Symbolic Preference Using Simple Scoring

Paula S. Newman
newmanp@acm.org

Abstract

Despite the popularity of stochastic parsers, symbolic parsing still has some advantages, but is not practical without an effective mechanism for selecting among alternative analyses. This paper describes the symbolic preference system of a hybrid parser that combines a shallow parser with an overlay parser that builds on the chunks. The hybrid currently equals or exceeds most stochastic parsers in speed and is approaching them in accuracy. The preference system is novel in using a simple, three-valued scoring method (-1, 0, or +1) for assigning preferences to constituents viewed in the context of their containing constituents. The approach addresses problems associated with earlier preference systems, and has considerably facilitated development. It is ultimately based on viewing preference scoring as an engineering mechanism, and only indirectly related to cognitive principles or corpus-based frequencies.

1 Introduction

Despite the popularity of stochastic parsers, symbolic parsing still has some advantages, but is not practical without an effective mechanism for selecting among alternative analyses. Without it, accept/fail grammar rules must either be overly strong or admit very large numbers of parses. .

Symbolic parsers have recently been augmented by stochastic post-processors for output disambiguation, which reduces their independence from corpora. Both the LFG XLE parser (Kaplan et.al. 2004), and the HPSG LinGO ERG parser (Toutanova et al. 2005) have such additions.

This paper examines significant aspects of a purely symbolic alternative: the preference and pruning system of the RH (Retro-Hybrid) parser (Newman, 2007). The parser combines a pre-existing, efficient shallow parser with an overlay parser that builds on the emitted chunks. The overlay parser is "retro" in that the grammar is related to ATNs (Augmented Transition Networks) originated by Woods (1970).

RH delivers single "best" parses providing syntactic categories, syntactic functions, head features, and other information (Figure 1). The parenthesized numbers following the category labels in the figure are preference scores, and are explained further on. While the parses are not quite as detailed as those obtained using "deep" grammars, the missing information, mostly relating to long distance dependencies, can be added at far less cost in a post-parse phase that operates only on a single best parse. Methods for doing so, for stochastic parser output, are described by Johnson (2002) and Cahill et al (2004).

The hybrid parser exceeds most stochastic parsers in speed, and approaches them in accuracy, even based on limited manual "training" on a particular idiom, so the preference system is a successful one (see Section 6), and continues to improve.

The RH preference system builds on earlier methods. The major difference is a far simpler scoring system, which has considerably facilitated overlay parser development. Also, the architecture allows the use of large numbers of preference tests without impacting parser speed. Finally, the treatment of coordination exploits the lookaheads afforded by the shallow parser to license or bar alternative appositive readings.

Section 2 below discusses symbolic preference systems in general, and section 3 provides an overview of RH parser structure. Section 4 describes the organization of the RH preference system and the simplified scoring mechanism. Section 5 discusses the training approach and Section 6 provides some experimental results. Section 7 summarizes, and indicates directions for further work.

Proceedings of the 10th Conference on Parsing Technologies, pages 83–92,
Prague, Czech Republic, June 2007.

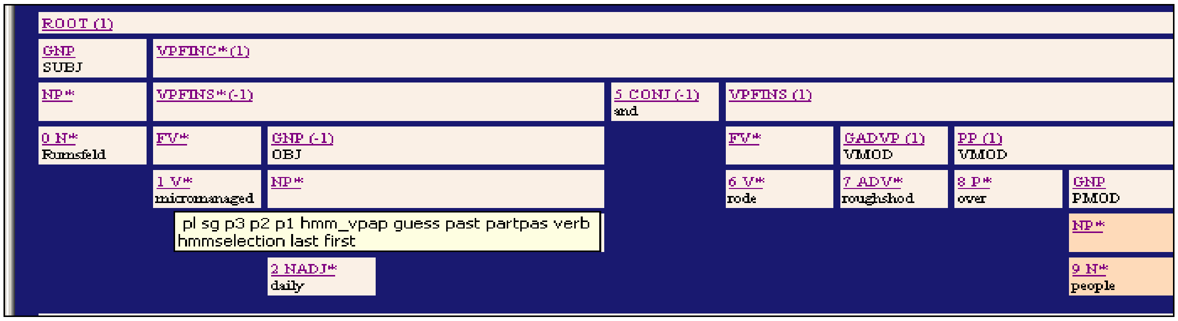

Figure 1. Output Parse Tree for *"Rumsfeld micromanaged daily briefings and rode roughshod over people."* * indicates head. Mouseover shows head features for "micromanaged".

2 Background: Symbolic Preference

2.1 Principles

Preference-based parsing balances necessarily permissive syntactic rules by preference rules that promote more likely interpretations. One of the earliest works in the area is by Wilks (1975), which presented a view of preference as based on semantic templates. Throughout the 1980's there was a considerable amount of work devoted to finding general principles, often cognitively oriented, for preference rules, and then to devise mechanisms for using them in practical systems. Hobbs and Bear (1990) provide a useful summary of the evolved principles. Slightly restated, these principles are:

1. Prefer attachments in the "most restrictive context".
2. If that doesn't uniquely determine the result, attach low and parallel, and finally
3. Adjust the above based on considerations of punctuation

Principle 1 suggests that the preference for a constituent in a construction should depend on the extent to which the constituent meets a narrow set of expectations. Most of the examples given by Hobbs and Bear use either (a) sub-categorization information, e.g., preferring the attachment of a prepositional phrase to a head that expects that particular preposition, or (b) limited semantic information, for example, preferring the attachment of a time expression to an event noun.

Principle 2 implies that in the absence of coordination, attachment should be low, and in the presence of coordination, parallel constituents should be preferred. Principle 3 relates primarily to the effect of commas in modifying attachment preferences.

2.2 Implementations

Abstractly, symbolic preference systems can be thought of as regarding a set of possible parses as a collection of spanning trees over a network of potential relationships, with each edge having a numeric value, and attempting to find the highest scoring tree.[1]

However, for syntactic parsers, in contrast with dependency parsers, it is convenient to associate scores with constituents as they are built, for consistency with the parser structure, and to permit within-parse pruning. A basic model for a preference system assigns preference scores to rules. For a rule

$C \rightarrow c_1, c_2, \ldots, c_n$

the preference score $PS(CC)$ of a resultant constituent CC is the sum:

$$PS(cc_1) + PS(cc_2) + \ \ +PS(cc_n) \\ + TRS\ (C, cc_1, cc_2, \ldots, cc_n)$$

where $PS(cc_i)$ is the non-contexted score of constituent cc_i, and the total relationship score TRS is a value that assesses the relationships among the sibling constituents of CC. The computation of TRS depends on the parser approach. For a top-down parser, TRS may be the sum of contexted relationship scores CRS, for example:

$$TRS = CRS\ (cc_1|C) + CRS\ (cc_2|C, cc_1), + \\ CRS\ (cc_3|C, cc_1, cc_2) + \ldots.. \\ + CRS\ (c_n\ |C,\ cc_1,\ldots.cc_{n-1})$$

where each $CRS\ (cc_i|\_\ )$ evaluates cc_i in the context of the prior content of the constituent CC and the category C..

Few publications specify details of how preference scores are assigned and combined. For example, Hobbs and Bear (1990) say only that "When a

[1] The idea has also been used directly in stochastic parsers that consider all possible attachments, for example, by McDonald et al. (2005).

non-terminal node of a parse tree is constructed, it is given an initial score which is the sum of the scores of its child nodes. Various conditions are checked during the construction of the node and, as a result, a score of 20, 10, 3, -3, -10, or -20 may be added to the initial score."

McCord (1993), however, carefully describes how the elements of *TRS* are computed in his slot grammar system. Each element value is the sum of the results of up to 8 optional, typed tests, relating to structural, syntactic, and semantic conditions. One of these tests, relating to coordination, is a complex test involving 7 factors assessing parallelism.

2.3 Multi-Level Contexted Scoring

The scores assigned by symbolic preference systems to particular relationships or combinations usually indicate not just whether they are preferred or dispreferred, but to what degree. For example, a score of 1 might indicate that a relationship is good, and 2 that it is better.

Such multi-level scores create problems in tuning parsers to remove undesirable interactions, both in the grammar and the preference system. Even for interactions foreseen in advance, one must remember or find out the sizes of the preferences involved, to decide how to compensate. Yamabana et al. (1993) give as an example a bottom-up parser, where an S constituent with a transitive verb head but lacking an object is initially given a strong negative preference, but when it is discovered that the constituent actually functions as a relative clause, the appropriate compensation must be found. (Their solution uses a vector of preference scores, with the vector positions corresponding to specific types of preference features, together with an accumulator. It allows the content of vector elements to be erased based on subsequently discovered compensating features.)

For unforeseen interactions, for example when a review of parser outputs finds that the best parse is not given the highest preference score, multi-level contexted scoring requires complex tracing of the contribution of each score to the total, remembering at each point what the score should be, to determine the necessary adjustments.

A different sort of problem of multi-level scoring stems from the unavoidable incompleteness of information. For example, in Figure 1, the attachment of an object to the "guessed" verb "micro-managed" is dispreferred because the verb is not identified as transitive. Here, the correct reading survives because there are no higher scoring ones. But in some situations, if such a dispreference were given a large negative score, the parser could be forced into very odd readings not compensated for by other factors.

2.4 Corpus-Based Preference

In the early 1990's, the increasing availability and use of corpora, together with a sense that multi-level symbolic preference scores were based on ad-hoc judgments, led to experiments and systems that used empirical methods to obtain preference weights. Examples of this work include a system by Liu et al (1990), and experiments by Hindle and Rooth (1993), and Resnik and Hearst (1993).[2]

These efforts had mixed success, suggesting that while multi-level preference scores are problematic, integrating some corpus data does not solve the problems. In light of later developments, this might be expected. Full-scale contemporary stochastic parsers use a broad range of interacting features to obtain their fine-grained results; frequencies of particular relationships are just one aspect.

2.5 OT-based Preference

A more recent approach to symbolic preference adapts optimality theory to parser and generator preference. Optimality Theory (OT) was originally developed to explain phonological rules (Prince and Smolensky, 1993). In that use, potential rules are given one "optimality mark" for each constraint they violate. The marks, all implicitly negative, are ranked by level of severity. A best rule ***R*** is one for which (a) the most severe level of constraint violation ***L*** is $\leq$ the level violated by any other rule, and (b) if other rules also violate level ***L*** constraints, the number of such violations is $\geq$ the number of violations by ***R***.

As adapted for use in the XLE processor for LFG (Frank et al. 1998) optimality marks are associated with parser and generator outputs. Positive marks are added, and also labeled inter-mark positions within the optimality mark ranking. The labeled positions influence processor behavior. For generation, they are used to disprefer infelicitous strings accepted in a parse direction. And for pars-

[2] McCord (1993) also includes some corpus-based information, but to a very limited extent.

ing they can be used to disprefer (actually ignore) rarely-applicable rules, in order to reduce parse time (Kaplan et al, 2004).

However, because the optimality marks are global, a single dispreference can rule out an entire parse. To partially overcome this limitation, a further extension (see XLE Online Documentation) allows direct comparisons of alternative readings for the same input extent. A different optimality mark can be set for each reading, and the use of one such mark in the ranking can be conditioned on the presence of another particular mark for the same extent. For example, a conditional dispreference can be set for an adjunct reading if an argument reading also exists. The extension does not address more global interactions, and is said (Forst et al. 2005) to be used mostly as a pre-filter to limit the readings disambiguated by a follow-on stochastic process.

2.6 A Slightly Different View

A slightly different view of preference–based parsing is that the business of a preference system is to work in tandem with a permissive syntactic grammar, to manipulate outcomes.

The difference focuses on the pragmatic role of preference in coercing the parser. In this light, the principles of section 2.1 are guidelines for desired outcomes, not bases for judging the goodness of a relationship or setting preference values. Instead, preference values should be set based on their effectiveness in isolating best parses. Also, in this light, the utility of a preference system lies not only in its contribution to accuracy, but also in its software-engineering convenience. These considerations led to the simpler, more practical scoring system of the RH overlay parser, described in section 4 below, in which contexted preference scores *CRS* can have one of only 3 values, -1, 0, or +1.

3 Background: The RH Parser

The RH parser consists of three major components, outlined below: the shallow parser, a mediating "locator" phase, and the overlay parser.

3.1 Shallow Parser

The shallow parser used, XIP, was developed by XRCE (Xerox Research Center Europe). It is actually a full parser, whose per-sentence output consists of a single tree of basic chunks, together with identifications of (sometimes alternative) typed dependences among the chunk heads (Ait-Mokhtar et al. 2002, Gala 2004). But because the XIP dependency analysis for English was not mature at the time that work on RH began, and because a classic parse tree annotated by syntactic functions is more convenient for some applications, we focused on the output chunks.

XIP is astonishingly fast, contributing very little to parse times (about 20%). It consists of the XIP processor, plus grammars for a number of languages. The grammar for a particular language consists of:

(a) a finite-state lexicon producing alternative part-of-speech and morphological analyses for each token, together with bit-expressed subcategorization and control features, and (some) semantic features,
(b) a substitutable tagger identifying the most probable part of speech for each token, and
(c) sequentially applied rule sets that extend and modify lexical information, disambiguate tags, identify named entities and other multiwords, and produce output chunks and inter-chunk head dependences (the latter not used in the hybrid).

Work on the hybrid parser has included large scale extensions to the XIP English rule sets.

3.2 Locator phase

The locator phase accumulates and analyses some of the shallow parser results to expedite the grammar and preference tests of the overlay parser.

For preference tests, for any input position, the positions of important leftward and rightward tokens are identified. These "important" tokens include commas, and leftward phrase heads that might serve as alternative attachment points.

Special attention is given to coordination, a constant source of inefficiency and inaccuracy for all parsers. To limit this problem, an input string is divided into spans ending at coordinating conjunctions, and the chunks following a span are examined to determine what kinds of coordination might be present in the span. For example, if a chunk following a span *Sp* is a noun phrase, and there are no verbs in the input following that noun phrase, only noun phrase coordination is considered within *Sp*. Also, with heuristic exceptions, the locator phase disallows searching for appositives within long sequences of noun and prepositional phrases ending with a coordinating conjunction.

3.3 Overlay Parser

The overlay parser uses a top-down grammar, expressed as a collection of ATN-like grammar networks. A recursive control mechanism traverses the grammar networks depth-first to build constituents. The labels on the grammar network arcs represent specialized categories, and are associated with tests that, if successful, either return a chunk or reinvoke the control to attempt to build a constituents for the category. The label-specific tests include both context-free tests, and tests taking into account the current context. For details see (Newman, 2007).

If an invocation of the control is successful, it returns an **output network** containing one or more paths, with each path representing an alternative sequence of immediate children of the constituent. An example output network is shown in figure 2. Each arc of the network references either a basic chunk, or a final state of a subordinate output network. Unlike the source grammar networks, the output networks do not contain cycles or converging arcs, so states represent unique paths.

The states contain both (a) information about material already encountered along the path, including syntactic functions and head features, and (b) a preference score for the path to that point. Thus the figure 2 network represents two alternative noun phrases, one represented by the path containing OS_0 and OS_1, and one containing OS_0, OS_1, and OS_2. State OS_2 contains the preference score (+1), because attaching a locative pp to a feature of the landscape is preferred.

From	To	Cat	Synfun	Reference
OS_o	OS_1	NP	HEAD	NPChunk (*The park*)
OS_1	OS_2	PP	NMOD	Final state of PP net for (*in Paris*)
States		**Score**	**Final?**	
OS_0		0	No	
OS_1		0	Yes	
OS_2		+1	Yes	

Figure 2. Output network for "The park in Paris"

Before an output network is returned from an invocation of the control mechanism, it is pruned to remove lower-scoring paths, and cached.

Output from the overlay parser is a single tree (Figure 1) derived from a highest scoring full path (i.e. final state) of a topmost output network. If there are several highest scoring paths, low attach considerations select a "best" one. The preference scores shown in Figure 1 in parentheses after the category labels are the scores at the succeeding states of the underlying output networks.

4 Preference System

Any path in an output network has the form:

$S_0, Ref_1, S_1, Ref_2, \ldots, S_{n-1}, Ref_n, S_n$

where S_i is a state, and Ref_i labels an arc, and references either a basic chunk, or a final state of another output network. A state S_i has total preference score $TPS(i)$ where:

- $TPS(0) = 0$
- $TPS(i),\ i>0 =$
  $TPS(i-1) + PS(Ref_i) + CRS(Ref_i)$
- $PS(Ref_i)$ is the non-contexted score of the constituent referenced by Ref_i, that is, the score at the referenced final state.
- $CRS(Ref_i)$ is the contexted score for Ref_i, in the context of the network category and the path ending at the previous state i-1.

For example, if Ref_i refers to a noun phrase considered a second object within a path, and the syntactic head along the path does not expect a second object, $CRS(Ref_i)$ might be (-1).

Each value CRS is limited to values in {-1, 0, +1}. Therefore, no judgment is needed to decide the degree to which a contexted reference is to be dispreferred or preferred. Also, if the desired parse result does not receive the highest overall score, it is relatively easy to trace the reason. Pruning (see below) can be disabled and all parses can be displayed, as in Figure 1, which shows the scores $TPS(i)$ in parentheses after the category labels for each Ref_i (with zero scores not shown). Then, if

$TPS(i) > (TPS(i-1) + PS(Ref_i))$

it is clear that the contexted reference is preferred. If multi-level contexted scoring were used instead, it would be necessary to determine whether the reference was preferred to exactly the right degree.

Test Block Type	Length Independent?	Indexed By
Coordinate	Y	Parent syncat
Subcat	Y	No index
FN1	Y	synfun
TAG1	Y	syncat
FN2	N	synfun
TAG2	N	syncat

Table 1. Preference Test Block Types

4.1 Preference test organization

To compute the contexted score *CRS* for a reference, relevant tests are applied until either (a) a score of -1 is obtained, which is used as *CRS* for the reference, or (b) the tests are exhausted. In the latter case, *CRS* is the higher of the values {0, +1} returned by any test.

For purposes of efficiency, the preference tests are divided into typed blocks, as shown in Table 1. At most one block of each type can be applied to a reference. Four of the blocks contain tests that are independent of referenced constituent length. They are applied at most once for a returned output network and the results are assumed for all paths. The other two blocks are length dependent.

Referring to Table 1, the length-independent coordinate tests are applied only to non-first siblings of coordinated constituents. The parent category indicates the type of constituents being coordinated and selects the appropriate test block. Tests in these blocks focus on the semantic consistency of a coordinated sibling with the first one.

Subcategorization tests are applied to prepositional, particle, and clausal dependents of the current head. These tests consist to a large extent of bit-vector implemented operations, comparing the expected dependent types of the head with lexical features of the prospective dependent. The tests are made somewhat more complex because of various exceptions, such as (a) temporal and locative phrases, and (b) the presence of a nearer potential head also expecting the dependent type.

The other test block types are selected and accessed either by the syntactic category or the syntactic function of the reference, depending on the focus of the test. The length-dependent tests include tests of noun-phrases within coordinations to determine whether post modifiers should be applied to the individual phrase or to the coordination as a whole.

The test blocks are expressed in procedural code. This has allowed the parser to be developed without advance prediction of the types of information needed for the tests, and also has contributed some efficiency. The blocks, usually short but occasionally long, generally consist of ordered (if-then-else) subtests.

4.2 Preference test scope

A contexted preference test can refer to material on three levels of the developing parse tree: (a) the syntactic category of the parent (available because of the top-down parser direction) (b) information about the current output network path, including head features, already-encountered syntactic functions, and a small collection of special-purpose information, and (c) information about the referenced constituent, specifically its head and a list of the immediately contained syntactic functions. The tests can also reference lookahead information furnished by the locator phase. This material is sufficient for most purposes. Limiting the kind of referenced information, particularly not permitting access to sibling constituents or deep elements of the referenced constituent, contributes to performance.

4.3 Pruning

Before an output network is completed, it is pruned to remove lower-scoring output network paths. Any path with the same length as another but with a lower score is pruned. Also, paths having other lengths but considerably lower preference scores than the best-scoring path are often pruned as well.

4.4 Usage Example

To illustrate how the simple scores and modular tests are used to detect and repair problems in the preference system, Figure 1 shows, as noted before, that the attachment of an object to the guessed verb "micromanaged" is dispreferred. In this case the probable reason is the lack of a transitive feature for the verb. To check this, we would look at the FN1 test block for OBJ and find that in fact the test assigns (-1) in this case. The required modification is best made by adding a transitive feature to guessed verbs.

But there is another problem here: the attachment of the pp "over people" is not given a positive preference. Checking the FN1 test block for

VMOD and the TAG1 test block for PP finds that there is in fact no subtest that prefers combinations of motion verbs and "over". While this doesn't cause trouble in the example, it could if there were a prior object in the verb phrase. A subtest or subcategorization feature could be added.

5 Training the Preference System

To obtain the preference system, an initial set of tests is identified, based primarily on subcategorization considerations, and then refined and extended based on manual "training" on large numbers of documents. Several problem situations result in changes to the system, besides random inspection of scores:

(a) the best parse identified is not the correct one, either because the correct parse is not the highest scoring one, or because another parse with the same score was considered "best" because of low-attach considerations.

(b) The best parse obtained is the correct one, but there are many other parses with the same score, suggesting a need for refinement, both to improve performance and to avoid errors in related circumstances when the correct parse does not "float" to the top.

(c) No parse is returned for an input, because of imposed space constraints, which indirectly control the amount of time that can be spent to obtain a parse.

In some cases the above problems can be solved by adjusting the base grammar, or by extending lexical information to obtain the appropriate preferences. For example, the preference scoring problems of Figure 1 can be corrected by adding subcategorization information, as described above.

In other cases, one or more modifications to the preference system are made, adding positive tests to better distinguish best parses, adding negative tests to disprefer incorrect parses, and/or refining existing tests to narrow or expand applicability.

Positive tests often just give credit to expected structures not previously considered to require recognition beyond acceptance by the grammar. Negative tests fall into many classes, such as:

(a) Tests for "ungrammatical" phenomena that should not be ruled out entirely by the grammar. These include lack of agreement, lack of expected punctuation, and presence of unexpected punctuation (such as a comma between a subject and a verb when there is no comma within the subject).

(b) Tests for probably incomplete constituents, based on the chunk types that follow them.

(c) Tests for unexpected arguments, except in some circumstances. For example, "benefactive" indirect objects ("John baked Mary a cake") are dispreferred if they are not in appropriate semantic classes.

Also, a large, complex collection of positive and negative tests, based on syntactic and semantic factors, are used to distinguish among coordinated and appositive readings, and among alternative attachments of appositives.

If the addition or modification of preference tests does not solve a particular problem, then some more basic changes can be made, such as the introduction of new semantic classes. And, in rare cases, new features are added to output network states in order to make properties of non-head constituents encountered along a path available for testing both further along the path and in the development of higher-level constituents. An example is the person and number of syntactic subjects, allowing contexted preference tests for finite verb phrases to check for subject consistency.

5.1 Relationship to "supervised" training

To illustrate the relationship between the above symbolic training method for preference scoring and corpus-based methods, perhaps the easiest way is to compare it to an adaptation (Collins and Roark, 2004) of the perceptron training method to the problem of obtaining a best parse (either directly, or for parse reranking), because the two methods are analogous in a number of ways.

The basic adapted perceptron training assumes a generator function producing parses for inputs. Each such parse is associated with a vector of feature values that express the number of times the feature appears in the input or parse. The features used are those identified by Roark (2001) for a top-down stochastic parser.

The training method obtains a weight vector W (initially 0) for the feature values, by iterating multiple times over pairs $\langle x_i, y_i \rangle$ where x_i is a training input, and y_i is the correct parse for x_i. For each pair, the best current parse z_i for x_i produced by the generator, with feature value vector $V(z_i)$, is selected based on the current value of $(W \cdot V(z_i))$. Then if $z_i \neq y_i$, W is incremented by $V(y_i)$, and decremented by $V(z_i)$. After training, the weights in W are divided by the number of training steps (# inputs * # iterations).

The method is analogous to the RH manual training process for preference in a number of ways. First, the features used were developed for suitability to a top-down parser, for example taking into account superordinate categories at several levels, some lexical information associated with non-head, left-side siblings of a node, and some right-hand lookahead. Although only one superordinate category is routinely used in RH preference tests, in order to allow caching of output networks for a category, the preference system allows for and occasionally uses the promotion of non-head features of nested constituents to provide similar capability.

Also, the feature weights obtained by the perceptron training method can be seen to focus on patterns that actually matter in distinguishing correct from incorrect parses, as does RH preference training. Intuitively, the difference is that while symbolic training for RH explicitly pinpoints patterns that distinguish among parses, the perceptron training method accomplishes something similar by postulating some more general features as negative or positive based on particular examples, but allowing the iterations over a large training set to filter out potentially indicative patterns that do not actually serve as such.

These analogies highlight the fact that preference system training, whether symbolic or corpus-based, is ultimately an empirical engineering exercise.

6 Some Experimental Results

Tables 2, 3, and 4 summarize some recent results as obtained by testing on Wall Street Journal section 23 of the Penn Treebank (Marcus et al. 1994). The RH results were obtained by about 8 weeks of manual training on the genre.

Table 2 compares speed and coverage for RH and Collins Model3 (Collins, 1999) run on the same CPU. The table also extrapolates the results to two other parsers, based on reported comparisons with Collins. One extrapolation is to a very fast stochastic parser by Sagae and Lavie (2005). The comparison indicates that the RH parser speed is close to that of the best contemporary parsers.

The second extrapolation is to the LFG XLE parser (Kaplan et al. 2004) for English, consisting of a highly developed symbolic parser and grammar, an OT-based preference component, and a stochastic back end to select among remaining alternative parser outputs. Two sets of values are given for XLE, one obtained using the full English grammar, and one obtained using a reduced grammar ignoring less-frequently applicable rules. The extrapolation indicates that the coverage of RH is quite good for a symbolic parser with limited training on an idiom.

While the most important factor in RH parser speed is the enormous speed of the shallow parser, the preference and pruning approach of the overlay parser make contributions to both speed and coverage. This can be seen in Table 2 by the difference between RH parser results with and without pruning. Pruning increases coverage because without it more parses exceed imposed resource limits.

Table 3 compares accuracy. The values for Collins and Sagae/Lavie are based on comparison with treebank data for the entire section 23. However, because RH does not produce treebank-style tags, the RH values are based only on a random

	Time	No full parse
Sagae/Lavie	***~ 4 min***	*1.1%*
RH Prune	5 min 14 sec	10.8%
RH NoPrune	7 min 5 sec	13.9 %
Collins m3	16 min	**.6%**
XLE reduced	*~24 minutes*	unknown
XLE full	*~80 minutes*	~21%

Table 2. Speeds and *Extrapolated speeds*

	Fully accurate	F-score	Avg cross bkts
Sagae/Lavie	unknwn	86%	unknwn
Collins Lbl	33.6%	88.2%	1.05
CollinsNoLbl	35.4%	**89.4 %**	1.05
RH NoLbl	**46%**	86 %	**.59**

Table 3. Accuracy Comparison

	Average	Median
RH Base	137.10	11
RH Pref	**5.04**	**2**

Table 4. Highest Scoring Parses per Input

100-sentence sample from section 23, and compared using a different unlabeled bracketing standard. For details see Newman (2007). For non-parsed sentences the chunks are bracketed. Accuracy is not extrapolated to XLE because available measurements give f-scores (all $\leq$ 80%) for dependency relations rather than for bracketed constituents.

As a partial indication of the role and effectiveness of the RH preference system, if non-parsed sentences are ignored, the percentage of fully accurate bracketings shown in Table 3 rises to approximately 46/89 = 51.6% (it is actually larger because coverage is higher on the 100-sentence sample). As further indication, Table 4 compares, for section 23, the average and median number of parses per sentence obtained by the base grammar alone (RH Base), and the base grammar plus the preference system (RH Pref).[3] The table demonstrates that the preference system is a crucial parser component. Also, the median of 2 parses per sentence obtained using the preference system explains why the fallback low-attach strategy is successful in many cases.

7 Summary and Directions

The primary contribution of this work is in demonstrating the feasibility of a vastly simplified symbolic preference scoring method. The preference scores assigned are neither "principle-based", nor "ad-hoc", but explicitly engineered to facilitate the management of undesirable interactions in the grammar and in the preference system itself. Restricting individual contexted scores to {-1, 0, +1} addresses the problems of multi-level contexted scoring discussed in Section 2, as follows:

- No abstract judgment is required to assign a value to a preference or dispreference.
- Information deficiencies contribute only small dispreferences, so they can often be overcome by preferences.
- Compensating for interactions that are foreseen does not require searching the rules to find necessary compensating values.
- For unforeseen interactions discovered when reviewing parser results, the simplified preference scores facilitate finding the sources of the problems and potential methods of solving them.

This approach to symbolic preference has facilitated development and maintenance of the RH parser, and has enabled the production of results with a speed and accuracy comparable to the best stochastic parsers, even with limited training on an idiom.

An interesting question is why this very simple approach does not seem to have been used previously. Part of the answer may lie in the lack of explicit recognition that symbolic preference scoring is ultimately an engineering problem, and is only indirectly based on cognitive principles or approximations to frequencies of particular relationships.

Ongoing development of the RH preference system includes continuing refinement based on "manual" training, and continuing expansion of the set of semantic features used as the parser is applied to new domains. Additional development will also include more encoding of, and attention to, the expected semantic features of arguments. Experiments are also planned to examine the accuracy/performance tradeoffs of using additional context information in the preference tests.

[3] The values are somewhat inflated because they include duplicate parses, which have not yet been entirely eliminated.

References

Salah Aït-Mokhtar, Jean-Pierre Chanod, Claude Roux. 2002. Robustness beyond shallowness: incremental deep parsing, *Natural Language Engineering* 8:121-144, Cambridge University Press.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations, In *Proc of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, Barcelona

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

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proc of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, Barcelona.

Martin Forst, Jonas Kuhn, and Christian Rohrer. 2005. Corpus-based Learning of OT Constraint Rankings for Large-scale LFG Grammars. In *Proc of the*

LFG05 Conference, Bergen. Available at http://cslipublications.stanford.edu/LFG/10/lfg05.pdf

Anette Frank., Tracy H. King, Jonas Kuhn, and John Maxwell. 1998. Optimality theory style constraint ranking in large-scale LFG grammars. In M. Butt and T. H. King, Eds. *Proc of the Third LFG Conference.* Available http://csli-publications.stanford.edu/LFG3/ Revised version in Peter Sells, ed. *Formal and Theoretical Issues in Optimality Theoretic Syntax*, CSLI Publications, 2001.

Nuria Gala. 2004. Using a robust parser grammar to automatically generate UNL graphs. *In Proc Workshop on Robust Methods for Natural Language Data at COLING'04,* Geneva

Donald Hindle and Mats Rooth. 1993. Structural Ambiguity and Lexical Relations. *Computational Linguistics*, 19:1,103–120.

Jerry R. Hobbs and John Bear. 1990. Two Principles of Parse Preference. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90),* Helsinki, Finland, August 1990.

Jerry. R. Hobbs, Douglas E. Appelt, John Bear, Mabry Tyson, and David Magerman. 1992. Robust processing of real-world natural language texts. In Paul S. Jacobs, ed., *Text-Based Intelligent Systems: Current Research and Practice in Information Extraction and Retrieval.* Lawrence Erlbaum, New Jersey, 1992.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02),* Philadelphia, July 2002, pp. 136-143.

Ronald M. Kaplan, Stephan Riezler, Tracy H. King, John T. Maxwell, Alex Vasserman. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proc HLT/NAACL'04,* Boston, MA.

Chao-Lin Liu, Jing-Shin Chang, and Keh-Yi Su. 1990. The Semantic Score Approach to the Disambiguation of PP Attachment Problem. In *Proceedings of ROCLING-90*, Taiwan

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn treebank**.** *Computational Linguistics***,** 19(2) pp.313--330**.**

Michael C. McCord. 1993. Heuristics for Broad-Coverage Natural Language Parsing. *Proceedings of the workshop on Human Language Technology 1993,* Princeton, NJ

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms**.** In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP 2005),* Vancouver.

Paula S. Newman. 2007. RH: A Retro-Hybrid Parser. In *Short Papers of Proceedings of NAACL/HLT 2007*, Rochester NY

Alan Prince and Paul Smolensky (1993). Optimality Theory: Constraint interaction in generative grammar, Rutgers University Center for Cognitive Science, New Brunswick, NJ. Report RuCCS-TR-2. [Reprinted in John J McCarthy, ed., *Optimality Theory in Phonology: A Book of Readings*, Blackwell (2003).]

Philip Resnik and Marti Hearst. 1993. *Structural Ambiguity and Conceptual Relations,* in Proc. of 1st Workshop on Very Large Corpora, 1993.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. In *Computational Linguistics*, 27(2), pages 249-276.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proc. 9th Int'l Workshop on Parsing Technologies*. Vancouver

Kristina Toutanova, Christopher D. Manning, Dan Flickinger, and Stephan Oepen. 2005. Stochastic HPSG Parse Disambiguation using the Redwoods Corpus. *Research in Language and Computation* 2005.

Yorick A. Wilks. 1975**.** An Intelligent Analyzer and Understander of English**.** *Communications of the ACM* 18(5), pp.264-274

William Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM* 13(10), pp.591-606

XLE Online Documentation. 2006. Available at http://www2.parc.com/isl/groups/nltt/xle/doc/xle.html#SEC15

Kiyoshi Yamabana, Shin'ichiro Kamei and Kazunori Muraki. On Representation of Preference Scores. In Proceedings of *The Fifth International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-93)*, Kyoto, pp. 92-101