Parser Evaluation: a Survey and a New Proposal

John Carroll
Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH, UK

Ted Briscoe
Computer Laboratory
University of Cambridge
Pembroke Street
Cambridge CB2 3QG, UK

Antonio Sanfilippo
Sharp Laboratories of Europe Ltd
Edmund Halley Road
Oxford Science Park
Oxford OX4 4GA, UK

Abstract

We present a critical overview of the state-of-the-art in parser evaluation methodologies and metrics. A discussion of their relative strengths and weaknesses motivates a new—and we claim more informative and generally applicable—technique of measuring parser accuracy, based on the use of grammatical relations. We conclude with some preliminary results of experiments in which we use this new scheme to evaluate a robust parser of English.

1. Introduction

A wide variety of parser and/or grammar evaluation methods have been used (and sometimes justified) in the literature. In §2 we describe the most important of these, each with a brief critique. We argue (§3) that no extant method is entirely satisfactory, particularly in view of the wide range of parsing technology currently in use in the NLP research community. With this in mind, in §4 we motivate and present a higher-level and more task orientated language in which to represent the information a parser should extract from a sentence, together with suitable evaluation measures. We conclude (§5) with some preliminary results of experiments in which we use this new scheme to evaluate a robust parser of English. This approach to evaluation is currently being elaborated in the CEC Language Engineering project 'SPARKLE'[1].

2. Survey of Parser Evaluation Methods

Parser evaluation methods divide into non-corpus and corpus-based methods, with the latter subdividing further into unannotated and annotated corpus-based methods.

Listing Linguistic Constructions Covered *(No corpus)* The traditional approach to evaluating a parser's grammatical coverage consisted of simply listing constructions covered, and hopefully not covered by a given grammar or parser (e.g. Briscoe *et al.*, 1987; Alshawi *et al.*, 1992). The advantages of this approach are that it requires no corpus resources, the grammar developer can easily give this information, and an indication of constructions *not* covered can be useful. Disadvantages are that as the distinction between core and peripheral aspects of constructions is not well-defined and much of the complexity can be in their interaction, it provides no precise information about actual coverage of data. For instance, a grammarian who claims to have covered English relative clauses will probably not have dealt with (rarish, dialect-specific, usually spoken) cases containing resumptive pronouns: *a poorly-drafted law which it is difficult to see how* **it** *could be enforced* or much more common non-parallel 'coordinations' of the form *an inch lower and it would have knocked him out* because of their perceived marginal status. Similarly, coverage of coordination and verbal complementation does not guarantee coverage of their interaction.

Coverage *(Unannotated corpus)* Calculation of the percentage of sentences from a given, unannotated corpus that are assigned one or more analyses by a parser/grammar is a useful first step in evaluation (e.g. Black, Garside & Leech, 1993; Briscoe & Carroll, 1995). It can be computed for a large corpus (given an efficient enough parser) and does not require corpus annotation. However, in the current context it is a weak measure because it does not guarantee that any of the analyses found are correct, so there are an unknown number of false positives in the result which can only be found by manually checking the output. In addition, as a sole criterion it is open to abuse: for example on this measure the trivial grammar `S` $\rightarrow$ `word*` would get a perfect score.

Average Parse Base *(Unannotated corpus)* Black, Garside & Leech (1993) define the parse base (PB) of a grammar as the geometric mean of the number of analyses divided by the number of input tokens in each sentence parsed. This yields a measure from which one can predict the average number of (undisambiguated) parses for a sentence of of length n words in a particular corpus and analysed with that grammar/parser. A problem with PB as defined by Black *et al.* is that in being based on the ratio of analyses to tokens it implicitly assumes numbers of parses grow roughly linearly with sentence length. However in practice the relationship is more nearly exponential, causing the PB measure to overestimate the expected number of parses for shorter sentences and underestimate the number for longer sentences. Briscoe & Carroll (1995) therefore propose an improved measure, the average parse base (APB): "the geometric mean over all sentences in the corpus of $\sqrt[n]{p}$, where n is the number of words in a sentence, and p, the number of parses for that sentence". Thus the APB to the power of n gives the number of parses that would be expected for a sentence of length n.

[1] Information about the SPARKLE project is available at <http://www.ilc.pi.cnr.it/sparkle.html>.

Again, given an efficient enough parser this measure is easily computed, even for large corpora, and gives a succinct measure of the degree of ambiguity in a grammar, especially with respect to different versions of a grammar developed within the same framework and tested on the same data. On the other hand, a very low coverage unambiguous grammar would do very well on this measure alone, so it is probably best utilised in conjunction with an independent measure of coverage. The main disadvantage though, is that it is not possible to meaningfully compare performance of different parsers on different data, as the inherent ambiguity in the data interacts with that of the grammar.

Entropy/Perplexity *(Unannotated corpus)* A probabilistic language model (stochastic language generator / grammar) can be used to calculate the entropy of a corpus (or equivalently perplexity of the model). This yields a measure of the degree to which a model captures regularities in the corpus by minimising unpredictability and ambiguity. Although in basic form, the measure is specific to a given probabilistic language model and corpus, the approach can be used to compare the effectiveness of different language models or the effectiveness of different training regimes (estimates of probabilities on the same test data). In addition, the approach can be generalised to provide a model-independent measure of the inherent complexity of a corpus (e.g. Sharman, 1990) by successively approximating n-gram language models of greater and greater n. This method of computing generalised entropy/perplexity is, however, expensive to compute.

The main advantages of the basic measure are that it has a clear interpretation in probabilistic context and allows meaningful comparison of different probabilistic models of the same corpus. The disadvantages are that it is only applicable to probabilistic models, and only provides, at best, a weak measure of the accuracy of (highly-ranked) derivations (as opposed to the predictability of the data or ambiguity in the model). The generalised measure could conceivably be used as a method for scaling results obtained using other corpus-dependent measures to allow for some degree of cross-corpus comparison and evaluation (such as when parser/grammars are applied to different languages and consequently test corpora must differ).

Part-of-speech Assignment Accuracy *(Annotated corpus)* The accuracy with which a part-of-speech tagger (or parser/ grammar) assigns the correct lexical syntactic category to a word in running text has been measured in a variety of ways: ratio of correct tags per word, possibly subdivided into ambiguous and unambiguous and/or known and unknown words (that is, words not seen in the training data), and so forth (e.g. Church, 1988; Karlsson *et al.*, 1995). Probably, the most satisfactory measure for this task is precision/recall as this generalises to the case where more than one tag remains.

An advantage of using this measure is that there is a great deal of manually-corrected part-of-speech tagged material to use as test corpora (at least for English and increasingly for other West European languages). The disadvantages are that many broad-coverage parsers take pre-tagged text as input, rendering the measure inappropriate for evaluating the parser/grammar *per se*. And, in other cases, it only provides a very partial measure of the accuracy of a syntactic analysis, so has been mostly used to evaluate taggers, and other 'lexical' parsers Karlsson *et al.*, 1995).

Structural Consistency *(Annotated corpus)* Black, Garside & Leech (1993) define structural consistency as the percentage of sentences in a manually-annotated test corpus which receive one or more analyses which are consistent with the correct analysis in the corpus (according to the crossing brackets measure, defined precisely below). The crossing brackets measure provides a weak measure of structural consistency in which some types of conflicting constituency are recognised. The measure is stronger than simply measuring coverage but requires an annotated corpus and makes minimal use of these annotations. Alone, crossing brackets is an inadequate measure as it will tend to favour systems which yield minimal structure—for example there will be no crossing brackets if the parser uses the grammar `S` $\rightarrow$ `word*`.

Best-first/Ranked Consistency *(Annotated corpus)* Briscoe & Carroll (1993) measure the accuracy of a probabilistic parsing system by computing the percentage of highest-ranked analyses output by the parser/ grammar which are identical to a manual analysis provided in an annotated test corpus (treebank). It is straightforward to extend this measure so that the top n analyses are compared, and scoring is relative to their ranking. This measure gives a meaningful score of how often a parser would deliver an appropriate analysis and also some indirect measure of degree of ambiguity in a grammar since, in general, the larger the set to choose from the less likelihood ranking techniques will succeed. However, it is very dependent on having access to an accurate, coherent and consistent annotated corpus which is fully compatible with parser output.

Tree Similarity *(Annotated corpus)* Sampson, Haigh & Atwell (1989) and others define tree similarity measures of various types, such as the ratio of rules correctly deployed in a derivation to all rules in that derivation computed from an annotated corpus. These measures are more fine-grained than full identity in that a grammar/ parser may never produce entirely correct analyses yet consistently produce analyses which are 'close' to the correct one. They are also more tolerant of errors in the manually annotated test data. However, for such weaker measures than full identity it is difficult to see how they map onto many parsing tasks: for instance with an '80% correct' tree, how important is the remaining 20% to the correct interpretation of the sentence? Finally, these measures still require a detailed and compatible annotated test corpus.

Grammar Evaluation Interest Group Scheme *(Annotated corpus)* The Grammar Evaluation Interest Group (GEIG; see Grishman, Macleod & Sterling, 1992) scheme is currently the most widely-used parser evaluation method. It is basically a relaxation of full identity as the success criterion to one which measures similarity of an analysis to a test corpus analysis. The emphasis is on comparing performance of different parsers utilising different grammatical frameworks. The original version of the scheme utilises only phrase-structure bracketing information from the annotated corpus[2] and computes the number of bracketing matches M with respect to the number of bracketings P returned by the parser (expressed as precision M/P) and with respect to the number C in the corpus (expressed as recall M/C), and the mean number of 'crossing' brackets per sentence where a bracketed sequence from the parser overlaps with one from the treebank and neither is properly contained in the other.

Advantages of GEIG are that a relatively undetailed (only bracketed), less parser-specific annotation is required, some level of cross framework/system comparison is achieved, and the measure is moderately fine-grained and robust to annotation errors. The disadvantages are that it is much weaker than full identity and it is unclear how much of a success (or failure) it is to achieve high (or low) scores: for example, Carroll & Briscoe (1996) remark that a single attachment mistake embedded n levels deep (and perhaps completely innocuous, such as an "aside" delimited by dashes) can lead to n crossings being assigned, whereas incorrect identification of arguments and adjuncts can go unpunished in some cases.

Carpenter & Manning (1997) observe that sentences in the Penn Treebank (PTB; Marcus, Santorini & Marcinkiewicz, 1993) contain relatively few brackets, so analyses are quite 'flat'. (The same goes for the other treebank of English in general use, SUSANNE; Sampson, 1995). Thus crossing bracket scores are likely to be small, however good or bad the parser is. Carpenter & Manning also point out that with the adjunction structure the PTB gives to post noun-head modifiers *(NP (NP the man) (PP with (NP a telescope)))*, there are zero crossings in cases where the VP attachment is incorrectly returned, and *vice-versa*. Conversely, Lin (1995) demonstrates that the crossing brackets measure can in some cases penalise mis-attachments more than once; Lin (1996) argues that a high score for phrase boundary correctness does not guarantee that a reasonable semantic reading can be produced. Conversely, many phrase boundary disagreements stem from systematic differences between parsers/grammars and corpus annotation schemes that are well-justified within the context of their own theories. GEIG does attempt to circumvent this problem by the removal from consideration of bracketing information in constructions for which agreement between analysis schemes in practice is low: i.e. negation, auxiliaries, punctuation, traces, and the use of unary branching structures.

However, in general there are still major problems with compatibility between the annotations in treebanks and analyses returned by parsing systems using manually-developed generative grammars (as opposed to grammars acquired directly from the treebanks themselves). The treebanks have been constructed with reference to sets of informal guidelines indicating the type of structures to be assigned. In the absence of a formal grammar controlling or verifying the manual annotations, the number of different structural configurations tends to grow without check. For example, the PTB implicitly contains more than 10000 distinct context-free productions, the majority occurring only once (Charniak, 1996). This makes it very difficult to accurately map the structures assigned by an independently-developed grammar onto the structures that appear (or should appear) in the treebank. A further problem is that the GEIG bracket precision measure penalises parsers that return more structure than the treebank annotation, even if it is correct (Srinivas, Doran & Kulick, 1995). To be able to use the treebank and report meaningful GEIG precision scores such parsers must necessarily 'dumb down' their output and attempt to map it onto (exactly) the distinctions made in the treebank[3]. This mapping is also very difficult to specify accurately. The results of evaluation are thus distorted.

In addition, since GEIG is based on measuring similarity between phrase-structure trees, it cannot be applied to grammars which produce dependency-style analyses, or to 'lexical' parsing frameworks such as finite-state constraint parsers which assign syntactic functional labels to words rather than producing hierarchical structure.

Dependency Structure-based Scheme *(Annotated corpus)* To overcome the GEIG grammar/treebank mismatch problems outlined above, Lin (1995) proposes evaluation based on dependency structure, in which phrase structure analyses from parser and treebank are both automatically converted into sets of dependency relationships. Each such relationship consists of a modifier, a modifiee, and optionally a label which gives the type of the relationship. Atwell (1996), though, argues that transforming standard constituency-based analyses into a dependency-based representation would lose certain kinds of grammatical information that might be important for subsequent processing, such as 'logical' information (e.g. location of traces, or moved constituents). Srinivas, Doran, Hockey & Joshi (1996) describe a related technique which could also be applied to partial (incomplete) parses, in which hierarchical phrasal constituents are flattened into chunks and the relationships between them are indicated by dependency links. Recall and precision are defined over dependency links.

The TSNLP (Lehmann *et al.*, 1996) project test suites (in English, French and German) contain dependency-based annotations for some sentences; this allows for "generalizations over potentially controversial phrase structure configurations" and also mapping onto a specific constituent structure. No specific evaluation measures are proposed, though.

3. Evaluating Parser Evaluation Methods

Before proceeding, we should make the distinction between parser evaluation methods that are useful in guiding

[2] More recent evaluations using GEIG (e.g. Magerman, 1995; Collins, 1996) have adapted it to incorporate constituent labelling information as well as just bracketing.

[3] Gaizauskas, Hepple & Huyck (1998) propose an alternative to the GEIG precision measure to address this shortcoming.

and monitoring the development of a particular grammar/ parsing system, and those that are appropriate for comparing different systems. For the former, one might use *listing linguistic constructions covered*, *coverage* and *average parse base* with respect to a particular unannotated corpus, and *structural consistency* and possibly *best-first/ranked consistency* with respect to a manually-constructed corpus of subjectively correct analyses. This approach, though, will not generalise simply to evaluation with other corpora or against other systems because there is no way of framework-neutrally measuring the difference in the syntactic complexity between two corpora, and different grammatical frameworks produce more/less informative descriptions which will make all these measures incommensurable.

Inter-system comparison is by far the harder problem, and it is clear that an evaluation scheme must include some method for assessing the utility of n ranked analyses, possibly in conjunction with ancilliary measures such as coverage, as mentioned above. The main weakness of all extant methods is that they do not relate analyses to potential parsing tasks or applications. In particular, an evaluation measure which treats full identity as a success criterion has a more straightforward interpretation in terms of putative parser tasks, such as construction of a logical form or identification of phrasal heads / boundaries, than one which relaxes this criterion to some measure of 'closeness'. This is because the definition of 'closeness' and the utility of an analysis that is less-than-perfect along some dimension will vary from task to task. Thus, for construction of logical form a criterion of total identity seems quite appropriate, while for recognition of phrasal heads for, say, extraction of statistical data relating to the semantic type of such heads, accuracy in extracting heads is important but correct attachment of prepositional phrases may not be.

Another problem which is not satisfactorily addressed by extant proposals is that the evaluation scheme needs to be applicable to the output of parsers based on rather different design philosophies applied to different languages and test corpora. To make these points more concretely, consider three extant parsers which produce very different representations. Here are three analyses for *John tried to open the window*:

- The robust shallow parser of Briscoe & Carroll (1995):

```
(S (NP (N1 (N John_NP1)))
 (VP (V tried_VVD)
  (VP (V to_TO)
   (VP (V open_VV0)
    (NP (DT the_AT)
     (N1 (N window_NN1)))))))
```

- Fast Partial Parser (Grefenstette, 1994):

```
SUBJ(try,John) DOBJ(open,window)
SUBJ(open,John) MODIF(open,try)
```

- Finite State Constraint Grammar Parser (Karlsson *et al.*, 1995):

```
John N SUBJ tried V MV MAINC^ to
INFMARK open V_INF mv OBJ^ the DET
window N obj
```

or, using indentation rather than ˆ to indicate (some) constituency:

```
subject: John
main verb: tried
object: main verb: open
        object: window
```

It is intuitively clear that each of these representations conveys similar but not identical information—for instance it seems less informative to, for example, use the same relation (MODIF) to indicate the relationship between *open* and *try* that is also used for adjective (or noun) predications, than to label *open window* the object of *try*. It also seems less informative to assign *John* the functional category SUBJ than to assign the two relations (SUBJ open,John) and (SUBJ try,John).

To make these intuitions precise one could develop translation functions between these notations or, taking one as the anchor, between this and all the others. However, this is fraught with problems. For example, we must necessarily throw information away when we translate from a 'more informative' to 'less informative' notation. But we don't even have a definitive interpretation for the target notation since it is system-specific.

A better approach would be to pick an independent language in which to represent the information a parser should extract. The agreed definition of the information ought to at least represent all grammatical relations (GRs) in an explicit and (ultimately) unambiguous notation, so we could use a feature structure (FS) and base our language loosely on LFG F-structure in AVM notation:

```
[ PRED try
  SUBJ John <1>
  XCOMP [ PRED open
          SUBJ <1>
          OBJ window]]
```

where coindexation with <n> indicates reentrancy in the underlying DAG. If we were to define the space of possible DAGs that a well-formed AVM can pick out by saying what set of GR attributes can occur (`PRED`, `SUBJ`, `XCOMP` etc.) and what their possible values are, then for any sentence we could measure the information returned by a parser by (1) computing the upper bound on the size of the set of possible DAGs for the sentence, (2) translating each parser's output into all possible well-formed AVM representations for that sentence, and (3) computing the size of the set denoted by the resulting AVM(s). Possibly step (1) could be dispensed with, just computing for the set returned the number of possible DAGs remaining.

The more a parser rules out possible DAGs, the more informative it will be. Thus, if a parser turns out to generate a set of AVMs like:

```
[ PRED try
  SUBJ John
  XCOMP [ PRED open
          SUBJ <?>
          OBJ window]]
```

(where <?> stands for a set of possible coindexations with any word in the sentence), then we enumerate the number of remaining DAGs and compare this with those for:

```
[ PRED try
  SUBJ John <1>
  ??  [ PRED open
          SUBJ <1>
          OBJ window]]
```

(where `??` stands for the set of possible GR attributes). This AVM (arguably) what would be recovered from Grefenstette's representation; compare these with those for Karlsson *et al.*'s which (arguably) would look like:

```
[ PRED try
  SUBJ <??>
  XCOMP [ PRED open
          SUBJ <?>
          OBJ window]]
```

where one of `<??>/<?>` would be replaced by *John*.

This scheme, although principled and possibly applicable to a wide range of parsers, languages and corpora, is considerably more ambitious than extant evaluation methods, and is perhaps over-complex. A simpler way of measuring information extracted would be to collapse the AVM descriptions into a flat conjunction of propositions, each representing a single grammatical relation between heads of constituents, e.g. `subj(try,John)`, and compute precision/recall measures in the standard fashion against a set of these relations resulting from manual annotation of the test corpus. We further motivate and describe such an approach in more detail in the next section[4].

4. A New Parser Evaluation Scheme

In specifying the grammatical relation annotation scheme (see also Carroll *et al.*, 1997a), our point of departure is the work on subcategorization standards developed within EAGLES by the lexicon/syntax interest group (Sanfilippo *et al.*, 1996). In EAGLES, a three-layered approach to the specification of grammatical dependencies for verbal arguments was followed. The first layer identifies the subject/ complement and ±predicative distinctions as the most general specifications; this layer is regarded as encoding mandatory information. The second layer provides a further partition of complements into direct and indirect as recommended specifications. Finally, a more fine-grained distinction qualified as useful is envisaged introducing further labels for clausal complements and second objects. For further details, see (Sanfilippo *et al.*, 1996).

The first step in tailoring the EAGLES standards to the needs of parser evaluation has been to make provisions for modifiers. These were not treated in EAGLES since only subcategorizable functions were taken into consideration. Secondly, the relationship among layers of grammatical dependency specifications has been interpreted in terms of a hierarchy of relation types.

[4] It should be noted that in proposing an evaluation scheme based on grammatical relations we are not advocating their use as a parser output representation for use in application tasks. Our goal is rather to specify an evaluation method that measures relative correctness of parser analyses with respect to a standard representation that is well-defined and both system- and task-independent.

In general, grammatical relations (GRs) are viewed as specifying the syntactic dependency which holds between a head and a dependent. In the event of morphosyntactic processes modifying head-dependent links (e.g. the passive, dative shift and causative-inchoative diatheses), two kinds of GRs can be expressed: (1) the initial GR, i.e. before the GR-changing process occurs; and (2) the final GR, i.e. after the GR-changing process occurs. For example, *Paul* in *Paul was employed by Microsoft* is both the initial object and the final subject of *employ*[5].

In relying on the identification of grammatical relations between headed constituents, we of course presuppose a parser/ grammar that is able to identify heads. In theory this may exclude certain parsers from using this scheme, although we are not aware of any contemporary computational parsing work which eschews the notion of head and moreover is unable to recover them. Thus, in computationally-amenable theories of language, such as HPSG (Pollard & Sag, 1994) and LFG (Kaplan & Bresnan, 1982), and indeed in any grammar based on some version of X-bar theory (Jackendoff, 1977), the head plays a key role. Likewise, in recent work on statistical treebank parsing, Magerman (1995) and Collins (1996) propagate information on each constituent's head up the parse tree in order to be able to capture lexical dependencies. Headedness is also important in each of the three robust parsing systems illustrated in §3.

The hierarchical organisation of GRs is shown graphically in figure 1. Each GR in the scheme is described individually below.

dependent(introducer,head,dependent) This is the most generic relation between a head and a dependent (i.e. it does not specify whether the dependent is an argument or a modifier). E.g.

dependent(in,live,Rome)	Marisa lives in Rome

mod(type,head,dependent) The relation between a head and its modifier; where appropriate, **type** indicates the word introducing the dependent; e.g.

mod(_,flag,red)	a red flag
mod(_,walk,slowly)	walk slowly
mod(with,walk,John)	walk with John
mod(while,walk,talk)	walk while talking
mod(_,Picasso,painter)	Picasso the painter

The **mod** GR is also used to encode the relationship between an event noun (including deverbal nouns) and its participants; e.g.

mod(of,gift,book)	the gift of a book
mod(by,gift,Peter)	the gift ... by Peter
mod(of,examination,patient)	the examination of the patient
mod('s,doctor,examination)	the doctor's examination

cmod, xmod, ncmod Clausal and non-clausal modifiers may (optionally) be distinguished by the use of the GRs

[5] In the EAGLES standards, initial and final GRs are expressed in the frame-set structures which relate systematic valency alternations of the same predicate.

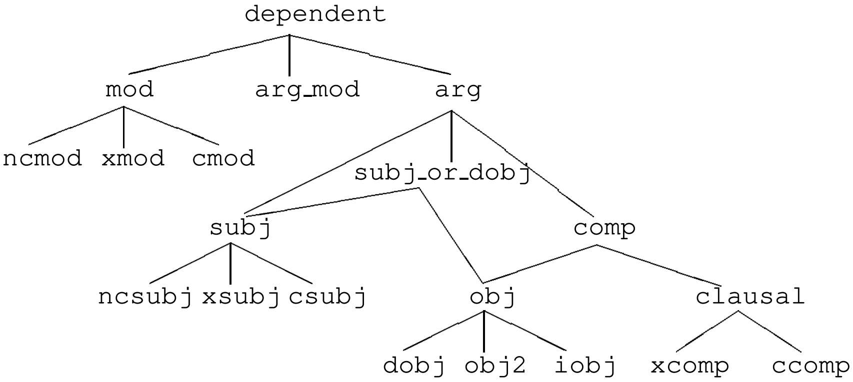

Figure 1: The Grammatical Relation Hierarchy

cmod / xmod, and **ncmod** respectively, each with slots the same as **mod**. The GR **ncmod** is for non clausal modifiers; **cmod** is for adjuncts controlled from within, and **xmod** for adjuncts controlled from without, e.g.

xmod(without,eat,ask)	he ate the cake without asking
cmod(because,eat,be)	he ate the cake because he was hungry

arg_mod(type,head,dependent,initial_gr) The relation between a head and a semantic argument which is syntactically realised as a modifier; thus a by-phrase can be analysed as a 'thematically bound adjunct'. The **type** slot indicates the word introducing the dependent: e.g.

arg_mod(by,kill,Brutus,subj)	killed by Brutus

arg(head,dependent) The most generic relation between a head and an argument.

subj_or_dobj(head,dependent) A specialization of the relation **arg** which can instantiate either subjects or direct objects. It is useful for those cases where no reliable bias is available for disambiguation. For example, both *Gianni* and *Mario* can be subject or object in the Italian sentence

Mario, non l'ha ancora visto, Gianni
'Mario has not seen Gianni yet'/'Gianni has not seen Mario yet'

In this case, the parser could avoid trying to resolve the ambiguity by using **subj_or_dobj**, e.g.

subj_or_dobj(vedere,Mario)
subj_or_dobj(vedere,Gianni)

subj(head,dependent,initial_gr) The relation between between a predicate and its subject; where appropriate, the **initial_gr** indicates the syntactic link between the predicate and subject before any GR-changing process:

subj(arrive,John,_)	John arrived in Paris
subj(employ,Microsoft,_)	Microsoft employed 10 C programmers
subj(employ,Paul,obj)	Paul was employed by IBM

With pro-drop languages such as Italian, when the subject is not overtly realised the annotation is, for example, as follows:

subj(arrivare,Pro,_)	arrivai in ritardo '(I) arrived late'

in which the dependent is specified by the abstract filler **Pro**, indicating that person and number of the subject can be recovered from the inflection of the head verb form.

csubj, xsubj, ncsubj The GRs **csubj** and **xsubj** indicate clausal subjects, controlled from within, or without, respectively. **ncsubj** is a non-clausal subject. E.g.

csubj(leave,mean,_)	that Nellie left without saying good-bye meant she was angry
xsubj(win,require,_)	to win the America's Cup requires heaps of cash

comp(head,dependent) The most generic relation between a head an complement.

obj(head,dependent) The most generic relation between a head an object.

dobj(head,dependent,initial_gf) The relation between a predicate and its direct object—the first non-clausal complement following the predicate which is not introduced by a preposition (for English and German); **initial_gf** is **iobj** after dative shift; e.g.

dobj(read,book,_)	read books
dobj(mail,Mary,iobj)	mail Mary the contract

iobj(type,head,dependent) The relation between a predicate and a non-clausal complement introduced by a preposition; **type** indicates the preposition introducing the dependent; e.g.

iobj(in,arrive,Spain)	arrive in Spain
iobj(into,put,box)	put the tools into the box
iobj(to,give,poor)	give to the poor

obj2(head,dependent) The relation between a predicate and the second non-clausal complement in ditransitive constructions; e.g.

obj2(give,present)	give Mary a present
obj2(mail,contract)	mail Paul the contract

clausal(head,dependent) The most generic relation between a head and a clausal complement.

xcomp(type,head,dependent) The relation between a predicate and a clausal complement which has no overt subject (for example a VP or predicative XP). The **type** slot is the same as for **ccomp** above. E.g.

xcomp(to,intend,leave)	Paul intends to leave IBM
xcomp(_,be,easy)	Swimming is easy
xcomp(in,be,Paris)	Mary is in Paris
xcomp(_,be,manager)	Paul is the manager

Control of VPs and predicative XPs is expressed in terms of GRs. For example, the unexpressed subject of the clausal complement of a subject-control predicate is specified by saying that the subject of the main and subordinate verbs is the same:

Paul intends to leave IBM	*subj(intend,Paul,_)*
	xcomp(to,intend,leave)
	subj(leave,Paul,_)
	dobj(leave,IBM,_)

ccomp(type,head,dependent) The relation between a predicate and a clausal complement which does have an overt subject; **type** indicates the complementiser / preposition, if any, introducing the clausal XP. E.g.

ccomp(that,say,accept)	Paul said that he will accept Microsoft's offer
ccomp(that,say,leave)	I said that he left

This annotation scheme is superficially similar to a syntactic dependency analysis in the style of Lin (1995). However, there are several differences: (1) the GR analysis of control relations cannot be expressed as a strict dependency tree since a single head would be the modifiee of two (or more) VP heads (as with *Paul* in *Paul intends to leave IBM*); (2) arguments "displaced" from their canonical positions by movement phenomena are associated with the underlying grammatical relation in the annotation[6]; (3) semantic arguments syntactically realised as modifiers (e.g. the passive by-phrase) can be indicated as such; and (4) arguments that are not lexically realised can appear in GRs (e.g. as **Pro** for subject when there is pro-drop). GRs are organised into a hierarchy so that they can be underspecified; the GR hierarchy has been developed and refined taking into account language phenomena in English, Italian, French and German.

As alluded to in the previous section, parser evaluation is based on computing recall and precision measures over grammatical relations. We compute for each sentence:

- *recall* as the ratio of the number of GRs returned by the parser that *match* (see below) GRs in the corresponding annotated sentence, divided by the total number of GRs in the annotated sentence; and

- *precision* as the ratio of the number of GRs returned by the parser that *match*, divided by the total number of GRs returned by the parser for that sentence.

[6] This scheme therefore does not fall prey to the criticism that Atwell makes (§2) of dependency grammar-based evaluation schemes.

Overall recall and precision are the means of the respective measures over all sentences in the test corpus.

GRs *match* if they are identical. The case where the functors stand in a subsumption relationship and the arguments are identical is defined as a *partial match*; for example, a **dobj** chunk is compatible with an underspecified chunk marked as **comp**, meaning one head stands in a relationship to the other as either an object or a clausal complement. Thus, a parser is able to return an underspecified representation—without being unduly penalised—in situations where it does not have access to information that would help it to resolve the ambiguity. A partial match should probably be assigned a score less than the value of 1 assigned to complete matches, but to date we have not attempted to determine this value precisely: it should probably depend on the degree of underspecification.

5. Using the New Evaluation Scheme

The new evaluation scheme is currently being used in SPARKLE to evaluate four wide-coverage parsers (for English, French, German and Italian) each based on a different design philosophy. In this section we briefly present some preliminary results of its use with the English parsing system (basically a more developed, minimally lexicalised version of the system described by Carroll & Briscoe, 1996). More details may be found in Carroll *et al.*, 1997b.

We used a test corpus consisting of 500 randomly chosen in-coverage but unseen sentences from SUSANNE, marked up manually to a constituent-based phrasal scheme, and also to the GR scheme by automatically extracting a preliminary set of GRs for each sentence, and then manually correcting the results. The relatively large size of the test corpus has meant that to date we have not marked up modification relations, so at this stage we report an evaluation with respect to argument relations only (but including the relation *arg_mod*). In general the parser returns the most specific (leaf) relations in the GR hierarchy, except when it is unable to determine the correct control alternative (i.e. *csubj* vs. *xsubj*, and *ccomp* vs. *xcomp*), in which case it returns *subj* or *clausal* as appropriate. The mean number of GRs per sentence in the test corpus is 4.15.

When computing matches between the GRs produced by the parser and those in the corpus annotation, a single level of subsumption is allowed: a relation from the parser is allowed to be one level higher in the GR hierarchy than the actual correct relation. For example, if the parser returns *clausal*, this is taken to match both the more specific *xcomp* and *ccomp*. Also, an unspecified filler (_) for the type slot in the *iobj* and *clausal* relations successfully matches any actual specified filler. Table 1 gives the results of the evaluation. In the near future we intend to validate the existing argument relations and extend the test corpus to cover modification relations. We also want to test the sensitivity of the two types of scheme—the conventional constituent-based scheme and the GR scheme—to enhancements to the parser. In particular, we have tentative empirical evidence that the GR scheme is better at reflecting improvements in the parser's ability to distinguish arguments and adjuncts.

Zero crossings (% sents.)	56.6
Mean crossings per sent.	1.10
Bracket recall (%)	83.1
Bracket precision (%)	83.1
GR recall (%)	88.1
GR precision (%)	88.2

Table 1: Constituent- and GR-based Evaluation Results

6. Acknowledgements

This work was supported by CEC Telematics Applications Programme project LE1-2111 'SPARKLE: Shallow PARsing and Knowledge extraction for Language Engineering', and by an EPSRC Advanced Fellowship to the first author. We would like to thank other researchers in the SPARKLE consortium—in particular Nicoletta Calzolari, Glenn Carroll, Stefano Federici, Greg Grefenstette, Simonetta Montemagni, Vito Pirrelli and Mats Rooth—for useful feedback during the formulation of the new grammatical relation evaluation scheme.

7. References

Alshawi, H. (Ed.) (1992). *The Core Language Engine*. Cambridge, MA: MIT Press.

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

Black, E., Garside, R. & Leech, G. (Eds.) (1993). *Statistically-driven computer grammars of English: The IBM / Lancaster approach*. Amsterdam, The Netherlands: Rodopi.

Briscoe, E. & Carroll, J. (1993). Generalised probabilistic LR parsing for unification-based grammars. *Computational Linguistics*, 19(1), 25–60.

Briscoe, E. & Carroll, J. (1995). Developing and evaluating a probabilistic LR parser of part-of-speech and punctuation labels. In *Proceedings of the 4th ACL/SIGPARSE International Workshop on Parsing Technologies* (pp. 48–58). Prague, Czech Republic.

Briscoe, E., Grover, C., Boguraev, B. & Carroll, J. (1987). A formalism and environment for the development of a large grammar of English. In *Proceedings of the IJCAI-87* (pp. 703–708). Milan, Italy.

Carpenter, B. & Manning, C. (1997). Probabilistic parsing using left corner language models. In *Proceedings of the 5th ACL/SIGPARSE International Workshop on Parsing Technologies*. MIT, Cambridge, MA.

Carroll, J. & Briscoe, E. (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* (pp. 92–100). University of Pensylvania, Philadelphia, PA.

Carroll, J., Briscoe, E., Calzolari, N., Federici, S., Montemagni, S., Pirrelli, V., Grefenstette, G., Sanfilippo, A., Carroll, G. & Rooth, M. (1997a). *SPARKLE WP1 specification of phrasal parsing*. <http://www.ilc.pi.cnr.it/sparkle.html>.

Carroll, J., Briscoe, E., Carroll, G., Light, M., Prescher, D., Rooth, M., Federici, S., Montemagni, S., Pirrelli, V., Prodanof, I. & Vanocchi, M. (1997b). *SPARKLE WP3.2 phrasal parsing software*. <http://www.ilc.pi.cnr.it/sparkle.html>.

Charniak, E. (1996). Tree-bank grammars. In *Proceedings of the 13th National Conference on Artificial Intelligence, AAAI-96* (pp. 1031–1036). .

Church, K. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd ACL Conference on Applied Natural Language Processing* (pp. 136–143). Austin, Texas.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Meeting of the Association for Computational Linguistics* (pp. 184–191). Santa Cruz, CA.

Gaizauskas, R., Hepple M. & Huyck, C. (1998). Modifying existing annotated corpora for general comparative evaluation of parsing. In *Proceedings of the LRE Workshop on Evaluation of Parsing Systems*. Granada, Spain.

Grefenstette, G. (1994). *Explorations in automatic thesaurus discovery*. Dordrecht, The Netherlands: Kluwer.

Grishman, R., Macleod, C. & Sterling, J. (1992). Evaluating parsing strategies using standardized parse files. In *Proceedings of the 3rd ACL Conference on Applied Natural Language Processing* (pp. 156–161). Trento, Italy.

Jackendoff, R. (1977). *X-bar syntax*. Cambridge, MA: MIT Press.

Kaplan, R. & Bresnan, J. (1982). Lexical-Functional Grammar: a formal system for grammatical representation. In J. Bresnan (Eds.), *The Mental Representation of Grammatical Relations* (pp. 173–281). Cambridge MA: MIT Press.

Karlsson, F., Voutilainen, A., Heikkilä, J. & Anttila, A. (1995). *Constraint grammar: a language-independent system for parsing unrestricted text*. Berlin, Germany: de Gruyter.

Lehmann, S., Oepen, S., Regnier-Prost, S., Netter, K., Lux, V., Klein, J., Falkedal, K., Fouvry, F., Estival, D., Dauphin, E., Compagnion, H., Baur, J., Balkan, L. & Arnold, D. (1996). TSNLP — test suites for natural language processing. In *Proceedings of the International Conference on Computational Linguistics, COLING-96* (pp. 711–716). Copenhagen, Denmark.

Lin, D. (1995). A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the IJCAI-95* (pp. 1420–1425). Montreal, Canada.

Lin, D. (1996). Dependency-based parser evaluation: a study with a software manual corpus. In R. Sutcliffe, H-D. Koch & A. McElligott (Eds.), *Industrial Parsing of Software Manuals* (pp. 13–24). Amsterdam, The Netherlands: Rodopi.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. Boston, MA.

Marcus, M., Santorini, B. & Marcinkiewicz (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 313–330.

Pollard, C. & Sag, I. (1994). *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.

Sampson, G. (1995). *English for the computer*. Oxford, UK: Oxford University Press.

Sampson, G., Haigh, R. & Atwell, E. (1989). Natural language analysis by stochastic optimization: a progress report on Project APRIL. *Journal of Experimental and Theoretical Artificial Intelligence*, 1, 271–287.

Sanfilippo, A., Barnett, R., Calzolari, N., Flores, S., Hellwig, P., Leech, P., Melero, M., Montemagni, S., Odijk, J., Pirrelli, V., Teufel, S., Villegas M. & Zaysser, L. (1996). *Subcategorization standards*. Report of the EAGLES Lexicon/Syntax Interest Group. Available through `eagles@ilc.pi.cnr.it`.

Sharman, R. (1990). Evaluating a grammar as a language model for speech. In L. Torres, E. Masgrau & M. Lagunas (Eds.), *Signal Processing V: Theories and Applications* (pp. 1271–1274). The Netherlands: Elsevier.

Srinivas, B., Doran, C., Hockey B. & Joshi A. (1996). An approach to robust partial parsing and evaluation metrics. In *Proceedings of the ESSLLI'96 Workshop on Robust Parsing*. Prague, Czech Republic.

Srinivas, B., Doran, C. & Kulick, S. (1995). Heuristics and parse ranking. In *Proceedings of the 4th ACL/SIGPARSE International Workshop on Parsing Technologies*. Prague, Czech Republic.