings of Empirical Methods in NLP, EMNLP-2*, pages 1–10.

J. Rissanen. 1983. A universal prior for integers and estimation by minimum description length. *The Annuals of Statistics*, 11 (2):416–431.

G. Schneider. 1998. A linguistic comparison of constitency, dependency and link grammar. Master thesis, Institut fur Informatik, Universitat Zurich.

K. Sima'an. 2000. Tree-gram Parsing: Lexical Dependencies and Structual Relations. In *Proceedings of the* 38^{th} *Annual Meeting of the Association for Computational Linguistics (ACL'00)*, pages 53–60, Hong Kong, China.

Temperley D Sleator D. 1991. Parsing english with a link grammar. Technical report, Carnegie Mellon University Computer Science.

Erik F. Tjong Kim Sang and Hervé Déjean. 2001. Introduction to the conll-2001 shared task: Clause identification. In *Proceedings of CoNLL-2001*, pages 53–57. Toulouse, France.

Evaluation of the Gramotron Parser for German

Franz Beil[1], Detlef Prescher[2], Helmut Schmid[3], Sabine Schulte im Walde[3]

[1]TEMIS, Rue de Ponthieu 59, 75008 Paris, France
franz.beil@temis-group.com

[2]DFKI GmbH, Language Technology Lab, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
prescher@dfki.de

[3]IMS, Universität Stuttgart, Azenbergstr. 12, 70174 Stuttgart, Germany.
{Helmut.Schmid,Sabine.Schulte.im.Walde}@IMS.Uni-Stuttgart.DE

Abstract

The paper describes an experiment in inside-outside estimation of a lexicalized probabilistic context free grammar for German. Grammar and formalism features which make the experiment feasible are described. Successive models are evaluated on precision and recall of phrase markup consisting of labels for noun chunks and subcategorization frames. Our approach to parsing is a blend of symbolic and stochastic methods where we use evaluation results in both incremental grammar development and validation of selected output to be used in lexical semantic clustering. Our results are that (i) scrambling-style free phrase order, case morphology, subcategorization, and NP-internal gender, number and case agreement can be dealt within a lexicalized probabilistic context-free grammar formalism, and (ii) inside-outside estimation appears to be beneficial, however relies on a carefully built grammar and an evaluation based on carefully selected linguistic criteria. Additionally, we report experiments on overtraining with inside-outside estimation, especially focusing on comparison of the results of mathematical and linguistic evaluations.

1. Introduction

From 1997 to 2000, the Gramotron group of the Institute for Natural Language Processing at Stuttgart University developed a stochastic parser for German (Beil et al. (1999), Schulte im Walde et al. (2001)). The symbolic component of the final parsing system is a manually written context-free grammar consisting of several thousand head-marked rules. Its stochastic component consists of probability weights assigned to the lexicalised grammar rules and to the lexical choice events by the so-called inside-outside algorithm (Lari and Young, 1990), the standard procedure for unsupervised training of a stochastic context-free grammar parsing free text. For training and parsing, the implementations of Carroll (1997b) and Schmid (1999a) were used.

The Gramotron parsing system was designed to be used for the induction of a semantically annotated lexicon of German nouns and verbs (Rooth et al., 1999). Accordingly, the grammar development focus was on the recognition of the grammatical relations between nouns and verbs.

Furthermore, since the parsing results were an intermediate step in an experiment to learn a semantic lexicon, reliable parsing results had to be acquired rapidly. We decided for an incremental grammar development, thus minimizing grammar development efforts in the early project phase.

The context-free grammar for German was developed in three stages: for (i) verb-final clauses, (ii) relative clauses, and (iii) verb-first and verb-second clauses. In this paper, we describe a concluded experiment and evaluation of the parsing system covering constructions (i) and (ii).

Grammar development and stochastic training was controlled by two types of evaluation: (i) an information-theoretic evaluation based on perplexity values measured on training and test corpora of free text, and (ii) a linguistic evaluation of noun chunks with case features and verb frame recognition on a manually annotated test corpus.

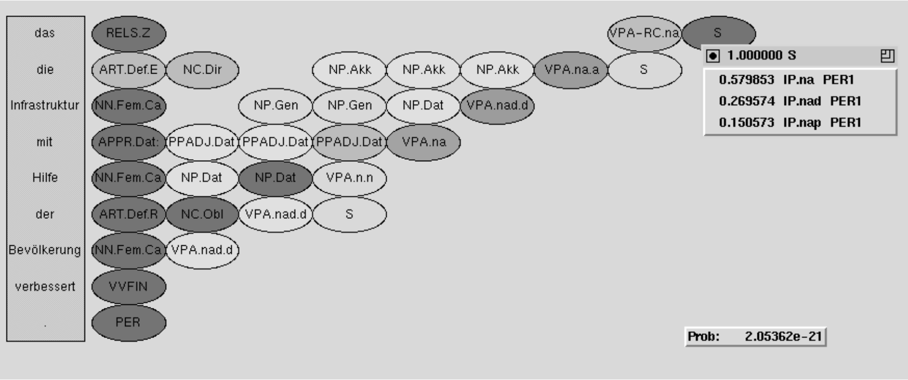
2. Data

The data for our experiments are two sub-corpora extracted from a 200 million token newspaper corpus, (a) a sub-corpus containing 450,000 verb-final clauses with a total of 4 million words, and (b) a sub-corpus containing 1,1 million relative clauses with a total of 10 million words. Apart from non-finite clauses as verbal arguments, there are no further clausal embeddings, and the clauses do not contain any punctuation except for a terminal period. The average clause length is 9.16 and 9.12 words per clause, respectively.

We used a finite-state morphological analyser (Schiller and Stöckert, 1995) to assign multiple morphological features such as part-of-speech tag, case, gender and number to the corpus words, partly collapsed to reduce the number of analyses. For example, the word *Bleibe* (either the case ambiguous feminine singular noun ‘residence’ or a person and mode ambiguous finite singular present tense verb form of ‘stay’) is analysed as follows:

```
analyse> Bleibe
1. Bleibe+NN.Fem.Akk.Sg
2. Bleibe+NN.Fem.Dat.Sg
3. Bleibe+NN.Fem.Gen.Sg
4. Bleibe+NN.Fem.Nom.Sg
5. *bleiben+V.1.Sg.Pres.Ind
6. *bleiben+V.1.Sg.Pres.Konj
7. *bleiben+V.3.Sg.Pres.Konj
```

Reducing the ambiguous categories leaves the two morphological analyses



```
1.000000 S
0.579853  IP.na  PER1
0.269574  IP.nad  PER1
0.150573  IP.nap  PER1
```

```
Prob:   2.05362e-21
```

Figure 1: Chart Browser for Grammar Development

```
Bleibe { NN.Fem.Cas.Sg, VVFIN }
```

Apart from assigning morphological analyses the tool in addition serves as lemmatiser (cf. (Schulze, 1996)).

3. The German Context-Free Grammar

The context-free grammar consists of 5,033 rules with lexical head markings. With very few exceptions (rules for coordination, S-rule), the rules do not have more than two daughters. The 220 terminal categories in the grammar correspond to the collapsed corpus tags assigned by the morphology.

Grammar development is facilitated by (a) grammar development environment of the feature-based grammar formalism YAP (Schmid, 1999b), and (b) a chart browser that permits a quick and efficient discovery of grammar bugs (Carroll, 1997a). Figure 1 shows that the ambiguity in the chart is quite considerable even though grammar and corpus are restricted.

The grammar covers 92.43% of the verb-final and 91.70% of the relative clauses, i.e. the respective part of the corpora are assigned parses.

In the following, we describe two essential parts of the grammar, the noun chunks and the definition of subcategorisation frames. For details concerning prepositional phrases, adjectival chunks, adverbial chunks, complex determiners, and the treatment of coordination see (Schulte im Walde, 2000).

3.1. Noun Chunks (NCs)

On nominal categories, in addition to the four cases Nom, Gen, Dat, and Akk, case features with a disjunctive interpretation (such as Dir for Nom or Akk) are used. The grammar is written in such a way that non-disjunctive features are introduced high up in the tree. Figure 2 illustrates the use of disjunctive features in noun projections: the terminal NN contains the four-way ambiguous Cas case feature; the N-bar (NN1) and noun chunk NC projections disambiguate to two-way ambiguous case features Dir and Obl; the weak/strong (Sw/St) feature of NN1 allows or prevents combination with a determiner, respectively; only at the noun phrase NP projection level, the case feature appears in disambiguated form. The use of disjunctive case features results in some reduction in the size of the parse forest. Essentially the full range of agreement inside the noun phrase is enforced. Agreement between the subject NP and the tensed verb is not enforced by the grammar, in order to control the number of parameters and rules.

The noun chunk definition refers to Abney's chunk grammar organisation (Abney, 1996): the noun chunk (NC) is a projection that excludes post-head complements and (adverbial) adjuncts introduced higher than pre-head modifiers and determiners, but includes participial pre-modifiers with their complements.

3.2. Subcategorisation Frames

The grammar distinguishes four subcategorisation frame classes: active (VPA), passive (VPP), non-finite (VPI) frames, and copula constructions (VPK). A frame may have maximally three arguments. Possible arguments in the frames are nominative (n), dative (d) and accusative (a) NPs, reflexive pronouns (r), PPs (p), and non-finite VPs (i). The grammar does not distinguish plain non-finite VPs from *zu*-non-finite VPs. The grammar is designed to distinguish between PPs representing a verbal complement or adjunct: only complements are referred to by the frame type. The number and the types of frames in the different frame classes are given in Table 1.

Frame Class	#	Frame Types
VPA	16	n, na, nd, np, nad, nap, ndp ni, di, nai, ndi nr, nar, ndr, npr, nir
VPP	18	n, np-s, d, dp-s, p, pp-s nd, ndp-s, np, npp-s, dp, dpp-s i, ip-s, ni, nip-s, di, dip-s
VPI	8	-, a, d, p, r, ad, ap, dp, pr
VPK	2	n, i

Table 1: Subcategorisation Frame Types

German, being a language with comparatively free phrase order, allows for scrambling of arguments. Scrambling is reflected in the particular sequence in which the arguments of the verb frame are saturated. Compare Figure 3 as example of a canonical subject-object order within an active transitive frame *der sie liebt* 'who loves her' and its scrambled object-subject order *den sie liebt* 'whom she loves'.

Abstracting from the active and passive realisation of

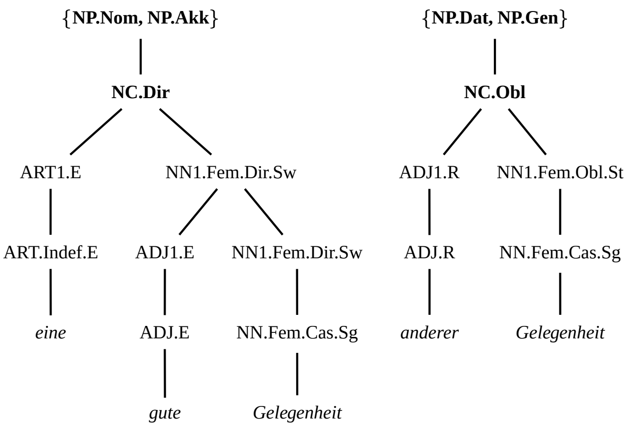


Figure 2: Noun Projections

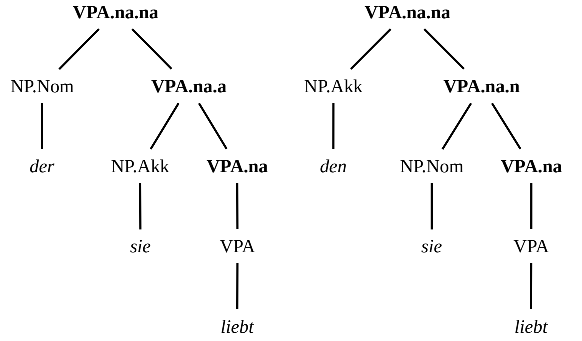


Figure 3: Realising Scrambling Effect in the Grammar Rules

an identical underlying deep-level syntax we generalise over the alternation by defining a top-level subcategorisation frame type, e.g. `IP.nad` for `VPA.nad`, `VPP.nd` and `VPP.ndp-s` (with `p-s` a prepositional phrase within passive frame types representing the deep-structure subject, realisable only by PPs headed by *von* or *durch* 'by'); see Figure 4 for an example, presenting the relative clauses *der die Frau verfolgt* 'who follows the woman', *die verfolgt wird* 'who is followed' and *die von dem Mann verfolgt wird* 'who is followed by the man'.

4. Probability Model

The probabilistic grammars are parsed with `LoPar`[1] (Schmid, 1999a), a head-lexicalised probabilistic context-free parser. The parser is an implementation of the Left-Corner algorithm for parsing and of the Inside-Outside algorithm for parameter estimation. Probabilistic context-free parsing (Lari and Young, 1990) maps a CFG to a probability model by assigning a probability to each grammar rule.

Innovative features of LoPar are head lexicalisation, lemmatisation, parameter pooling, and a sophisticated smoothing technique.

Syntactically, a head-lexicalised probabilistic context-free grammar (HPCFG) (Carroll, 1995; Carroll and Rooth, 1998) is a PCFG in which one of the right hand side categories of each grammar rule is marked as the head of the projection. The lexical head of a terminal category is the respective word form. Thus, lexical head properties, i.e. words, are propagated through head chains.

HPCFGs assign the following probability[2] to a parse tree `T`:

$$\begin{aligned} P(T) \quad = \quad & P_{start}(\text{cat}(\text{root}(T))) \cdot \\ & P_{start}(\text{head}(\text{root}(T))|\text{cat}(\text{root}(T))) \cdot \\ & \prod_{\substack{n \in T \\ n\,:\ \text{non-terminal}}} P_{rule}(\text{rule}(n)|\text{cat}(n), \text{head}(n)) \cdot \\ & \prod_{\substack{n \in T \\ n \neq root(T)}} P_{choice}(\text{head}(n)|\text{cat}(n), \text{cat}(\text{p}(n)), \text{head}(\text{p}(n))) \cdot \\ & \prod_{\substack{n \in T \\ n\,:\ \text{terminal}}} P_{rule}(\texttt{<terminal>}|\text{cat}(n), \text{head}(n)) \cdot \\ & \prod_{\substack{n \in T \\ n\,:\ \text{terminal}}} P_{lex}(\text{word}(n)|\text{cat}(n), \text{head}(n)) \end{aligned}$$

Five families of probability distributions are relevant here. $P_{start}(C)$ is the probability that C is the category of the root node of a parse tree. $P_{start}(h|C)$ is the probability that a root node of category C bears the lexical head h. $P_{rule}(r|C, h)$ is the probability that a node of category C with lexical head h is expanded by rule r. $P_{choice}(h|C, C_p, h_p)$ is the probability that a (non-head) node of category C has the lexical head h given that the parent category is C_p and the parent head is h_p. $P_{rule}(\texttt{<terminal>}|C, h)$ is the probability that a node of category C with lexical head h is a terminal node. $P_{lex}(w|C, h)$, finally, is the probability that a terminal node with category C and lexical head h expands to the word form w.

In order to reduce the prohibitively large number of lexical parameters that have to be estimated, we employed linguistic generalisations for parameter reduction: lemmatisation and parameter pooling. Using uninflected lemma rather than inflected word form for lexicalisation eliminates splitting of estimated frequencies among inflectional forms. Parameter pooling is based on the assumption that lexical choice probabilities are unlikely to depend on inflectional features like gender, case, number etc. of categories or argument order in verb frames. For instance, there are (at least) nine different inflectional patterns for projecting the adjective *alt* (old) and *Buch* (book) to an `NN1` category. Instead of assigning a lexical choice probability

$$P_{choice}(alt|\texttt{ADJ.w}_\texttt{i}, \texttt{NN1.x}_\texttt{i}.\texttt{y}_\texttt{i}.\texttt{z}_\texttt{i}, Buch)$$

[1] LoPar is basically a re-implementation of the Galacsy tools which were developed by Glenn Carroll in the SFB, but LoPar provides additional functionality.

[2] The auxiliary functions `cat`, `head`, `p(arent)`, `word` and `rule` return the syntactic category, the lexical head, the parent node, the dominated word or the expanding grammar rule of a node. `root` returns the root node of a parse tree and `<terminal>` is a constant.

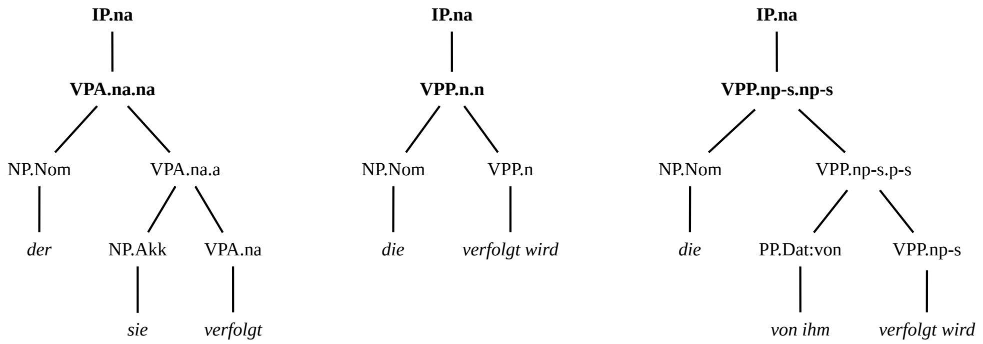


Figure 4: Generalising over the Active-Passive Alternation of Subcategorisation Frames

for each possible combination of w, x, y, z, the combinations are pooled to a single distribution

$$P_{choice}(alt|\texttt{ADJ}, \texttt{NN1}, \text{Buch})$$

for all inflectional variations of `NN1 -> ADJ NN1`. We obtain a single probability distribution for adjectival modifiers. In result, frequent observation of *altes Buch* in the trainng data also increases the probability of *alter Bücher*. For argument filling into verb frames, categories of the form `VP.x.y` are pooled to `VP.x` and active, passive and non-finite verb frames are pooled according to shared arguments, disregarding the saturation state of the frame. For instance, P_{choice} of a particular noun is the same as accusative NP head in the transitive active frame or nominative NP head in the passive frame of a particular verb (*[dass] sie den Hund füttert 'she feeds the dog', der Hund gefüttert wird 'the dog is fed'*).

5. Grammar Training

5.1. Training Strategy

The training in our main experiment was performed in the following steps:

1. Initialisation of all CFG rules with identical frequencies. (Comparative initialisations with random frequencies had no effect on the model development.)

2. Unlexicalised training: The training corpus was parsed once, re-estimating the frequencies twice.

3. Lexicalisation: The unlexicalised model was turned into a lexicalised model by (i) setting the probabilities of the lexicalised rule probabilities to the values of the respective unlexicalised probabilities and (ii) initialising the lexical choice and lexicalised start probabilities uniformly.

4. Lexicalised training:
   Three training iterations were performed on the training corpus, re-estimating the frequencies after each iteration.

For training the model parameters we used 90% of the corpora, a total of 1.4 million clauses. The remaining 10% of serve as heldout data to measure overtraining.

Our experiments have shown that training an unlexicalised model first improves overall results. The optimal training strategy proceeds with few parameter re-estimations of an unlexicalised model. Without re-estimations or with a large number of re-estimations the model was effected to its disadvantage. With less unlexicalised training more changes during lexicalised training take place later on.

Comparative numbers of iterations (up to 40 iterations) in lexicalised training showed that more iterations did not have any further effect on the model.

6. Evaluation

Our evaluation methods were chosen to monitor the development of the grammar, to control the grammar training, and compare different training regimes. As part of our larger project of lexical semantic clustering, the parsing system had the specific task to collect corpus frequencies for pairs of a verbal head and its subcategorisation frame and frequencies for the nominal fillers of slots in a subcategorisation frame. The linguistic evaluation focuses on the reliability of these parsing results.

6.1. Mathematical evaluation

A		B		C	
1:	52.0199	1:	53.7654	1:	49.8165
2:	25.3652	2:	26.3184	2:	23.1008
3:	24.5905	3:	25.5035	3:	22.4479
⋮	⋮	⋮	⋮	⋮	⋮
15:	24.2861	57:	25.0549	90:	22.1443
16:	24.2861	58:	25.0549	95:	22.1443
17:	24.2867	59:	25.055	96:	22.1444

Table 2: Overtraining (iteration: cross-entropy on heldout data)

In order to control the amount of unlexicalised training, we measured overtraining by comparing the perplexity of the model on training and heldout data (or, respectively, cross-entropy[3] on heldout data in the experiments

[3] For a corpus consisting of sentences of a certain average length (`avg`), one can easily transform these cross-entropy values (`cross`) to the better known values of word perplexity (perp)

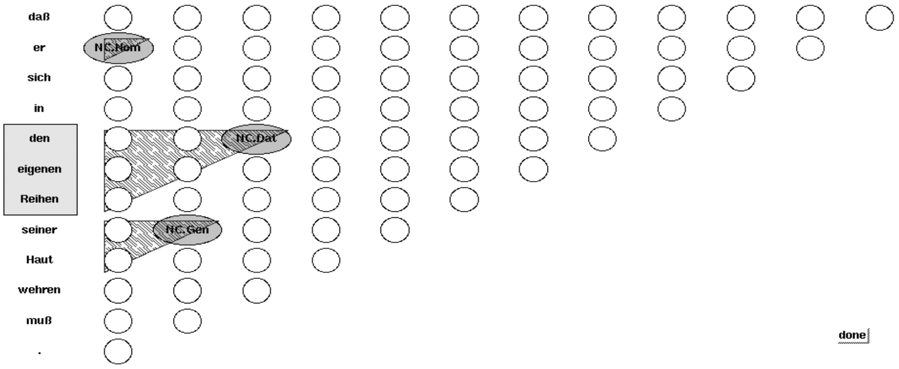


Figure 5: Chart Browser for manual constituent markup

in (Beil et al., 1999)). While perplexity on training data is theoretically guaranteed to converge through subsequent iterations, increasing perplexity on heldout data indicates overtraining. Table 2 shows comparisons of different sizes of training and heldout data (training/heldout) for unlexicalised training in an older experiment (Beil et al., 1999): (A) 50k/50k, (B) 500k/500k, (C) 4.1M/500k. The overtraining effect is indicated by the increase in cross-entropy from the penultimate to the ultimate iteration in the tables.

In previous experiments (Beil et al., 1999), we compared in more detail the mathematical evaluation with the linguistic evaluation of precision/recall measures on categories of different complexity through iterative unlexicalised training. The comparison shows that the mathematical criterion of overtraining may lead to bad results from a linguistic point of view. While precision/recall measures for low-level structures such as NCs converge, iterative unlexicalised training up to the overtraining threshold is disadvantageous for the evaluation of complex categories like subcategorisation frames. We observed precision/recall values for verb frames settling even below the results with a randomly initialised grammar. So the mathematical evaluation can only serve as a rough indicator whether the model reaches towards an optimum, but linguistic evaluation determines the optimum.

6.2. Linguistic evaluation

Although an appropriate treebank is available for German (the NEGRA treebank, cf. Skut et al. (1997) for an overview), we did not use it for our evaluation. One reason for this is the restriction of our initial grammar development to verb final and relative clauses while the treebank, of course, annotates full clauses. It turned out to be difficult to extract respective sub-treebanks. On the other hand, we did not intend to carry out the standard parser evaluation method of measuring precision/recall on phrase boundaries and crossing brackets (the PARSEVAL scheme) for which treebanks are widely used. Bracketing information is rather uninteresting for our objectives and we reckoned that rich structures as generated by our grammar would likely punished by the crossing bracket measure. (For a more general overview of problems using the crossing brackets measure for parser evaluation see (Carroll et al., 1998).)

Moreover, in transforming our bracketing to treebank annotation standards, we feared to loose too much information deemed important for our evaluation. In our efforts to find a transformation that maps treebank structures to a selection of ours (noun and verb chunks), we found two mapping problems: (i) mapping treebank phrase spans to our chunk spans and (ii) finding an information-preserving mapping from our labels to treebank labels. Concerning the first, it turned out to be difficult to define noun chunk ends within treebank NPs. An even harder problem is finding the rich information in our verbal category labels (i.e. type and frame annotation) in treebank VPs.

So we decided to build our own test data: Rather than pursuing the efforts of finding an appropriate treebank-to-gramotron transformation, we performed detailed evaluations of individual frames and of a set of selected verbs.

Test data The linguistic parameters of the models were evaluated concerning the identification of NCs and subcategorisation frames. We randomly extracted 200 relative clauses and 200 verb-final clauses from the test data and hand-annotated the relative clauses with noun chunk labels, and all of the clauses with frame labels. In addition, we extracted 100 randomly chosen relative clauses for each of the six verbs *beteiligen* ‘participate’, *erhalten* ‘receive’, *folgen* ‘follow’, *verbieten* ‘forbid’, *versprechen* ‘promise’, *versuchen* ‘try’, and hand-annotated them with their subcategorisation frames. The particular selection of verbs aims to be representative for the variety of verb frames defined in our grammar.

The manual annotation was facilitated by use of a chart browser. The labellers filled the appropriate chart cells with category names by selecting category labels from a given list that is displayed on clicking a cell. Figure 5 gives an example of NC-labelling which visualises the determination of NC-ranges via cell selection. Frames are annotated as IP

using the formula

$$\texttt{perp} = 10^{\texttt{avg}^{-1}\cdot\texttt{cross}}$$

(assuming that the cross-entropy is computed by a logarithm based on 10). For example, an average lenghth of avg=9.2 and a cross-entropy of cross=24.2 yields a word perplexity perp=427.0, which is a value comparable to the values presented in Schulte im Walde et al. (2001).

labels, i.e. they are always in the same chart cell and frame ranges are trivial.

Best-first consistency Our linguistic evaluation of the probability models is a version of measuring best-first consistency (Briscoe and Carroll, 1993). We made the models determine the Viterbi parses (i.e. maximum probability parses) of the test data and extracted the categories of interest (i.e. noun chunks and subcategorisation frame types). Only the relevant categories but not the entire Viterbi parses were compared with the annotated data. NCs were evaluated according to (i) range and (ii) range and label, i.e. category name. The subcategorisation frames were evaluated according to the frame label only. Precision and recall measures are defined as follows:

$$precision = \frac{correct}{guesses} \qquad recall = \frac{correct}{baseline}$$

with *baseline* referring to the set of annotated categories in the test corpus, *guesses* referring to the set of range/label annotated categories identified in Viterbi parses, and *correct* counting the cases where the chunk/label identified by the parser is a match to the annotator's choice ($correct = guesses \cap baseline$).

Overall results The precision values of the "best" model according to the training strategy were as in Table 3.

Noun Chunks		Subcategorisation Frames on Sub-Corpora	
range	range+label	relative clauses	verb final clauses
98%	92%	63%	73%

Subcategorisation Frames on Specific Verbs					
beteiligen 'participate'	*erhalten* 'receive'	*folgen* 'follow'	*verbieten* 'forbid'	*versprechen* 'promise'	*versuchen* 'try'
48%	61%	88%	59%	80%	49%

Table 3: Precision Values on Noun Chunks and Subcategorisation Frames

For comparison reasons, we evaluated the subcategorisation frames of 200 relative clauses extracted from the training data. Interestingly, there were no striking differences concerning the precision values.

Evaluation of training regimes Figure 6 present the strongly different development of noun chunk and subcategorisation frame representations within the models, ranging from the untrained model until the fifth iteration of lexicalised training. NCs were modelled sufficiently by an unlexicalised trained grammar. Unexpectedly, lexicalisation impaired the modelling slighlty. This observation is supported by related experiments of German noun chunking on an unrestricted text corpus (Schmid and Schulte im Walde, 2000). It remains to be explored whether the number of low-frequent nominal heads is—despite the use of lemmatisation for parameter reduction—still prohibitively large because of the pervasive morpho-syntactic process of noun compounding in German.

Verb phrases in general needed a combination of unlexicalised and lexicalised training, but the representation strongly depended on the specific item. Unlexicalised training advanced frequent phenomena (compare, for example, the representation of the transitive frame with direct object for *erfahren* and with indirect object for *folgen*), lexicalisation and lexicalised training improved the lexicalised properties of the verbs, as expected.

Parameter pooling Regarding the frame evaluation, we also did a test on the effects of parameter pooling in lexicalised traininng. Without pooling of frame categories the precision values for low-frequent phenomena such as non-finite frame recognition was significantly lower, e.g. the precision for the verb *versuchen* was 9% less than with pooling. This result suggests investigations into the importance of training data size and research into other pooling possibilities.

6.3. Error Analysis

A detailed investigation of frame recognition showed the following interesting feature developments:

- Highly common subcategorisation types such as the transitive frame are learned in unlexicalised training and then slightly unlearned in lexicalised training. Less common subcategorisation types such as the demand for an indirect object are unlearned in unlexicalised training, but improved during lexicalised training.

- It is difficult and was not effectively learned to distinguish between prepositional phrases as verbal complements and adjuncts.

- The active present perfect verb complexes and passive of condition were confused, because both are composed by a past participle and a form of *to be*, e.g. *geschwommen ist* 'has swum' vs. *gebunden ist* 'is bound'.

- Copula constructions and passive of condition were confused, again because both may be composed by a past participle and a form of *to be*, e.g. *verboten ist* 'is forbidden' vs. *erfahren ist* 'is experienced'.

- Noun chunks belonging to a subcategorised non-finite clause were partly analysed main verb arguments. For instance, *der ihn zu überreden versucht* 'who him$_{acc}$ tried to persuade' was parsed as demanding an accusative plus a non-finite clause instead of recognising that the accusative object is subcategorised by the embedded infinitival verb.

- Reflexive pronouns may trigger either a reflexive or, by virtue of projecting to an accusative or dative noun chunk, a transitive frame. The correct or wrong choice of frame type containing the reflexive pronoun was learned consequently right or wrong for different verbs. For instance, the verb *sich befinden* 'to be situated' was generally parsed as a transitive, not as inherently reflexive.

6.4. Shortcomings and evaluation alternatives

We are aware that there are some desirable aspects missing from our evaluation.

Firstly, we did not evaluate the relations between lexical heads directly, the main task our parsing system was designed for. Subcategorisation frame and noun chunk label

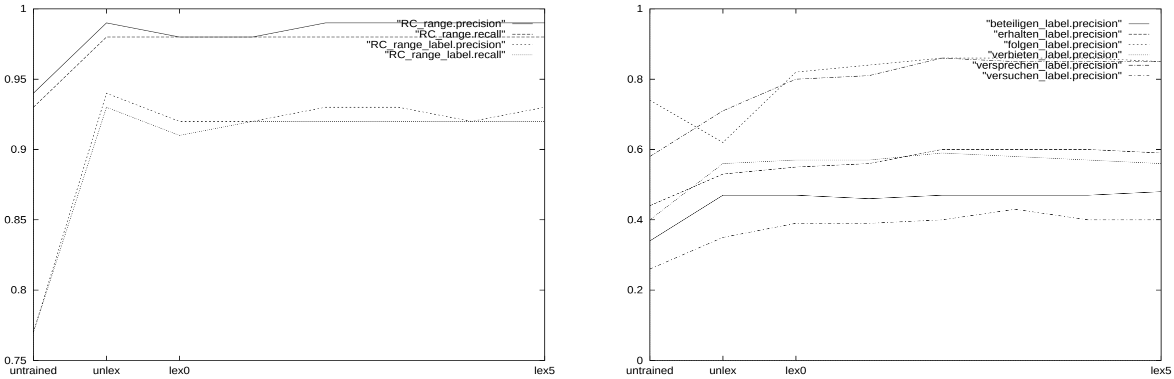


Figure 6: Development of Precision and Recall Values on Noun Chunk Range and Label (left-hand side), and Precision Values on Subcategorisation Frames for Specific Verbs (right-hand side)

recognition serve only as indirect evidence of how well our model does on recognising scrambling of verbal arguments. Because noun chunk annotation is not confined to verb argument slots—PP embedded noun chunks were annotated as well—and a detailed error analysis on noun chunk labels is missing, it remains unclear whether scrambled nominal arguments are subject to more errors than the remarkable 92% precision on NC labels suggests. Similarly, correctly recognised verb frames with a prepositional argument have not been evaluated as to whether the assigned PP argument is actually the correct one.

Secondly, we did not evaluate the correctness of lexical heads of phrases.

Relevant evaluation schemes that capture our shortcomings are the evaluation of dependency structure as described in (Lin, 1995) or the proposal of evaluating of grammatical relations of Carroll et al. (1998). Both evaluation proposals address the importance of selectively evaluating parsing systems with respect to specific types of syntactic phenomena rather than measuring overall performance as in “traditional” evaluation schemes. Selective evaluation is a definite desideratum for our own evaluation task. The proposals also point to a way to automatically extract evaluation relevant relations from an annotated corpus. Inquiring about the feasibility of mapping Negra, the treebank for German, to a respective test corpus will hopefully provide a more comprehensive basis for our future evaluations of head–head relations.

7. Conclusion

Our approach to parsing is a combination of symbolic and stochastic methods. The symbolic component usually involves a very high degree of overgeneration leaving disambiguation to the stochastic component. To facilitate disambiguation by statistical means, the symbolic component relies on certain categorial generalizations and uses non-standard categories to reduce the parameter space or allow for parameter pooling. We used evaluation results in both incremental grammar development and validation of selected output to be used in lexical semantic clustering.

Our principal result is that scrambling-style free-er phrase order, case morphology and subcategorization, and NP-internal gender, number and case agreement can be dealt with in a head-lexicalized PFCG formalism. A second result is that inside-outside estimation appears to be beneficial, however relies on a carefully built grammar where parses can be evaluated by carefully selected linguistic criteria.

Furthermore, we reported experiments on overtraining with inside-outside estimation. These experiments are made possible by the carefully built grammar and our evaluation tools, especially allowing to compare and to relate the results of our mathematical and linguistic evaluation. In combination, these provide a general framework for investigating training regimes for lexicalized PCFGs.

However, there are two relevant aspects missing from our evaluation. First, we did not evaluate grammatical relations directly. Frame and NC case recognition give only a crude idea of how well our model does on recognizing e.g. scrambled subject and direct object. Because NC evaluation is not confined to verb argument slots, the picture is distorted. Second, we did not evaluate the correctness of lexical heads of phrases. Clearly, if we can overcome our difficulties to map Negra, the treebank for German, to a respective test corpus, a more valuable basis for future evaluations of head–head relations supplied by the gramotron parsing system is provided.

Finally, although there is no guarantee that the maximization of the likelihood of the training data (which the inside-outside algorithm performs) also improves the linguistic correctness of the resulting syntactic analyses, our experiments show that in practice this is the case. Gaining more insight into the relationship between linguistic plausibility and likelihood of linguistic analyses will be an interesting future research topic.

8. References

Steven Abney. 1996. Chunk stylebook. Technical report, SfS, Universität Tübingen.

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth. 1999. Inside-outside estimation of a lexicalized PCFG for German. In *Proceeding of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, College Park, Maryland.

Ted Briscoe and John Carroll. 1993. Generalised probabilistic LR parsing for unification-based grammars. *Computational Linguistics*, 19(1):25–60.

Glenn Carroll and Mats Rooth. 1998. Valence induction with a head-lexicalized PCFG. In *Proceedings of EMNLP-3*, Granada.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, Granada, Spain.

Glenn Carroll. 1995. *Learning Probabilistic Grammars for Language Modeling*. Ph.D. thesis, Department of Computer Science, Brown University.

Glenn Carroll, 1997a. *Manual pages for* `charge`*,* `hyparCharge`. IMS, Universität Stuttgart.

Glenn Carroll, 1997b. *Manual pages for* `supar`, `ultra`, `hypar`. IMS, Universität Stuttgart.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *IJCAI-95*.

M. Rooth, S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proc. of ACL'99*.

Anne Schiller and Chris Stöckert, 1995. *DMOR*. IMS, Universität Stuttgart.

Helmut Schmid and Sabine Schulte im Walde. 2000. Robust German Noun Chunking with a Probabilistic Context-Free Grammar. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 726–732, Saarbrücken, Germany, August.

Helmut Schmid, 1999a. *LoPar. Design and Implementation*. Insitut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Helmut Schmid. 1999b. *YAP: Parsing and Disambiguation with Feature-Based Grammars*. Ph.D. thesis, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Sabine Schulte im Walde, Helmut Schmid, Mats Rooth, Stefan Riezler, and Detlef Prescher. 2001. Statistical grammar models and lexicon acquisition. In *Linguistic Form and its Computation*. CSLI, Stanford, CA.

Sabine Schulte im Walde. 2000. The German statistical grammar model: Development, training and linguistic exploitation. Arbeitspapiere des Sonderforschungsbereichs 340 *Linguistic Theory and the Foundations of Computational Linguistics* 162, Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, December.

Bruno Maximilian Schulze, 1996. *GermLem – ein Lemmatisierer für deutsche Textcorpora*. IMS, Universität Stuttgart.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing ANLP-97*, Washington, DC.

Evaluating a Wide-Coverage CCG Parser

Stephen Clark and Julia Hockenmaier

Division of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh. EH8 9LW
Scotland, UK
{stephenc,julia}@cogsci.ed.ac.uk

Abstract

This paper compares three evaluation metrics for a CCG parser trained and tested on a CCG version of the Penn Treebank. The standard Parseval metrics can be applied to the output of this parser; however, these metrics are problematic for CCG, and a comparison with scores given for standard Penn Treebank parsers is uninformative. As an alternative, we consider two evaluations based on head-dependencies; one considers local dependencies defined in terms of the derivation tree, and one considers dependencies defined in terms of the CCG categories. The latter set of dependencies includes long-range dependencies such as those inherent in coordination and extraction phenomena.

1. Introduction

In this paper, we compare the advantages and shortcomings of three evaluation metrics for a statistical parser based on Combinatory Categorial Grammar (CCG, Steedman (2000)). The parser (described in Hockenmaier and Steedman (2002b)) is trained and tested on a treebank of CCG normal-form derivations which has been derived (semi)-automatically from the Penn Treebank (Marcus et al., 1993).

We apply the standard Parseval metrics to compare the derivation trees produced by the parser with those in the gold standard. However, CCG derivation trees are binary-branching, and the set of CCG categories is much larger than the set of nonterminal labels in the Penn Treebank. Therefore, a comparison with Parseval figures given for standard Penn Treebank parsers is uninformative. Furthermore, in the presence of left and right modifiers to the same constituent, there are equivalent normal-form derivations, which Parseval does not take into account.

We also consider two dependency evaluations. Like the standard Penn Treebank parsers of Collins (1999) and Charniak (2000), the CCG parser models word-word dependencies defined in terms of local rule applications. Collins (1999) proposes an evaluation based on these dependencies, which we apply to our parser. This allows a direct comparison with Collins' parser and overcomes the problem of equivalent normal-form derivations.

Unlike the phrase-structure trees returned by standard Penn Treebank parsers, CCG derivations encode the long range dependencies involved in constructions such as raising, control, extraction and coordination. In order to evaluate another CCG parser, Clark et al. (2002) introduce an evaluation which incorporates the long range, as well as local, dependencies. This evaluation is applied to the output of the normal-form parser, using the Clark et al. parser to extract the relevant dependencies from the derivation trees. This evaluation is much closer to the dependency-based evaluations of Lin (1995) and Carroll et al. (1998).

2. Combinatory Categorial Grammar

A CCG grammar consists of a lexicon, which pairs words with lexical categories, and a set of combinatory rules, which specify how categories combine. Categories are either atomic or complex. Examples of atomic categories include $\mathsf{S[dcl]}$ (declarative sentence), NP (noun phrase), N (noun) and PP (prepositional phrase).

Complex categories are functors which express the type and directionality of their arguments, and the type of the result. For example, the category for the transitive verb *bought* specifies that one NP is required to the right of the verb, and one NP to the left, resulting in a sentence:

(1) bought := $(\mathsf{S[dcl]}\backslash\mathsf{NP})/\mathsf{NP}$

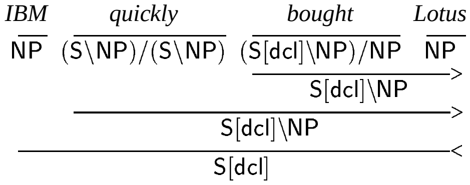
Other examples of complex categories expressing subcategorisation are as follows ([*pt*] denotes a past participle and [pss] denotes a passive):

(2) has := $(\mathsf{S[dcl]}\backslash\mathsf{NP})/(\mathsf{S[pt]}\backslash\mathsf{NP})$
been := $(\mathsf{S[pt]}\backslash\mathsf{NP})/(\mathsf{S[pss]}\backslash\mathsf{NP})$
bought := $(\mathsf{S[pt]}\backslash\mathsf{NP})/\mathsf{NP}$
bought := $\mathsf{S[pss]}\backslash\mathsf{NP}$

Complex categories of the form X/X or $\mathsf{X}\backslash\mathsf{X}$ can express modification:

(3) big := N/N
quickly := $(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$

Constituents combine according to a set of combinatory rules, including function application, function composition and type-raising (see Steedman (2000) for the details). For example, the following derivation uses forward (>) and backward (<) application:

(4)


Composition and type-raising are necessary for certain types of extraction and coordination phenomena. In the following object-extraction example, type-raising ($>T$) first turns the NP for *IBM* into a functor looking for a verb-phrase, which then combines with the category for *bought* using forward composition ($>B$):

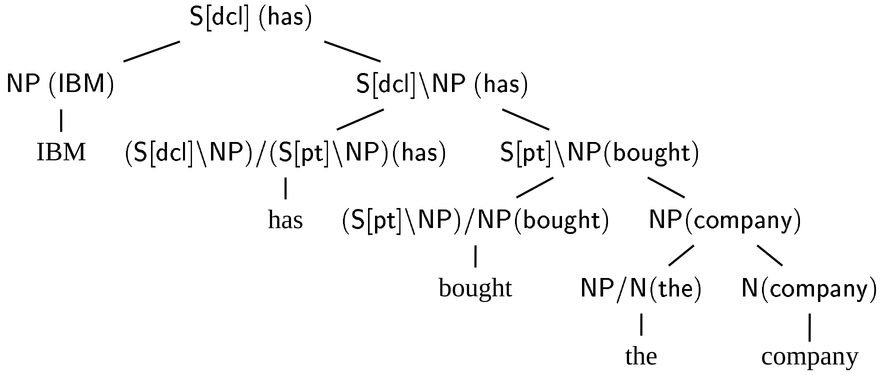
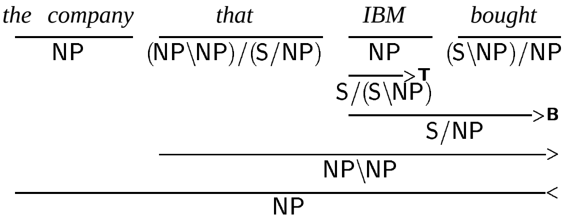


Figure 1: A derivation tree marked with heads

(5)



Note that the use of composition introduces so-called “spurious ambiguity”, in which distinct derivations for a sentence lead to the same semantic interpretation. Even a simple sentence such as *IBM bought Lotus* has several derivations, one using only function application, and the others using type-raising and composition. However, all derivations lead to the same interpretation: that *IBM* is the buyer and *Lotus* is the buyee.

One solution to the problem of spurious ambiguity is to only apply function composition when syntactically necessary; such a derivation is called *normal-form*. The corpus that we use to train and test the parser described here contains only normal-form derivations.

3. The parser

The parser that we evaluate is described in Hockenmaier and Steedman (2002b), and is based on a generative model of CCG derivation trees. Like most recent work in statistical parsing – including the generative models of Collins (1997) and Charniak (2000) – the parser models the word-word dependencies defined by local subtrees. Each constituent is assumed to have one lexical head (a word and its lexical category). The example derivation in Figure 1 shows how heads are percolated through the derivation tree.

The statistical model assumes a top-down tree-generating process in which heads are generated at the maximal projection of a constituent. Unless this maximal projection is the root of the entire tree, the constituent is a complement or adjunct of another constituent, and there is a dependency between the the heads of both constituents. This dependency is expressed in the statistical model by conditioning the head of complements or adjuncts on the head of the parent node and the local tree which defines the dependency relation. For example, in Figure 1, *bought* is not only conditioned on its lexical category (S[pt]\NP)/NP, but also on the fact that it appears within a local tree with head word *has*, parent S[dcl]\NP, left (head) daughter (S[dcl]\NP)/(S[pt]\NP) and right (non-head) daughter S[pt]\NP.

The parser is trained and tested on a treebank of CCG normal-form derivations. This corpus, which we call CCGbank, has been derived (semi)-automatically from the Penn Treebank (Marcus et al., 1993), using sections 02-21 for training and section 23 for testing. For further details of CCGbank we refer readers to Hockenmaier and Steedman (2002a).

4. Evaluation metrics

This section describes the different evaluation metrics, which we illustrate by evaluating the (fictitious) output tree in the bottom of figure 2 against the correct derivation given in the top of figure 2.

4.1. Parseval

The first measures are the standard Parseval metrics bracketed precision/recall and labelled precision/recall used to compare the normal-form derivation trees produced by the parser with those in the gold standard (section 23 of the CCGbank).

Following common practice, we disregard punctuation marks. Since CCG derivation trees are at most binary branching, punctuation marks introduce a separate level into the tree, which we also disregard in the evaluation.

Consider the trees given in figure 2. Discarding the lexical categories (but not their unary projections), the gold standard has six nodes, three of which are correctly identified in the output tree. The output tree has seven nonterminal nodes. Hence, labelled and bracketed precision are both 3/7; labelled and bracketed recall are both 3/6. Note that Parseval does not take the correctness of lexical categories into account, which is important for CCG since categories encode subcategorisation information. Therefore, we also give the accuracy of lexical categories (again disregarding punctuation marks), which in this case is 4/6.

4.2. Dependency evaluation 1

Collins (1999) gives an alternative evaluation to Parseval, measuring the recovery of word-word dependencies. According to his definition, there is a dependency between two words w and w' if the parse contains a local tree such that w' is the head of this tree and w is the head of a non-

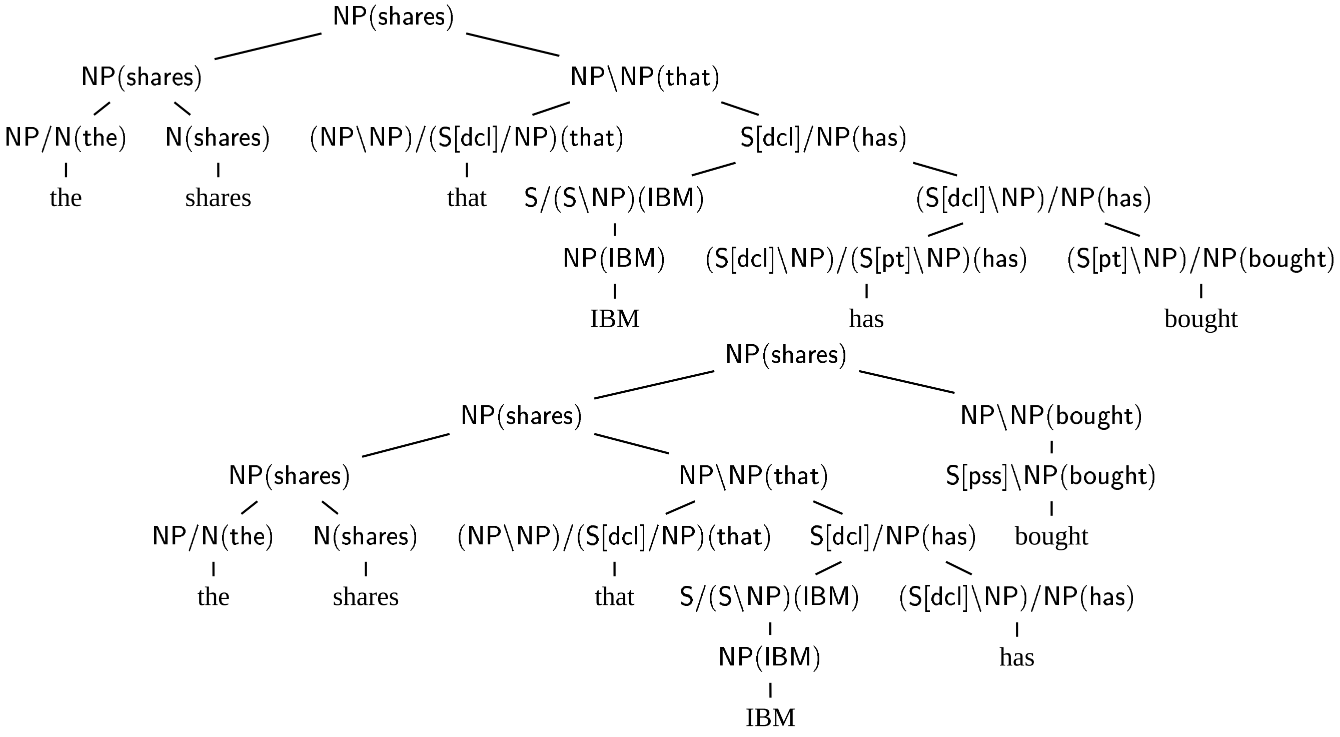
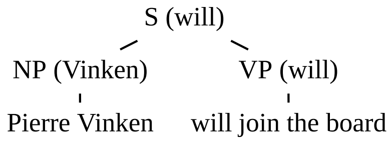


Figure 2: Example trees for evaluation: the top tree is the gold standard.

head daughter. The following tree defines a dependency between *Vinken* and *will*:



The dependency relation is determined by the label of the parent node (S), the label of the head daughter (VP), the label of the non-head daughter (NP), and the direction of the non-head daughter (left): $\langle S, VP, NP, left\rangle$. Furthermore, if the non-head daughter is a complement, its category carries a complement feature −C. In Collins' original evaluation, coordinate constructions are distinguished by a further element CC. We adapt this evaluation to CCG; however, since the directionality of the head is directly encoded in the categories, there is no need for this feature. A similar comment applies to the complement feature. Also, in CCGbank, binary nodes within a coordinate construction carry a special coordination feature, and so the CC-feature is redundant as well.

The way in which these dependencies are defined means there is exactly one relation to be determined for each word. There is a special relation for the head of the sentence (which is not dependent on any other word). Collins gives scores for labelled and unlabelled dependencies. Unlabelled dependency scores only take into account whether there is a relation between w and w' such that w' is the head and w its modifier or complement, but not whether the local tree which defines this dependency is correctly labelled.

Returning to our example, the gold standard in figure 2 defines the following dependencies:

Relation $\langle \mathit{Parent}, \mathit{Head}, \mathit{Sister}\rangle$	**Head**	**Dep**
$\langle \mathsf{NP}, \mathsf{N}, \mathsf{NP/N}\rangle$	*shares*	*the*
$\langle$Head$\rangle$		*shares*
$\langle \mathsf{NP}, \mathsf{NP}, \mathsf{NP\backslash NP}\rangle$	*shares*	*that*
$\langle \mathsf{NP\backslash NP}, \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathsf{S[dcl]/NP}\rangle$	*that*	*has*
$\langle \mathsf{S[dcl]/NP}, \mathsf{(S[dcl]\backslash NP)/NP}, \mathsf{S/(S\backslash NP)}\rangle$	*has*	*IBM*
$\langle \mathsf{(S[dcl]\backslash NP)/NP}, \mathsf{(S[dcl]\backslash NP)/(S[pt]\backslash NP)}, \mathsf{(S[pt]\backslash NP)/NP}\rangle$	*has*	*bought*

These are the dependencies in the incorrect analysis:

Relation $\langle \mathit{Parent}, \mathit{Head}, \mathit{Sister}\rangle$	**Head**	**Dep**
$\langle \mathsf{NP}, \mathsf{N}, \mathsf{NP/N}\rangle$	*shares*	*the*
$\langle$Head$\rangle$		*shares*
$\langle \mathsf{NP}, \mathsf{NP}, \mathsf{NP\backslash NP}\rangle$	*shares*	*that*
$\langle \mathsf{NP\backslash NP}, \mathsf{(NP\backslash NP)/(S[dcl]/NP)}, \mathsf{S[dcl]/NP}\rangle$	*that*	*has*
$\langle \mathsf{S[dcl]/NP}, \mathsf{(S[dcl]\backslash NP)/NP}, \mathsf{S/(S\backslash NP)}\rangle$	*has*	*IBM*
$\langle \mathsf{NP}, \mathsf{NP}, \mathsf{NP\backslash NP}\rangle$	*shares*	*bought*

Thus, according to this measure, five out of six dependencies are correct. Note that this measure is not always affected by errors in the lexical categories. For example, the dependency between *has* and *that* is considered correct, even though the gold standard analyses *has* as an auxiliary and the incorrect derivation analyses *has* as a transitive verb.

4.3. Dependency evaluation 2

The parser in Clark et al. (2002) can be used to yield a third measure. This parser uses CCG categories extended with head and dependency information and captures the "deep" dependencies inherent in cases such as raising, control, and extraction and coordination phenomena, as well as the standard local dependencies. Figure 3 is an example from Clark et al. (2002), with the links expressing dependencies. (The labels are omitted for clarity.) Note that *investors* and *managers* are both subjects of *want*, and subjects of *lock*.

An example of an extended category for the transitive verb *bought* is as follows:

(6) bought := $(\mathsf{S}_{bought}\backslash\mathsf{NP}_{1})/\mathsf{NP}_{2}$

There are two dependency relations encoded: the subject of the transitive verb – here marked 1 – and the direct object – here marked 2. The subscript on the S category indicates that the head of the resulting sentence is *bought*. Since the argument slots in CCG categories correspond closely to the grammatical relations used by Carroll et al. (1998), this dependency evaluation is very much in the spirit of the Carroll et al. evaluation (and that of Lin (1995)).

A dependency is formally defined as a 4-tuple: $\langle h_f, f, s, h_a\rangle$, where h_f is the head word of the functor, f is the functor category (extended with dependency information), s is the argument slot, and h_a is the head word of the argument. For example, in the sentence *IBM bought Lotus*, the subject-verb dependency is as follows:

(7) $\langle \mathit{bought}, (\mathsf{S}\backslash\mathsf{NP}_{1})/\mathsf{NP}_{2}, 1, \mathit{IBM}\rangle$

The category set used by the parser consists of 398 category types (chosen according to frequency), derived automatically from the CCGbank. Each category has been manually marked-up with head and dependency information, and at this stage we encode every argument slot as a dependency relation. In future work we may use only a subset of the argument slots.

In order to recover such dependencies from the trees produced by the normal-form parser, the Clark et al. (2002) parser is run over the trees output by the normal-form parser, tracing out the derivations and outputting the dependencies. This method can also be applied to the trees in the test set, in order to provide a set of gold standard dependency structures. Note that the marked-up categories used by the Clark et al. parser are necessary to obtain these dependencies; without this information, they cannot be derived from the local dependencies used in the first dependency evaluation.

The evaluation metrics we use are precision and recall over the dependencies (labelled and unlabelled). To obtain a point for a labelled dependency, the head, dependent, functor category, and slot must all be correct. To obtain a point for an unlabelled dependency, the head and dependent must have appeared together in some relation (in any order) in the gold standard. The dependencies obtained from the trees in Figure 2 are given in table 1. The scores for the incorrect tree are 3/6 labelled precision, 3/7 labelled recall, 5/6 unlabelled precision, and 5/7 unlabelled recall.

5. Results and discussion

The results for the three evaluation metrics on Section 23 of CCGbank are given in Table 2. BP is bracketed precision; LP is labelled precision; UP is unlabelled precision. BR, LR, UR are defined similarly for recall. The scores for each evaluation are accumulated over all sentences in the test set, rather than averaged per sentence. We also give the score for accuracy of the lexical categories.

Gold standard
$\langle \mathit{the}, \mathsf{NP/N}_{1}, 1, \mathit{shares}\rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_{1})/(\mathsf{S[dcl]}_{2}\backslash\mathsf{NP}), 1, \mathit{shares}\rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_{1})/(\mathsf{S[dcl]}_{2}\backslash\mathsf{NP}), 2, \mathit{has}\rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash NP_{1})/(\mathsf{S[pt]}_{2}\backslash\mathsf{NP}), 1, \mathit{IBM}\rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash\mathsf{NP}_{1})/(\mathsf{S[pt]}_{2}\backslash\mathsf{NP}), 2, \mathit{bought}\rangle$
$\langle \mathit{bought}, (\mathsf{S[pt]}\backslash\mathsf{NP}_{1})/\mathsf{NP}_{2}, 1, \mathit{IBM}\rangle$
$\langle \mathit{bought}, (\mathsf{S[pt]}\backslash\mathsf{NP}_{1})/\mathsf{NP}_{2}, 2, \mathit{shares}\rangle$
Example tree
$\langle \mathit{the}, \mathsf{NP/N}_{1}, 1, \mathit{shares}\rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_{1})/(\mathsf{S[dcl]}_{2}\backslash\mathsf{NP}), 1, \mathit{shares}\rangle$
$\langle \mathit{that}, (\mathsf{NP}\backslash\mathsf{NP}_{1})/(\mathsf{S[dcl]}_{2}\backslash\mathsf{NP}), 2, \mathit{has}\rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash NP_{1})/\mathsf{NP}_{2}, 1, \mathit{IBM}\rangle$
$\langle \mathit{has}, (\mathsf{S[dcl]}\backslash NP_{1})/\mathsf{NP}_{2}, 2, \mathit{shares}\rangle$
$\langle \mathit{bought}, \mathsf{S[pss]}\backslash NP_{1}, 1, \mathit{shares}\rangle$

Table 1: Dependencies for the trees in Figure 2

<table>
<tr><th colspan="4">Accuracy of lexical categories</th></tr>
<tr><td colspan="4">92.0%</td></tr>
<tr><th colspan="4">Parseval</th></tr>
<tr><td>LP</td><td>LR</td><td>BP</td><td>BR</td></tr>
<tr><td>81.6%</td><td>81.9%</td><td>85.5%</td><td>85.9%</td></tr>
<tr><th colspan="4">Tree dependencies</th></tr>
<tr><td colspan="2">Labelled recall</td><td colspan="2">Unlabelled recall</td></tr>
<tr><td colspan="2">84.0%</td><td colspan="2">90.1%</td></tr>
<tr><th colspan="4">“Deep” dependencies</th></tr>
<tr><td>LP</td><td>LR</td><td>UP</td><td>UR</td></tr>
<tr><td>83.7%</td><td>84.2%</td><td>90.5%</td><td>91.1%</td></tr>
</table>

Table 2: Results for the three evaluation metrics

5.1. The Parseval scores

It is hard to draw conclusions from the Parseval scores because of the difficulty in comparing results across different tree representations. Our figures are below the 88.3%/88.0% labelled precision/recall of Collins (1999). However, a direct comparison of the Parseval result with Penn Treebank parsers is not informative, even for the same set of sentences. Because Penn Treebank trees are very flat, they contain far fewer brackets than CCG derivation trees; hence the rate of crossing brackets (and bracketed precision and recall) will automatically be much lower than for a grammar which produces at most binary-branching trees. The flat trees also mean that Parseval is too lenient towards mis-attachments produced by Penn Treebank parsers (Manning and Schütze, 1999). Furthermore, the set of node labels for Penn Treebank trees and the set of CCG categories are not comparable.

Hockenmaier (2001) notes a further problem with applying Parseval metrics to CCG derivation trees. Consider verb phrases, $(\mathsf{S}\backslash\mathsf{NP})$, which can have left and right modifiers ($(\mathsf{S}\backslash\mathsf{NP})/(\mathsf{S}\backslash\mathsf{NP})$ and $(\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})$) with the following two rule instantiations:

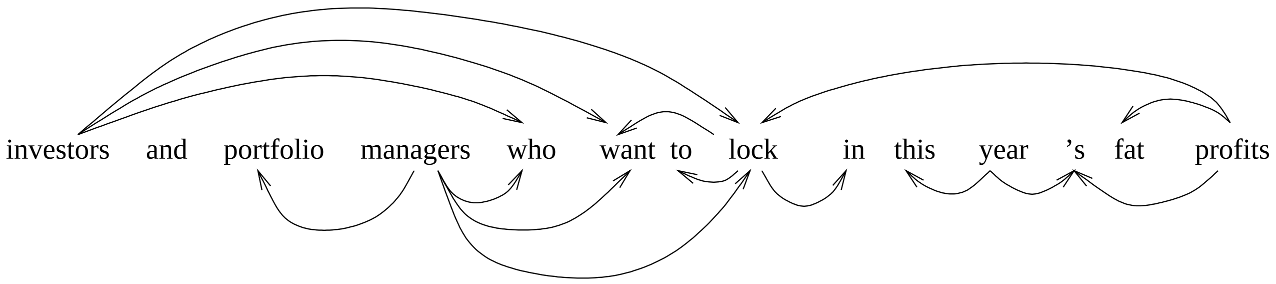
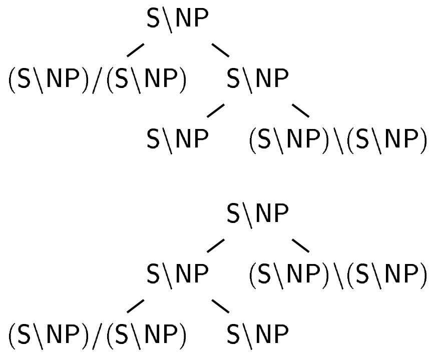


Figure 3: Example dependency structure

(8) $\mathsf{S\backslash NP} \rightarrow \mathsf{(S\backslash NP)/(S\backslash NP)}\ \ \mathsf{S\backslash NP}$
$\quad\ \mathsf{S\backslash NP} \rightarrow \mathsf{S\backslash NP}\ \ \mathsf{(S\backslash NP)\backslash(S\backslash NP)}$

For any parsing model which is defined in terms of (possibly headed or lexicalized) local trees, the following two trees are equivalent:



This is also the case for the normal-form parser described above. A similar problem arises with coordinations/lists involving more than two conjuncts. The dependency evaluations described below do not suffer from this problem because the same dependencies are produced for each derivation.

5.2. Dependency evaluation 1

As expected, the results for the tree dependencies are higher than the Parseval scores. Unlike Parseval, the dependency measure is neutral with respect to the branching factor of the trees produced by the grammar. In particular, for a given sentence, the number of dependencies is identical to the number of words in the sentence. Since this is the same for any parser, unlabelled recovery of dependencies can be used to indicate how parsers based on different grammars compare. Note that our unlabelled figures (90.1% recall) are similar to those of Collins (90.9%).

However, a direct comparison with the labelled figures given by Collins (88.3% recall) is again problematic. First, the sets of labels are very different. In order for labelled dependencies as defined by a CCG derivation tree to be correct, complement-adjunct distinctions as well as extraction cases have to be correctly recovered. Extraction is not indicated in the trees returned by Collins' parser, and can therefore not be evaluated. Mistaking a complement daughter for an adjunct or vice versa has a much greater effect on the labelled scores for CCG than for Penn Treebank parse trees. In Collins' parser, the complement-adjunct distinction is only expressed in the label of the particular node in question. However, in CCG this can affect the entire tree below the parent – both the subtree underneath the head daughter and the subtree underneath the non-head daughter.

In addition, Collins performs the following preprocessing steps on the output of his parser and the Gold standard: all POS tags are replaced by a single token "TAG". All complement markings on the parent and head node are removed, so that one attachment decision made higher up in the tree does not affect the evaluation of its daughter. We cannot readily perform the same preprocessing steps: the choice of lexical categories can affect the tree at several levels, not just at the leaf nodes; furthermore, complement-adjunct distinctions are also encoded in all intermediate categories, not just a constituent's maximal projection.

5.3. Dependency evaluation 2

One of the advantages of a dependency-style evaluation is that the scores can be broken down by relation, as shown in Table 4, which gives scores for some of the most frequent types.[1] The table also gives some indication of the kinds of relations used in the evaluation.

The relations are defined in terms of CCG categories, which raises the question of how these compare with a more generic set such as that proposed by Carroll et al. (1998). First note that there are many more relations in our scheme: around 700 in total compared with 20 for Carroll et al. We have so many relations because each argument slot in each category (of which there are 398) encodes a separate relation.

Clearly there is room for generalisation in our scheme. For example, Carroll et al. have one relation for subjects, whereas we have a different relation for each category type encoding a subject. Examples of two categories encoding subject relations are $\mathsf{(S[dcl]\backslash NP_{\mathit{1}})/NP_{\mathit{2}}}$ and $\mathsf{(S[b]\backslash NP_{\mathit{1}})/NP_{\mathit{2}}}$. In future work we will investigate mapping our relations onto Carroll et al.'s.

One potential weakness of our evaluation (which follows from encoding all argument slots as relations) is that some relations are effectively counted more than once. For

[1] #ref is the number of dependencies with the given relation type in the gold standard; #test is the number of dependencies with the given relation type produced by the parser; LP/LR are labelled precision/recall; and the F-score is calculated as (2*LP*LR)/(LP+LR).

$\langle P,H,S\rangle$	#ref	#test	LP%	LR%
$\langle NP,NP,NP\backslash NP\rangle$	3,765	3,626	75.2	72.4
$\langle HEAD\rangle$	2371	2367	94.5	94.4
$\langle NP,NP,NP[conj]\rangle$	935	1,075	61.3	70.5
$\langle S[dcl]\backslash NP,S[dcl]\backslash NP,(S\backslash NP)\backslash(S\backslash NP)\rangle$	914	905	60.9	60.3
$\langle S[dcl]\backslash NP,(S[dcl]\backslash NP)/NP,NP\rangle$	880	858	86.7	84.6
$\langle S[pss]\backslash NP,S[pss]\backslash NP,(S\backslash NP)\backslash(S\backslash NP)\rangle$	442	470	70.6	75.1

Table 3: Some dependency relations in evaluation 1

Functor	Slot	Category description	LP %	# test	LR %	# ref	F-score
$N_X/N_{X,1}$	1	*nominal modifier*	94.4	7,856	93.2	7,955	93.8
$NP_X/N_{X,1}$	1	*determiner*	96.7	4,548	96.4	4,566	96.5
$(NP_X\backslash NP_{X,1})/NP_2$	2	*np modifying preposition*	82.1	2,659	81.2	2,690	81.6
$(NP_X\backslash NP_{X,1})/NP_2$	1	*np modifying preposition*	76.0	2,449	76.2	2,443	76.1
$(S_X\backslash NP_Y)\backslash(S_{X,1}\backslash NP_Y)/NP_2$	2	*vp modifying preposition*	68.7	1,327	66.1	1,379	67.4
$(S_X\backslash NP_Y)\backslash(S_{X,1}\backslash NP_Y)/NP_2$	1	*vp modifying preposition*	66.2	1,247	65.0	1,271	65.6
$(S[dcl]\backslash NP_1)/NP_2)$	1	*transitive verb*	83.2	885	82.0	898	82.6
$(S[dcl]\backslash NP_1)/NP_2)$	2	*transitive verb*	80.3	885	78.4	907	79.3
$(S_X\backslash NP_Y)\backslash(S_{X,1}\backslash NP_Y)$	1	*adverbial modifier*	81.5	961	82.2	953	81.8
(PP/NP_1)	1	*preposition complement*	61.5	993	75.7	807	67.9
$(S[b]\backslash NP_1)/NP_2)$	2	*infinitival transitive verb*	86.6	719	85.2	731	85.9
$(S[dcl]\backslash NP_{X,1})/(S[b]_2\backslash NP_X)$	2	*auxiliary*	97.6	631	98.6	625	98.1
$(S[dcl]\backslash NP_{X,1})/(S[b]_2\backslash NP_X)$	1	*auxiliary*	92.2	638	95.0	619	93.6
$(S[b]\backslash NP_1)/NP_2$	1	*infinitival transitive verb*	80.6	566	83.1	549	81.8
$(NP_X/N_{X,1})\backslash NP_2$	1	*s genitive*	96.6	472	95.2	479	95.9
$(NP_X/N_{X,1})\backslash NP_2$	2	*s genitive*	92.5	482	95.3	468	93.9
$(S[dcl]\backslash NP_1)/S[dcl]_2$	1	*sentential complement verb*	93.0	431	95.5	420	94.2
$(NP_X\backslash NP_{X,1})/(S[dcl]_2\backslash NP_X)$	1	*subject relative pronoun*	71.9	295	72.6	292	72.2
$(NP_X\backslash NP_{X,1})/(S[dcl]_2\backslash NP_X)$	2	*subject relative pronoun*	94.5	289	95.5	286	95.0

Table 4: Results for dependency evaluation 2 by relation; only a subset of the relations are shown

example, in the sentence *John has been eating beans*, *John* is evaluated as a subject three times: as the subject of *has*, *been* and *eating*. But if the subject of *eating* is correct in this example, then the subjects of the auxiliary verbs will be correct as well.

We would also like to make a distinction between arguments that have been extracted from a predicate, and those that are "in situ". Currently the direct object of a verb, for example, is the same relation whether it has been extracted or not. It would be useful to at least have the option to make this distinction.

5.4. Comparing the dependency evaluations

The dependencies expressed in dependency evaluation 1 are not simply a subset of the relations used in the second dependency evaluation. When the relations are broken down individually, this leads to an interesting comparison.

In dependency evaluation 2, it is possible to determine how well nominal prepositions have been recovered, whereas in dependency evaluation 1, we can only evaluate how well NP postmodifiers have been recovered.

In contrast to dependency evaluation 2, dependency evaluation 1 includes a separate relation for the head of a sentence $\langle HEAD\rangle$ (assuming a single head for each sentence, including coordinate structures).

In dependency evaluation 1, it can be seen for each type of constituent whether coordination is recovered properly, e.g. $\langle NP,NP,NP[conj]\rangle$. In dependency evaluation 2, coordination relations are not represented explicitly.

Some relations in dependency evaluation 1 (like the direct object of transitive declaratives, $\langle S[dcl]\backslash NP,(S[dcl]\backslash NP)/NP,NP\rangle$) seem to be the same as in evaluation 2. However, in dependency evaluation 1 only non-extracted cases are taken into account.

6. Conclusion

We have presented three evaluations for a wide-coverage CCG parser. Of these, Parseval seems the least appropriate, especially if a comparison is to be made with existing Penn Treebank parsers. In an attempt to compare with the Collins parser, we adopted a dependency evaluation in which dependencies are defined in terms of local trees; however, the different labelling used in the CCG derivation tree compared to the Penn Treebank made the comparison of labelled dependencies problematic. The comparison of unlabelled dependencies was more appropriate, however.

One of the features of CCG is its analysis of long-range dependencies. In an attempt to incorporate such dependencies into the evaluation, we proposed a second dependency evaluation, in which the dependency relations are defined in terms of the CCG categories. This is closer to evaluations based on grammatical relations, although if a comparison is to be made with parsers using such an evaluation, a mapping is required between the CCG dependencies and the set of grammatical relations.

7. Acknowledgements

This research was funded by EPSRC grant GR/M96889/01 and an EPSRC studentship to the second author. We would like to thank Mark Steedman for his guidance and expert help with this work.

8. References

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st LREC Conference*, pages 447–454, Granada, Spain.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL (to appear)*, Philadelphia, PA.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the ACL*, pages 16–23, Madrid, Spain.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (to appear)*, Las Palmas, Spain.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL (to appear)*, Philadelphia, PA.

Julia Hockenmaier. 2001. Statistical parsing for CCG with simple generative models. In *Proceedings of Student Research Workshop, 39th Meeting of the ACL*, Toulose, France.

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

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, MA.

Mitchell Marcus, Beatrice Santorini, and Mary Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

A Comparison of Evaluation Metrics for a Broad-Coverage Stochastic Parser

Richard Crouch, Ronald M. Kaplan, Tracy H. King, Stefan Riezler

Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA, 94025
{crouch|kaplan|thking|riezler}@parc.com

Abstract

This paper reports on the use of two distinct evaluation metrics for assessing a stochastic parsing model consisting of a broad-coverage Lexical-Functional Grammar (LFG), an efficient constraint-based parser and a stochastic disambiguation model. The first evaluation metric measures matches of predicate-argument relations in LFG f-structures (henceforth the LFG annotation scheme) to a gold standard of manually annotated f-structures for a subset of the UPenn Wall Street Journal treebank. The other metric maps predicate-argument relations in LFG f-structures to dependency relations (henceforth DR annotations) as proposed by Carroll et al. (Carroll et al., 1999). For evaluation, these relations are matched against Carroll et al.'s gold standard which was manually annnotated on a subset of the Brown corpus. The parser plus stochastic disambiguator gives an F-measure of 79% (LFG) or 73% (DR) on the WSJ test set. This shows that the two evaluation schemes are similar in spirit, although accuracy is impaired systematically by mapping one annotation scheme to the other. A systematic loss of accuracy is incurred also by corpus variation: Training the stochastic disambiguation model on WSJ data and testing on Carroll et al.'s Brown corpus data yields an F-score of 74% (DR) for dependency-relation match. A variant of this measure comparable to the measure reported by Carroll et al. yields an F-measure of 76%. We examine divergences between annotation schemes aiming at a future improvement of methods for assessing parser quality.

1. Introduction

Recent years have seen increased interest in parsing systems that capture predicate-argument relations instead of mere phrase-structure representations. In aiming for this goal, considerable progress has been made by combining systems of hand-coded, linguistically fine-grained grammars with robustness techniques and stochastic disambiguation models. However, it can reasonably be argued that the standard evaluation procedure for stochastic parsing—precision and recall of matching labeled bracketing to section 23 of the UPenn Wall Street Journal (WSJ) treebank (Marcus et al., 1994)—is not appropriate for assessing the quality of parsers on matching predicate-argument relations. A new standard for evaluation on predicate-argument relations and for annotating a gold standard is needed.

In this paper we present a stochastic parsing model consisting of a broad-coverage Lexical-Functional Grammar (LFG), a constraint-based parser and a stochastic disambiguation model, and discuss the evaluation of this system on two distinct evaluation metrics for assessing the quality of the stochastic parsing model on matching predicate-argument relations. The first evaluation metric measures matches of predicate-argument relations in LFG f-structures (henceforth the LFG annotation scheme) to a gold standard of manually annotated f-structures for a representative subset of the WSJ treebank. The evaluation measure counts the number of predicate-argument relations in the f-structure of the parse selected by the stochastic model that match those in the gold standard annotation.

The other metric we employed maps predicate-argument relations in LFG f-structures to the dependency relations (henceforth the DR annotation scheme) proposed by Carroll et al. (Carroll et al., 1999). Evaluation with this metric measures the matches of these relations to Carroll et al.'s gold standard corpus.

Our parser plus stochastic disambiguator gives an F-measure of 79% (LFG) or 73% (DR) on the WSJ test set, showing that the two evaluation schemes are similar in spirit. However, accuracy is systematically impaired by mapping one annotation scheme to the other. A systematic loss of accuracy is incurred also by corpus variation: Training the stochastic disambiguation model on WSJ data and testing on Carroll et al.'s Brown corpus data gives a DR F-measure of 74% for matching dependency relations. For a direct comparison of our results with Carroll et al.'s system, we also computed an F-measure that does not distinguish different types of dependency relations. Under this measure we obtain 76% F-measure.

One goal of this paper is to highlight possible pitfalls and error sources in translating between different annotation schemes and gold standards. We believe that a thorough investigation of divergences in annotation schemes will facilitate a future standard for predicate-argument evaluation and annotation.

This paper is organized as follows. After introducing the grammar and parser used in this experiment, we describe in section 2. the robustness techniques employed to reach 100% grammar coverage on unseen WSJ text (in the sense of the proportion of sentences for which at least one analysis is found). Furthermore, we give in section 3. a short account of the stochastic model used for disambiguating LFG parses. Experiments on evaluating the combined system of parser and stochastic disambiguator on the two distinct evaluation measures and corpora are described in section 4.

2. Robust Parsing using LFG

2.1. A Broad-Coverage Lexical-Functional Grammar

The grammar used for this project has been developed in the ParGram project (Butt et al., 1999). It uses LFG as a formalism, producing c(onstituent)-structures (trees) and f(unctional)-structures (attribute value matrices) as output. The c-structures encode constituency. Each c-structure has at least one corresponding f-structure. F-structures encode predicate-argument relations and other grammatical information, e.g., number, tense. The XLE parser (Maxwell and Kaplan, 1993) was used to produce packed representations, specifying all possible grammar analyses of the input.

The grammar has 314 rules with regular expression right-hand sides which compile into a collection of finite-state machines with a total of 8,759 states and 19,695 arcs. The grammar uses several lexicons and two guessers: one guesser for words recognized by the morphological analyzer but not in the other lexicons and one for those not recognized. As such, most common and proper nouns, adjectives, and adverbs have no explicit lexical entry. The main verb lexicon contains 9,652 verb stems and 23,525 subcategorization frame-verb stem entries; there are also lexicons for adjectives and nouns with subcategorization frames and for closed class items such as prepositions.

For estimation and testing purposes using the WSJ treebank, the grammar was modified to parse part of speech tags and labeled bracketing. A stripped down version of the WSJ treebank was created that used only those POS tags and labeled brackets relevant and reliable for determining grammatical relations. The WSJ labels are given entries in a special LFG lexicon, and these entries constrain both the c-structure and the f-structure of the parse. For example, the WSJ's ADJP-PRD label must correspond to an AP in the c-structure and an XCOMP in the f-structure. In this version of the corpus, all WSJ labels with -SBJ are retained and are restricted to phrases corresponding to SUBJ in the LFG grammar; in addition, it contains NP under VP (OBJ and OBJth in the LFG grammar), all -LGS tags (OBL-AG), all -PRD tags (XCOMP), VP under VP (XCOMP), SBAR- (COMP), and verb POS tags under VP (V in the c-structure). For example, our labeled bracketing version of wsj_1305.mrg is *[NP-SBJ His credibility] is/VBZ_ also [PP-PRD on the line] in the investment community.*

Some mismatches between the WSJ labeled bracketing and the LFG grammar remain. These often arise when a given constituent fills a grammatical role in more than one clause, usually when it is a SUBJ or OBJ in one clause and also the SUBJ of an XCOMP complement. For example, in wsj_1303.mrg *Japan's Daiwa Securities Co. named Masahiro Dozen president.*, the noun phrase *Masahiro Dozen* is labeled as an NP-SBJ, presumably because it is the subject of a small clause complement. However, the LFG grammar treats it also as the OBJ of the matrix clause. As a result, the labeled bracketed version of this sentence does not receive a full parse, even though the LFG output from parsing its unlabeled, string-only counterpart is well-formed. Some other bracketing mismatches remain between this stripped down WSJ corpus and the LFG grammar; these are usually the result of adjunct attachment. Such mismatches occur in part because, besides minor modifications to match the bracketing for special constructions, e.g., negated infinitives, the grammar was not altered to mirror the WSJ bracketing.

2.2. Robustness Techniques

To increase robustness, the standard grammar has been augmented with a FRAGMENT grammar. This grammar parses the sentence as well-formed chunks specified by the grammar, in particular as Ss, NPs, PPs, and VPs. These chunks have both c-structures and f-structures corresponding to them, just as in the standard grammar. Any substring that cannot be parsed as one of these chunks is parsed as a TOKEN chunk. The TOKENs are also recorded in the c- and f-structures. The grammar has a fewest-chunk method for determining the correct parse. For example, if a string can be parsed as two NPs and a VP or as one NP and an S, the NP-S option is chosen.

A final capability of XLE that increases coverage of the standard plus fragment grammar on the WSJ corpus is a SKIMMING technique. Skimming is used to avoid timeouts and memory problems when parsing unusually difficult sentences in the corpus. When the amount of time or memory spent on a sentence exceeds a threshhold, XLE goes into skimming mode for the constituents whose processing has not been completed. When XLE skims these remaining constituents, it does a bounded amount of work per subtree. This guarantees that XLE finishes processing a sentence in a polynomial amount of time, although it does not necessarily return the complete set of analyses. In parsing section 23, 7.2% of the sentences were skimmed; 26.1% of the skimmed sentences resulted in full parses, while 73.9% were fragment parses.

The final grammar coverage achieved 100% of section 23 as unseen unlabeled data: 74.7% of those were full parses, 25.3% FRAGMENT and/or SKIMMED parses.

3. Discriminative Statistical Estimation from Partially Labeled Data

3.1. Exponential Probability Models on LFG Parses

The probability model we employed for stochastic disambiguation is the well-known family of exponential models. These models have already been applied successfully for disambiguation of various constraint-based grammars (LFG (Johnson et al., 1999), HPSG (Bouma et al., 2000), DCG (Osborne, 2000)).

In this paper we are concerned with conditional exponential models of the form:

$$p_{\boldsymbol{\lambda}}(x|y) = Z_{\boldsymbol{\lambda}}(y)^{-1} e^{\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)}$$

where $X(y)$ is the set of parses for sentence y, $Z_{\boldsymbol{\lambda}}(y) = \sum_{x \in X(y)} e^{\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)}$ is a normalizing constant, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^n$ is a vector of log-parameters, $\boldsymbol{f} = (f_1, \ldots, f_n)$ is a vector of property-functions $f_i : \mathcal{X} \rightarrow \mathbb{R}$ for $i = 1, \ldots, n$ on the set of parses $\mathcal{X}$, and $\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)$ is the vector dot product $\sum_{i=1}^{n} \lambda_i f_i(x)$.

In our experiments, we employed around 1000 complex property-functions comprising information about c-structure, f-structure, and lexical elements in parses, similar to the properties used in Johnson et al. (1999). For example, there are property functions for c-structure nodes and c-structure subtrees, indicating attachment preferences. High versus low attachment is indicated by property functions counting the number of recursively embedded phrases.

Other property functions are designed to refer to f-structure attributes, corresponding to grammatical functions in LFG, or to atomic attribute-value pairs in f-structures. More complex property functions are designed to indicate, for example, the branching behaviour of c-structures and the (non)-parallelism of coordinations on both c-structure and f-structure levels. Furthermore, properties refering to lexical elements based on an auxiliary distribution approach as presented in Riezler et al. (2000) are included in the model. Here tuples of head words, argument words, and grammatical relations are extracted from the training sections of the WSJ, and fed into a finite mixture model for clustering grammatical relations. The clustering model itself is then used to yield smoothed probabilities as values for property functions on head-argument-relation tuples of LFG parses.

3.2. Discriminative Estimation

Discriminative estimation techniques have recently received great attention in the statistical machine learning community and have already been applied to statistical parsing (Johnson et al., 1999; Collins, 2000; Collins and Duffy, 2001). In discriminative estimation, only the conditional relation of an analysis given an example is considered relevant, whereas in maximum likelihood estimation the joint probability of the training data to best describe observations is maximized. Since the discriminative task is directly kept in mind during estimation, discriminative methods can yield improved performance. In our case, discriminative criteria cannot be defined directly with respect to "correct labels" or "gold standard" parses since the WSJ annotations are not sufficient to disambiguate the more complex LFG parses. However, instead of retreating to unsupervised estimation techniques or creating small LFG treebanks by hand, we use the labeled bracketing of the WSJ training sections to guide discriminative estimation. That is, discriminative criteria are defined with respect to the *set of parses consistent with the WSJ annotations*[1].

The objective function in our approach, denoted by $P(\boldsymbol{\lambda})$, is the joint of the negative log-likelihood $-L(\boldsymbol{\lambda})$ and a Gaussian regularization term $-G(\boldsymbol{\lambda})$ on the parameters $\boldsymbol{\lambda}$. Let $\{(y_j, z_j)\}_{j=1}^m$ be a set of training data, consisting of pairs of sentences y and partial annotations z, let $X(y, z)$ be the set of parses for sentence y consistent with annotation z, and $X(y)$ be the set of all parses produced by the grammar for sentence y. Furthermore, let $p[f]$ denote the expectation of function f under distribution p. Then $P(\boldsymbol{\lambda})$ can be defined for a conditional exponential model $p_{\boldsymbol{\lambda}}(z|y)$ as:

$$\begin{aligned}
P(\boldsymbol{\lambda}) &= -L(\boldsymbol{\lambda}) - G(\boldsymbol{\lambda}) \\
&= -\log \prod_{j=1}^{m} p_{\boldsymbol{\lambda}}(z_j|y_j) + \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \\
&= -\sum_{j=1}^{m} \log \frac{\sum_{X(y_j,z_j)} e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)}}{\sum_{X(y_j)} e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)}} + \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \\
&= -\sum_{j=1}^{m} \log \sum_{X(y_j,z_j)} e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)} \\
&\quad + \sum_{j=1}^{m} \log \sum_{X(y_j)} e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)} + \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2}.
\end{aligned}$$

Intuitively, the goal of estimation is to find model parameters which make the two expectations in the last equation equal, i.e. which adjust the model parameters to put all the weight on the parses consistent with the partial annotation, modulo a penalty term from the Gaussian prior for too large or too small weights.

Since a closed form solution for such parameters is not available, numerical optimization methods have to be used. In our experiments, we adapted a conjugate gradient routine to our task (see Press (1992)), yielding a fast converging optimization algorithm where at each iteration the negative log-likelihood $P(\boldsymbol{\lambda})$ and the gradient vector have to be evaluated.[2]. For our task the gradient takes the form:

$$\nabla P(\boldsymbol{\lambda}) = \left\langle \frac{\partial P(\boldsymbol{\lambda})}{\partial \lambda_1}, \frac{\partial P(\boldsymbol{\lambda})}{\partial \lambda_2}, \ldots, \frac{\partial P(\boldsymbol{\lambda})}{\partial \lambda_n} \right\rangle, \text{ and}$$

$$\begin{aligned}
\frac{\partial P(\boldsymbol{\lambda})}{\partial \lambda_i} &= -\sum_{j=1}^{m} (\sum_{x \in X(y_j,z_j)} \frac{e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)} f_i(x)}{\sum_{x \in X(y_j,z_j)} e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)}} \\
&\quad - \sum_{x \in X(y_j)} \frac{e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)} f_i(x)}{\sum_{x \in X(y_j)} e^{\boldsymbol{\lambda}\cdot \boldsymbol{f}(x)}}) + \frac{\lambda_i}{\sigma_i^2}.
\end{aligned}$$

The derivatives in the gradient vector intuitively are again just a difference of two expectations

$$-\sum_{j=1}^{m} p_{\boldsymbol{\lambda}}[f_i|y_j, z_j] + \sum_{j=1}^{m} p_{\boldsymbol{\lambda}}[f_i|y_j] + \frac{\lambda_i}{\sigma_i^2}.$$

Note also that this expression shares many common terms with the likelihood function, suggesting an efficient implementation of the optimization routine.

4. Experimental Evaluation

Training: The basic training data for our experiments are sections 02-21 of the WSJ treebank. As a first step, all sections were parsed, and the packed parse forests unpacked and stored. For discriminative estimation, this data set was restricted to sentences which receive a full parse (in contrast to a FRAGMENT or SKIMMED parse) for both its partially labeled and its unlabeled variant. Furthermore, only sentences which received at most 1,000 parses were

[1] An earlier approach using partially labeled data for estimating stochastics parsers is Pereira and Schabes (1992) work on training PCFG from partially bracketed data. Their approach differs from the one we use here in that Pereira and Schabes take an EM-based approach maximizing the joint likelihood of the parses and strings of their training data, while we maximize the conditional likelihood of the sets of parses given the corresponding strings in a discriminative estimation setting.

[2] An alternative numerical method would be a combination of iterative scaling techniques with a conditional EM algorithm (Jebara and Pentland, 1998) However, it has been shown experimentally that conjugate gradient techniques can outperform iterative scaling techniques by far in running time (Minka, 2001).

taken under consideration. From this set, sentences from which a discriminative learner cannot possibly take advantage, i.e. sentences where the set of parses assigned to the partially labeled string was not a proper subset of the parses assigned the unlabeled string, were removed. These successive selection steps resulted in a final training set consisting of 10,000 sentence each with parses for partially labeled and unlabeled versions. Altogether there were 150,000 parses for partially labeled input and 500,000 for unlabeled input.

For estimation, a simple property selection procedure was applied to the full set of around 1000 properties. This procedure is based on a frequency cutoff on instantiations of properties for the parses in the labeled training set. The result of this procedure is a reduction of the property vector to about half of its size. Furthermore, a held-out data set was created from section 24 of the WSJ treebank for experimental selection of the variance parameter of the prior distribution. This set consists of 150 sentences which received only full parses, out of which the most plausible one was selected by manual inspection.

Testing: Two different sets of test data were used: (i) 700 sentences randomly extracted from section 23 of the WSJ treebank and given gold-standard f-structure annotations according to our LFG scheme, and (ii) 500 sentences from the Brown corpus given gold standard annotations by Carroll et al. (1999) according to their dependency relations (DR) scheme[3]. Both the LFG and DR annotation schemes are discussed in more detail below, as is a mapping from LFG f-structures to DR annotations.

Gold standard annotation of the WSJ test set was bootstrapped by parsing the test sentences using the LFG grammar and also checking for consistency with the Penn Treebank annotation. Starting from the (sometimes fragmentary) parser analyses and the Treebank annotations, gold standard parses were created by manual corrections and extensions of the LFG parses. Manual corrections were necessary in about half of the cases.

Performance on the LFG-annotated WSJ test set was measured using both the LFG and DR metrics, thanks to the LFG-to-DR annotation mapping. Performance on the DR-annotated Brown test set was only measured using the DR metric, owing to the absence of an inverse map from DR to LFG annotations.

Results: In our evaluation we report F-measures for the respective types of annotation, LFG or DR, and for three types of parse selection, (i) *lower bound*: random choice of a parse from the set of analyses, (ii) *upper bound*: selection of the parse with the best F-measure according to the annotation scheme used, and (iii) *stochastic*: the parse selected by the stochastic disambiguator. The *error reduction* row lists the reduction in error rate relative to the upper and lower bounds obtained by the stochastic disambiguation model. F-measures is defined as $2 \times precision \times recall / (precision + recall)$.

Table 1 gives results for 700 examples randomly selected from section 23 of the WSJ treebank, using both LFG and DR measures. The effect of the quality of the parses on disambiguation performance can be illustrated by breaking down the F-measures according to whether the parser yields full parses or FRAGMENT or SKIMMED parses or both for the test sentences. The percentages of test examples which belong to the respective classes of quality are listed in the first row of Table 2. F-measures broken down according to classes of parse quality are recorded in the following rows. The first column shows F-measures for all parses in the test set, as in Table 1, the second column shows best F-measures when restricting attention to examples which receive only full parses. The third column reports F-measurs for examples which receive only non-full parses, i.e., FRAGMENT or SKIMMED parses or SKIMMED FRAGMENT parses. Columns 4–6 break down non-full parses according to examples which receive only FRAGMENT, only SKIMMED, or only SKIMMED FRAGMENT parses. Since most results on predicate-argument matching have been reported for length-restricted test sets (20–30 words), we also provide for comparison results for a subset of 500 sentences in our sample which had less than 25 words. These results are reported in Table 3.

Table 1: Disambiguation results for 700 examples randomly selected from section 23 of the WSJ treebank using LFG and DR measures.

	LFG	DR
upper bound	84.7	80.7
stochastic	78.7	72.9
lower bound	75.0	68.8
error reduction	38	35

Table 3: Disambiguation results on 500 examples restricted to $<$ 25 words randomly selected from section 23 of the WSJ treebank using LFG and DR measures.

	LFG	DR
upper bound	88.0	85.4
stochastic	82.8	77.5
lower bound	78.0	72.6
error reduction	42	38

Results of the evaluation on Carroll et al.'s Brown test set are given in Tables 4 and 5. Table 4 presents an analysis of evaluation results according to parse-quality for the DR measure applied to the Brown corpus test set. In Table 5 we show the DR measure along with an evaluation measure which facilitates a direct comparison of our results to those of Carroll et al. (1999). Following Carroll et al. (1999) we count a depedency relation as correct if the gold standard has a relation with the same governor and dependent but perhaps with a different relation-type. This dependency-only (DO) measure thus does not reflect mismatches be-

[3] Both corpora are available online. The WSJ f-structure bank at www.parc.com/istl/groups/nltt/fsbank/, and Carroll et al.'s corpus at www.cogs.susx.ac.uk/lab/nlp/carroll/greval.html.

Table 2: LFG F-measures broken down according to parse quality for the 700 WSJ test examples.

	all	full	non-full	fragments	skimmed	skimmed fragments
% of test set	100	74.7	25.3	20.4	1.4	3.4
upper bound	84.7	91.3	69.8	72.0	73.1	60.5
stochastic	78.8	84.6	65.2	67.4	67.8	55.9
lower bound	75.0	80.1	63.9	65.9	66.2	55.3

Table 4: DR F-measures broken down according to parse quality for the 500 Brown test examples.

	all	full	non-full	fragments	skimmed	skimmed fragments
% of test set	100	79.6	20.4	20.0	2.0	1.6
upper bound	79.6	84.0	65.2	65.2	55.5	52.9
stochastic	73.7	77.6	61.1	61.0	52.3	49.4
lower bound	70.8	74.4	58.8	58.7	50.8	48.3

tween arguments and modifiers in a small number of cases.

Table 5: Disambiguation results on 500 Brown corpus examples using DO measure and DR measures.

	DO	DR
upper bound	81.6	79.6
stochastic	75.8	73.7
lower bound	72.9	70.8
error reduction	33	34

5. Comparison of Evaluation Metrics

Tables 1 and 3 point to systematically lower F-scores under the DR measure than under the LFG measure, though both indicate similar reductions in error rate due to stochastic disambiguation.

5.1. LFG Evaluation Metric

The LFG evaluation metric is based on the comparison of 'preds-only' f-structures. A preds-only f-structure is a subset of a full f-structure that strips out grammatical attributes (e.g. tense, case, number) that are not directly relevant to predicate-argument structure. More precisely, a preds-only f-structure removes all paths through the f-structure that do not end in a PRED attribute. Figures 1 and 2 illustrate the difference between the full and preds-only f-structures for one parse of the sentence *Meridian will pay a premium of $30.5 million to assume a deposit of $2 billion.* As this example shows, the preds-only f-structure lacks some semantically important information present in the full f-structure, e.g. the marking of future tense, the marking of a purpose clause, and the attribute showing that *a deposit* is an indefinite.

Figure 2 also shows the set of individual feature specifications that define the preds-only f-structure. The first property indicates that the f-structure denoted by `n0` has the semantic form `sf(pay,i15,[n5,n3],[])` as the value of its PRED attribute. `pay` is the predicate, `i15` is a lexical id, `[n5,n3]` a list of f-structure nodes serving as thematic arguments, and `[]` an (empty) list of non-thematic arguments. The grammatical roles associated with thematic and non-thematic arguments are identified by the corresponding `subj`, `obj`, etc., predicates. In this experiment, we measured precision and recall by matching at the granularity of these individual features.

The matching algorithm attempts to find the maximum number of features that can be matched between two structures. It proceeds in a stratified manner, first maximizing the matches between attributes like `pred`, `adjunct` and `in_set`, and then maximizing the matches of any remaining attributes.

5.2. Comparison with DR Metric

As a brief review (see Carroll et al. (1999) for more detail), the DR annotation for our example sentence (obtained via the mapping described below) is

(aux _ pay will)
(detmod _ premium a)
(mod _ $ million)
(dobj pay premium _)
(mod _ $ billion)
(dobj assume $ _)
(subj pay Meridian _)
(mod _ million 30.5)
(mod of premium $)
(mod _ billion 2)
(mod in $ deposit)
(mod to pay assume)

Some obvious points of comparison with the f-structure features are: (i) The DR annotation encodes some information, e.g. the 'detmod' relation, that is not encoded in preds-only f-structures (though it is encoded in full f-structures). (ii) Different occurrences of the same word (e.g. "$") are distinguished via different lexical ids in the LFG representation but not in the DR annotations so that correctly matching DR relations can be problematic. (iii) The DR annotation has 12 relations instead of the 34 feature-specifications. This is because a given predicate-argument relation in the f-structure is broken down into several different feature-specifications. For example, the DR 'mod' relation involves an f-structure path through an ADJUNCT, IN_SET and two PRED attributes; the DR 'subj' relation is a combination of an f-structure PRED and SUBJ attribute. Thus the LFG metric is more sensitive to fine-grained aspects of predicate-

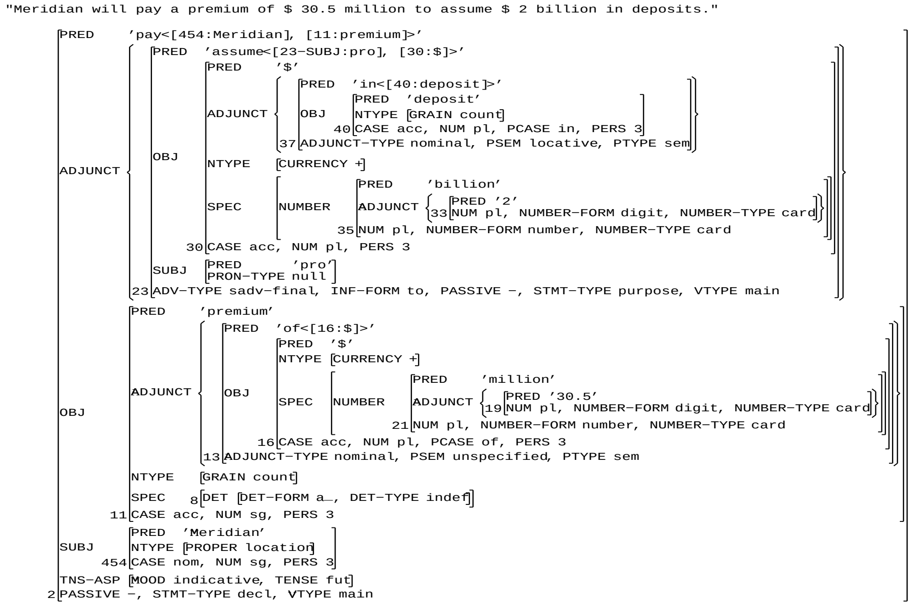
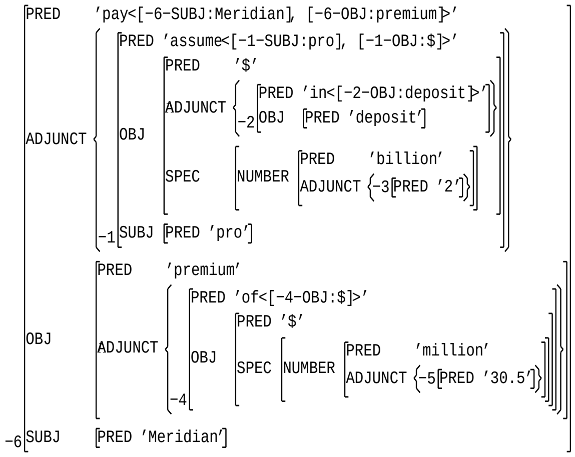

"Meridian will pay a premium of $ 30.5 million to assume $ 2 billion in deposits."
PRED 'pay<[454:Meridian], [11:premium]>'
ADJUNCT
PRED 'assume<[23-SUBJ:pro], [30:$]>'
OBJ
PRED '$'
ADJUNCT
PRED 'in<[40:deposit]>'
OBJ
PRED 'deposit'
NTYPE [GRAIN count]
40 CASE acc, NUM pl, PCASE in, PERS 3
37 ADJUNCT-TYPE nominal, PSEM locative, PTYPE sem
NTYPE [CURRENCY +]
SPEC NUMBER
PRED 'billion'
ADJUNCT
PRED '2'
33 NUM pl, NUMBER-FORM digit, NUMBER-TYPE card
35 NUM pl, NUMBER-FORM number, NUMBER-TYPE card
30 CASE acc, NUM pl, PERS 3
SUBJ
PRED 'pro'
PRON-TYPE null
23 ADV-TYPE sadv-final, INF-FORM to, PASSIVE -, STMT-TYPE purpose, VTYPE main
OBJ
PRED 'premium'
ADJUNCT
PRED 'of<[16:$]>'
OBJ
PRED '$'
NTYPE [CURRENCY +]
SPEC NUMBER
PRED 'million'
ADJUNCT
PRED '30.5'
19 NUM pl, NUMBER-FORM digit, NUMBER-TYPE card
21 NUM pl, NUMBER-FORM number, NUMBER-TYPE card
16 CASE acc, NUM pl, PCASE of, PERS 3
13 ADJUNCT-TYPE nominal, PSEM unspecified, PTYPE sem
NTYPE [GRAIN count]
SPEC 8 [DET [DET-FORM a_, DET-TYPE indef]]
11 CASE acc, NUM sg, PERS 3
SUBJ
PRED 'Meridian'
NTYPE [PROPER location]
454 CASE nom, NUM sg, PERS 3
TNS-ASP [MOOD indicative, TENSE fut]
2 PASSIVE -, STMT-TYPE decl, VTYPE main

Figure 1: Full f-structure



pred(n0,sf(pay,i15,[n5,n3],[])) pred(n5,sf(Meridian,il4,[],[]))
pred(n3,sf(premium,i18,[],[])) pred(n19,sf('2',i70,[],[]))
pred(n28,sf('30.5',i26,[],[])) pred(n7,sf(assume,i64,[n8,n9],[]))
pred(n8,sf(pro,i107,[],[])) pred(n9,sf($,i67,[],[]))
pred(n17,sf(billion,i71,[],[])) pred(n11,sf(in,i84,[n12],[]))
pred(n12,sf(deposit,i86,[],[])) pred(n4,sf(million,i27,[],[]))
pred(n23,sf(of,i21,[n24],[])) pred(n24,sf($,i23,[],[]))
adjunct(n0,n2) in_set(n7,n2)
adjunct(n9,n14) in_set(n11,n14)
adjunct(n17,n18) in_set(n19,n18)
adjunct(n3,n20) in_set(n23,n20)
adjunct(n4,n31) in_set(n28,n31)
subj(n0,n5) subj(n7,n8)
obj(n0,n3) obj(n7,n9) obj(n11,n12) obj(n23,n24)
number(n16,n17) number(n26,n4) spec(n9,n16) spec(n24,n26)

Figure 2: Preds-only f-structure: graphical & clausal representation as produced by XLE

argument relations. However, it imposes a greater penalty than DR on a modifier that is misattached to something that does not have any other modifiers. The LFG measure counts both an extra ADJUNCT feature and an extra IN_SET feature as mismatches, whereas DR only counts a single mismatched MOD. Conversely, LFG gives more credit for getting the singleton attachments correct. Similarly for argument structure. The LFG metric penalizes getting arguments wrong, counting both a PRED and a grammatical relation mismatch, but conversely gives more credit if the argument structure is exactly right.

5.3. Mapping F-structures to DR Annotations

The DR evaluation metric matches the dependency relations provided by the Carroll et al. gold standard with relations determined from information contained in the LFG representations. This enables us to measure the accuracy of our system with a separately defined predicate-argument-oriented standard and to compare our results to other systems that may use the same metric (at this point, perhaps only the Carroll et al. grammar/parser). The DR metric also enables a cross-validation assessment of the LFG-derived predicate-argument measure.

Carroll and Briscoe provide conveniently downloadable files containing the raw input sentences and the corresponding sets of gold standard dependency relations. We assumed it would be relatively straightforward to run the sentences through our system and extract dependency relations that could be compared to the gold standard. But for reasons that ranged from the ridiculous to the sublime, this turned out to be a surprisingly difficult task. One of the lessons learned from this experiment is that even at the level of abstract dependencies it is still very hard to create a standard that does not incorporate unintended framework-specific idiosyncracies.

One set of problems arose from the way the sentences are recorded in the input file. The 'raw' sentences are not formed as they would appear in natural text. They are provided instead as pre-tokenized strings, with punctuation split off by spaces from surrounding words. Thus commas and periods stand as separate tokens and *I'm* and *clients' guilt* show up as *I 'm* and *clients ' guilt*. This preprocessed format may be helpful for parsing systems that embody this particular set of tokenizing conventions or that learn (a la tree bank grammars) from the data at hand. But our system includes a hand-written finite-state tokenizer that is tightly integrated with our grammar and lexicon, and it is designed to operate on text that conforms to normal typographical conventions. It provides less accurate guesses when text is ill-formed in this way, for example, introducing an ambiguity as to whether the quote in *clients ' guilt* is attached as a genitive marker to the left or as an open quote to the right. Another peculiar and troublesome feature of the raw text is that some non-linguistic elements such as chemical formulas are replaced by the meta-symbol <*formul*>; our tokenizer splits this up at the angle brackets and tries to guess a meaning for the word *formul* surrounded by brackets. Faced with these low-level peculiarities, our first step in the evaluation was to edit the raw text as best we could back into normal English.

The gold standard file presented another set of relatively low-level incompatibilities that resulted in spurious mismatches that were somewhat harder to deal with. First, the input sentences conform to American spelling conventions but the head-words in the gold standard relations use British spelling (*neighbor* is coded as *neighbour*). Second, in the gold standard the head-words are converted to their citation forms (e.g. "*walking*" in the text appears as *walk* in the relations). Generally these match the head-words that are easily read from the LFG f-structures, but there are many discrepancies that had to be tracked down. For example, our f-structures do not convert *should* to *shall*, as the gold standard does, whereas we do convert *himself* to *he* (with a reflexive feature) while the gold standard leaves it as *himself*. We ended up creating by trial-and-error a coercion table for this test set so that we could properly match different manifestations of the same head.

The experiment revealed some higher-level conceptual issues. In LFG it is the f-structure rather than the c-structure that most closely encodes the properties on which a non-tree, dependency-oriented evaluation should be based. So we defined our task to be the construction of a routine for reading dependencies from the f-structure alone. It turns out, however, that the Carroll et al. dependencies encode a mixture of superficial phrase-structure properties in addition to underlying dependencies, and it proved a challenge to recreate all the information relevant to a match from the f-structure alone. For example, our f-structures do not represent the categories (NP, S) of the phrases that correspond to the functions, but the gold standard dependencies make tree-based distinctions between non-clausal (e.g. NP) subjects, clausal (e.g. sentential) subjects, and open-complement (VP) subjects. We avoided this kind of discrepancy by neutralizing these distinctions in the gold standard prior to making any comparisons. As another example, our English grammar decodes English auxiliary sequences into features such as PERFECT, PROGRESSIVE, and PASSIVE while the gold standard provides a set of AUX relations that represent the left-to-right order in which *have* and *be* appeared in the original sentence. To obtain the intuitively correct matches, our mapping routine in effect had to simulate a small part of an English generator that decodes our features into their typical left-to-right ordering. In at least one case we simply gave up—it was too hard to figure out under which conditions there might have been do-support in the original string; instead, we removed the few aux-do relations from the gold standard before comparing.

There were a number of situations where it was difficult to determine exactly the gold standard coding conventions either from the documentation or from the examples in the gold standard file. Some of the confusions were resolved by personal communication with Carroll and Briscoe, leading in some cases to the correction of errors in the standard or to the clarification of principles. We discovered for some phenomena that there were simple differences of opinion of how a relation should be annotated. The corpus contains many parentheticals, for example, whose proper attachment is generally determined by extrasyntactic, discourse-level considerations. The default in the LFG grammar is to associate parentheticals at the clause-level whereas the Carroll-Briscoe gold standard tends to associate them with the constituent immediately to the left—a constituent that we cannot identify from the f-structure alone. As other examples, there are still some mysteries about whether and how unexpressed subjects of open-complements are to be encoded and whether and how the head of a relative clause appears in a within-clause dependency.

With considerable effort we solved most but not all of these cross-representation mapping problems, as attested by the relatively high F-scores we have reported. Our current results probably understate to a certain extent our true degree of matching, but the relative differences between sentences using the DR measure are quite informative. A low F-score is an accurate indication that we did not obtain the correct parse. For F-scores above 90 but below 100 it is often the case that we found exactly the right parse but our mapping routine could not produce all the proper relations.

6. Discussion

The general conclusion to draw from our results is that the two metrics, LFG and DR, show broadly similar behavior, for the upper bounds, for the lower bounds, and for the reduction in error relative to the upper bound brought about by the stochastic model. The correlation between the upper bound F-scores for the LFG and DR measures on the WSJ test set is .89. The lower reduction in error rate relative to the upper bound for DR evaluation on the Brown corpus can be attributed to a corpus effect that has also been observed by Gildea (2001) for training and testing PCFGs on the WSJ and Brown corpora.[4] Breaking down evaluation results according to parse quality shows that irrespective of evaluation measure and corpus around 5% overall per-

[4] Gildea reports a decrease from 86.1%/86.6% recall/precision on labeled bracketing to 80.3%/81% when going from training and testing on the WSJ to training on the WSJ and testing on the Brown corpus.

formance is lost due to non-full parses, i.e. FRAGMENT or SKIMMED parses or both.

While disambiguation performance of around 79% F-score on WSJ data seems promising, from one perspective it only offers a 4% absolute improvement over a lower bound random baseline. We think that the high lower bound measure highlights an important aspect of symbolic constraint-based grammars (in contrast to treebank grammars): the symbolic grammar already significantly restricts/disambiguates the range of possible analyses, giving the disambiguator a much narrower window in which to operate. As such, it is more appropriate to assess the disambiguator in terms of reduction in error rate (38% relative to the upper bound) than in terms of absolute F-score. Both the DR and LFG annotations broadly agree in their measure of error reduction.

Due to the lack of standard evaluation measures and gold standards for predicate-argument matching, a comparison of our results to other stochastic parsing systems is difficult at the moment. To our knowledge so far the only direct point of comparison is the parser of Carroll et al. (1999) which is also evaluated on Carroll et al.'s test corpus. They report an F-measure of 75.1% for a DO evaluation that ignores predicate labels but counts dependencies only. Under this measure, our system of parser and stochastic disambiguator achieves 75.8% F-measure. A further point of comparison is the parsing system presented by Bouma et al. (2000). They report comparable relations on lower bounds and upper bounds for their constraint-based parsing systems. On test corpora of a few hundred sentences of up to 20 words an upper bound of 83.7% F-score and a lower bound of 59% is reported; the best disambiguation models achieves 75% F-score.

7. References

Gosse Bouma, Gertjan von Noord, and Robert Malouf. 2000. Alpino: Wide-coverage computational analysis of Dutch. In *Proceedings of Computational Linguistics in the Netherlands*, Amsterdam, Netherlands.

Miriam Butt, Tracy King, Maria-Eugenia Niño, and Frédérique Segond. 1999. *A Grammar Writer's Cookbook*. Number 95 in CSLI Lecture Notes. CSLI Publications, Stanford, CA.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*, Bergen, Norway.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14(NIPS'01)*, Vancouver.

Michael Collins. 2000. Discriminative reranking for natural language processing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, Stanford, CA.

Dan Gildea. 2001. Corpus variation and parser performance. In *Proceedings of 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Pittsburgh, PA.

Tony Jebara and Alex Pentland. 1998. Maximum conditional likelihood via bound maximization and the CEM algorithm. In *Advances in Neural Information Processing Systems 11 (NIPS'98)*.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, College Park, MD.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop*.

John Maxwell and Ron Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–589.

Thomas Minka. 2001. Algorithms for maximum-likelihood logistic regression. Department of Statistics, Carnegie Mellon University.

Miles Osborne. 2000. Estimation of stochastic attribute-value grammars using an informative sample. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware.

William H. Press, Saul A. Teukolsky, Willam T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York.

Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000. Lexicalized Stochastic Modeling of Constraint-Based Grammars using Log-Linear Measures and EM Training. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong.