1 Methodology

1.1 Experimental Setup

Experiments were designed and executed to test the relative merit of a large collection of features for parse disambiguation. These experiments were all conducted on the WikiWoods portion of the Redwoods 7th Growth treebank. This is a collection of text from Wikipedia on topics related to computational linguistics, with an average sentence length of ??. Attention was restricted to those sentences which received at least two analyses, at least one of which was judged correct. This resulted in a set of 8607 sentences, with a random choice baseline accuracy of 8.1%.

The trees underwent a minor surgery before participating in the experiments: the labels of the preterminals were changed from the ERG lexeme names to the names of their corresponding lexical types[1].

Each feature set was tested in combination with a fixed baseline set of features (see below), rather than in isolation. The purpose of this setup was to determine not whether the additional features had any information content on their own, but whether they offered new information beyond that contained in the simple baseline features. For each configuration of features, 10-fold cross-validated evaluations were conducted at each of 50 different values of the regularization variance meta-parameter (ranging from 0.001 to 1000)[2]. For some experiments, a graph of performance vs. variance is given.

1.2 Evaluation Metric

The present experiments followed (Toutanova et al. 2002) in using exact match accuracy as the evaluation metric. The model is judged correct on a sentence if it selects a tree that was judged correct by the treebanker. In cases where several trees tie under the model, the score is reported as the fraction of tied trees that were judged correct.

Many other evaluation metrics have been proposed in the literature, including the PARSEVAL metric, dependency-based metrics, and end-to-end application metrics. These all have merits, but were not included in this study.

1.3 Baseline

The baseline feature set had a feature for every local subtree of depth one. That is, for every node in a tree, a feature is generated describing the labels on that node and its daughters. This feature set scored 40.3% at its best variance parameter, and all of its features are included in all subsequent experiments.

[1] This is done for increased comparability to Toutanova et al. 2005. The surgery does in fact discard some information; however it appears to improve performance, at least for some feature types. See also the lexeme-name lexicalization experiment.

[2] Due to details in the formulation of the optimization, variance values may not be comparable between publications

1.4 Syntactic Features and Decorations

Several of the feature types (including the baseline feature set) are essentially cookie-cutters which extract variously shaped pieces of the input syntax tree. These interact in a special way with another class of feature types, dubbed *decorations*. A decoration is an additional label that is computed for each node in the syntax tree (e.g. the value of the HEAD feature in HPSG). When a particular decoration and a particular syntactic feature type are both enabled in an experiment, the syntactic feature is instantiated both on the basic tree (with its original labels) and on a revised tree whose labels are tuple (original label, decoration). Optionally (only when noted in the experiments section below), the syntactic features can be instantiated a third time on another revised tree whose labels are only the decorations.

1.5 Basic Experiments

A basic experiment was conducted for each feature type, to determine the performance offered by the feature type in combination with the baseline features. Accuracy is reported at the optimal variance parameter.

1.6 Compound Experiments

Additionally, certain combinations of several feature types were selected for compound experiments. In these experiments, several different feature types were active simultaneously in addition to the baseline features.

2 Experiments and Results

Below is a description of each experiment performed. Some of the salient details are summarized in a table at the end of this section. Recall that the baseline feature set scored 40.3%. Some features improved upon this figure substantially; others did not.

Note that wth the exception of the Compound Experiments section, the experiments described below are all independent; that is, they do not build on each other.

The following diagram gives a typical candidate analysis; for coherence, feature examples will be given relative to this analysis whenever possible.

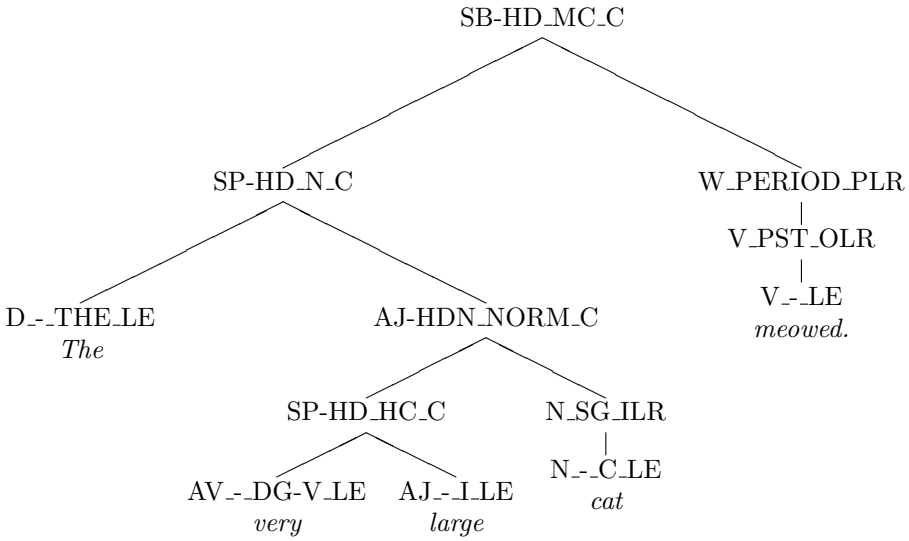


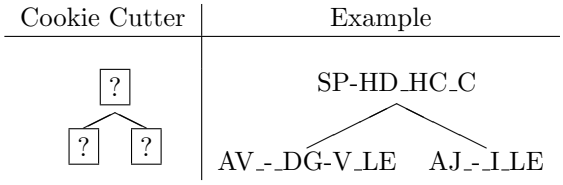
Table 1 summarizes the results of the individual experiments.

2.1 Syntactic Features

Recall that the *syntactic* feature templates are like cookie-cutters that select various shapes from the input syntax tree. In the discussion below, the basic cookie-cutter shape and an example from the tree above will be given.

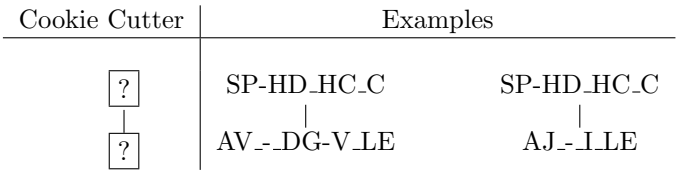
2.1.1 Baseline

The example baseline feature configuration below dominates the substring *very large* in our sample tree.



2.1.2 Fragments

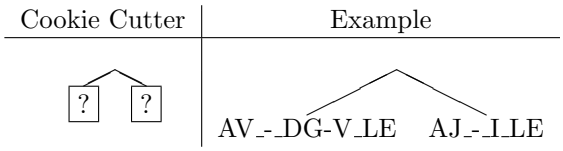
The fragment features are essentially equivalence classes on the baseline features obtained by ignoring all but one daughter in a rule. The example Baseline feature given above splits into these two features:



These features scored 41.19%, which represents a fairly minor improvement of less than 1% over the baseline.

2.1.3 Unparenting

Similar to the fragment features, these are equivalence classes on the baseline features obtained by ignoring the parent in a rule. The example baseline feature from above would give rise to the unparented feature:



The score for the unparenting experiment was 40.54%, which may not be a significant improvement on the baseline.

2.1.4 Grandparenting

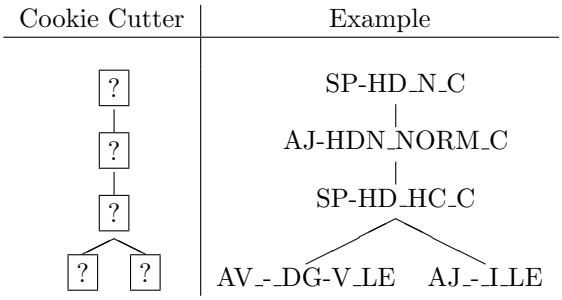
This set of experiments investigated the merit of adorning the baseline feature configurations with data about more distant ancestors. Whereas the baseline features were of the type [PARENT DTR1 DTR2], features for single-level grandparenting (GP[1]) are of the type [GRANDPARENT PARENT DTR1 DTR2], and features for double-level grandparenting (GP[2]) are of the type [GREATGRANDPARENT GRANDPARENT PARENT DTR1 DTR2].

GP[1] achieved a score of 43.22% – nearly a 3 percentage point improvement over the baseline configuration.

GP[2] achieved a score of 44.66%, a further improvement of 1.4% over GP[1].

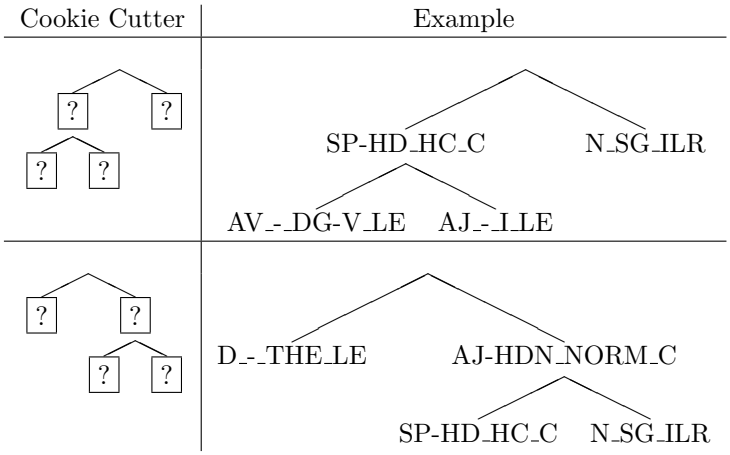
GP[3] scored 44.95% and GP[4] scored 44.62%, suggesting that grandparenting beyond two or three levels is of little use.

Note that GP[1] includes all of the baseline features, GP[2] includes all of the GP[1] features, etc.



2.1.5 Uncles

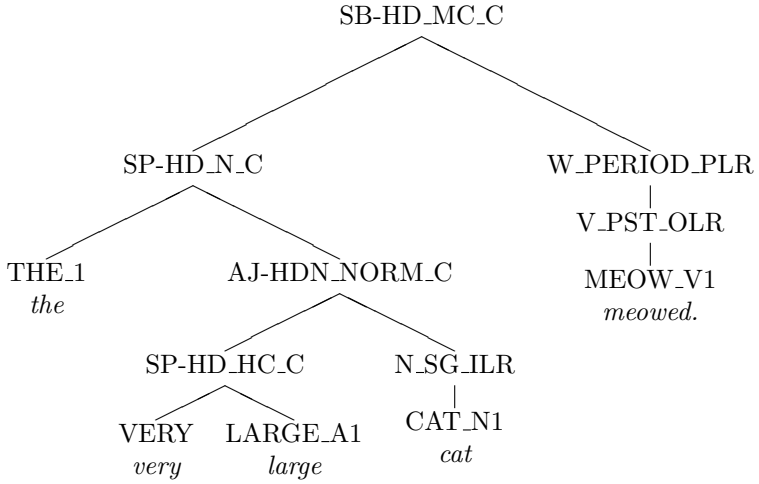
These features adorn the baseline features with information about the sibling(s) of the parent node, if any, e.g. [left-uncle:UNCLE PARENT DTR1 DTR2] or [right-uncle:UNCLE PARENT DTR1 DTR2]. Note that in the trees under consideration, there are no rules of arity greater than 2, so there is never more than 1 uncle.



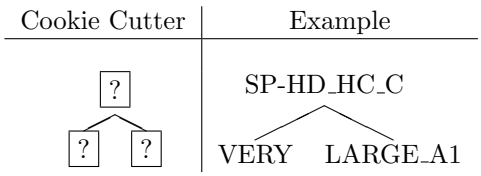
This information proved about as useful as single level grandparenting, scoring a healthy 43.38%.

2.1.6 Lexeme Names

Whereas in other experiments the tree preterminals are the names of the lexical types of the lexemes, this experiment built baseline features using the lexeme names instead as preterminals. For our example analysis, the modified syntax tree is:



This yielded a slight improvement over the unmodified baseline, with a score of 40.82%.



2.1.7 Balance Features

These features attempt to describe whether the subtree under a particular binary node is heavier on the left, heavier on the right, or well-balanced.

Several different feature types were included in this experiment, including a binned difference between the heights of the left and right subtrees; the binned height of the left subtree; and the binned height of the right subtree.

Each feature is included adorned only by the local node label, and also adorned with that node's parent's label. Below is a sample feature from our tree, comparing the subtrees under SP-HD_N_C:

$$[SB - HD\_MC\_C, SP - HD\_N\_C, \text{ height difference: } 2]$$

The hope was that these features would allow the model to capture preferences for left-branching or right-branching structures for particular construction types (e.g. PP attachment). The somewhat disappointing score of 40.80%, however, indicates that more research is required if such preferences are to be profitably exploited.

2.1.8 Surface N-grams

The features investigated in this experiment depart from the rule-configuration-based features of the above experiments. The cookie-cutter shapes for this and the remaining 3 syntactic feature templates are more abstract, and will not be depicted. Instead, the features are N-grams over the string of preterminals in the tree (i.e. the lexeme nodes), represented by their lexical types. Artificial start and stop types were added to the string.

The string of surface lexical types for our example tree is:

[*start*, D_-_THE_LE, AV_-_DG-V_LE, AJ_-_I_LE, N_-_C_LE, V_-_LE, *end*]

Here are a few example N-grams:

Bigrams	Trigrams
[*start*, D_-_THE_LE]	[N_-_C_LE, V_-_LE, *end*]
[AJ_-_I_LE, N_-_C_LE]	

Bigrams yielded a 41.02% accuracy, while trigrams yielded a 40.97% accuracy.

2.1.9 Leaf Projection Path N-grams

These features follow Toutanova's "Leaf Projection Path" paper. For each lexeme, the features are a tuple (LEXEME, N-GRAM) where N-GRAM is an N-gram on the string of parents leading from the lexeme in question to the root of the tree.

Here is a sample trigram built from our example tree:

(CAT_N1, [SP-HD_N_C, SB-HD_MC_C, *end*])

Unigrams scored 41.55%, bigrams yielded 41.63%, and trigrams achieved 41.50%. All three are clear improvements over the baseline, but the differences between the N-gram sizes are small.

2.1.10 Syntactic Dependencies

These features represent a simplification of the tree structure into a series of dependencies between the words in the sentence, governed by the headedness of the rules.

For each binary node in the tree, the lexical head of the head daughter and the lexical head of the nonhead daughter are determined, and the rule name is used as a description of the relationship holding between these two words. The lexical types of the words in question are also included. Here are two syntactic dependencies from our example tree:

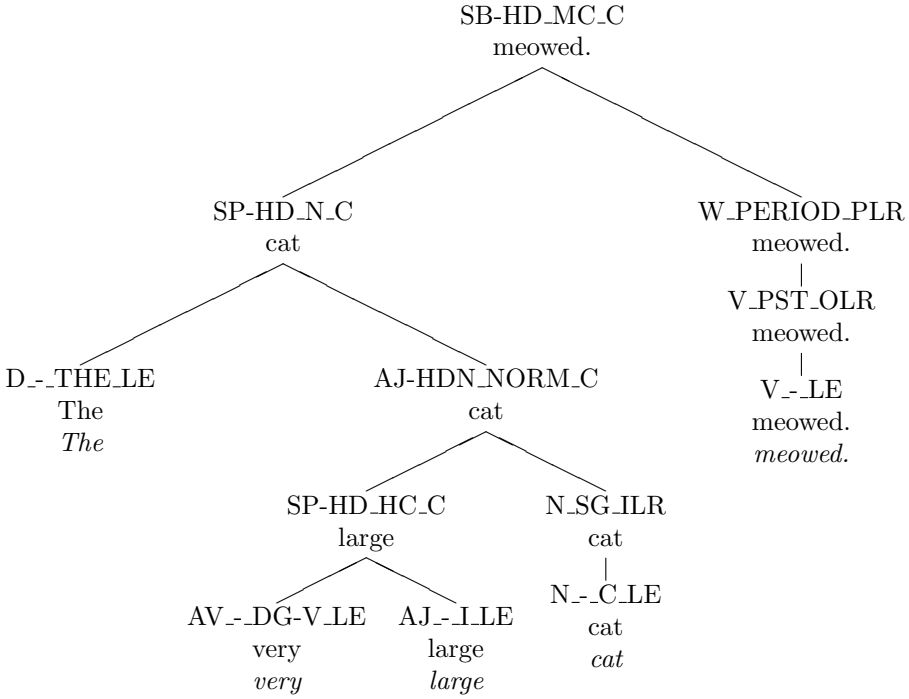
SP-HD_N_C (cat : N_-_C_LE, The : D_-_THE_LE)
SB-HD_MC_C (meowed. : V_-_LE, cat : N_-_C_LE)
The score for the syntactic dependency features was 42.93%.

2.2 Lexicalization Decorations

A number of experiments were conducted with decoration features derived from so-called *lexicalizations*. These features decorate each node in the syntax tree with information gleaned from the lexical head of the constituent dominated by that node. Several different types of information were tried:

2.2.1 Form Lexicalization

The decoration is the surface form of the lexical head. Our decorated example tree looks like:

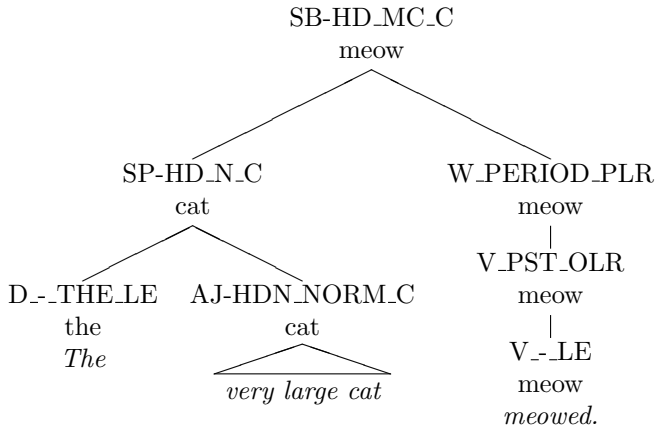


The score for baseline style features extracted from this decorated tree was 43.78%.

For the sake of brevity, the descriptions of the remaining lexicalization experiments will show abridged trees.

2.2.2 Stem Lexicalization

The decoration is the value of the ORTH feature of the (uninflected) lexical entry representing the lexical head. The abridged decorated example tree is:

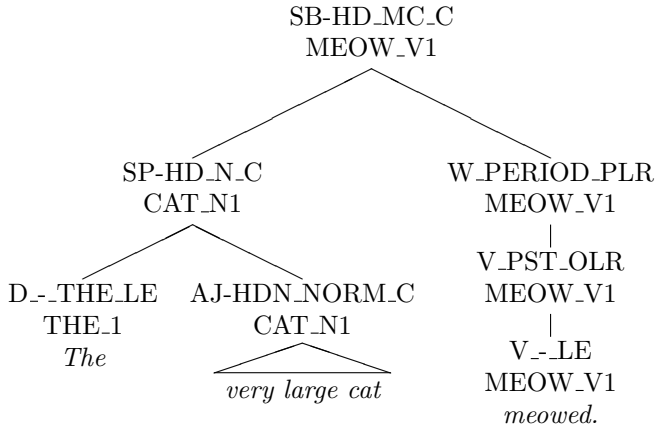


The score was 43.85%.

While similar to the form lexicalization, stem lexicalization captures a slight generalization in that features are not differentiated based on inflections (e.g. singular vs plural markers). Although one might expect that discarding information in this manner would yield a degradation in performance, empirically we do not see that effect for this experiment.

2.2.3 Lexeme Name Lexicalization

The decoration is the (grammar-internal) name of the lexical entry representing the lexical head. The abridged decorated example tree is:



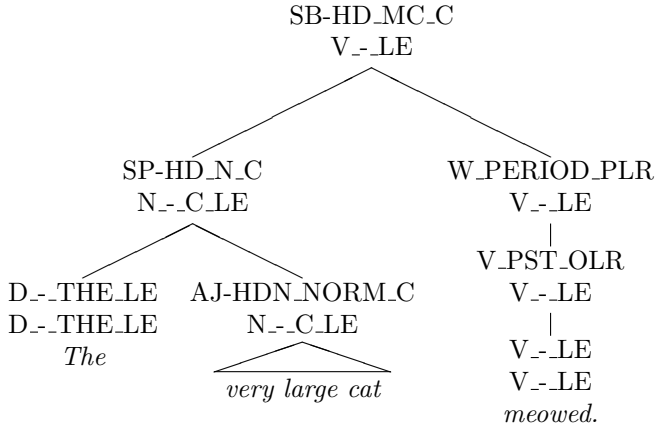
The score was 44.15%.

This decoration is again quite similar to stem lexicalization and form lexicalization. As with stem lexicalization, inflectional information (e.g. plural markers) is discarded. However, we add back in a different bit of information, namely the distinction between different lexical entries with the same orthography. This differentiates words that look the same but have different syntactic properties (part of speech, argument structure, etc.), such as "make" in "to make a mistake" vs "to make things right" vs "to make the problem go away".

The ERG has at least 10 different verbal lexemes for "make" as well as several nonverbal ones!

2.2.4 Lexical Type Lexicalization

The decoration is the name of the type of the lexical entry representing the lexical head. The abridged decorated example tree is:

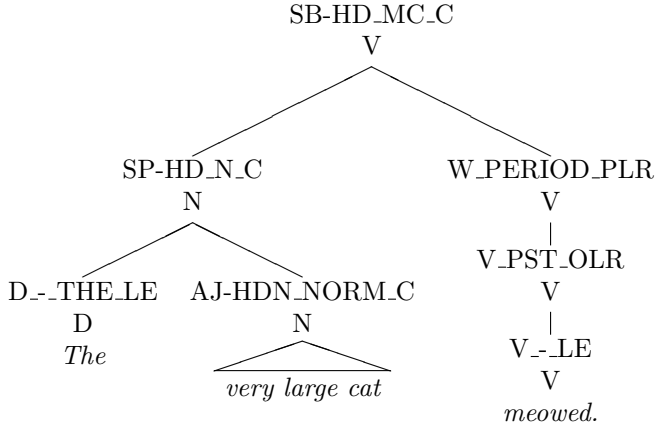


The score was 41.87%.

Here, the information that uniquely identifies a surface word is discarded, but (most of) the distinctions regarding syntactic properties of a word are maintained. Thus, the the clauses "to make the problem go away" and "to let the problem go away" would be decorated identically under this scheme.

2.2.5 Simplified Lexical Type Lexicalization

The decoration is the first letter of the name of the type of the lexical entry representing the lexical head. The abridged decorated example tree is:

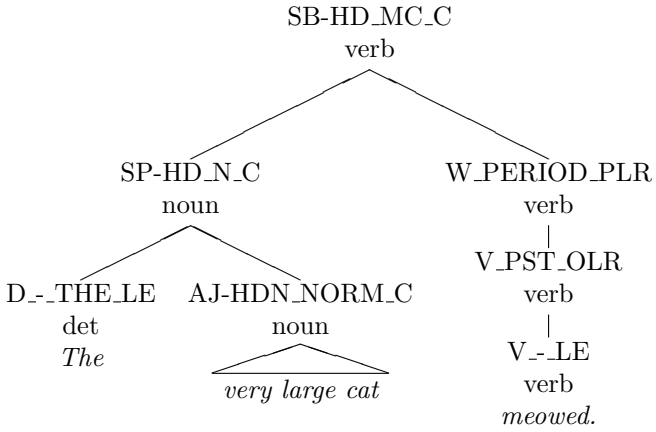


The score was 40.58%.

This decoration discards even more information, essentially retaining only the part of speech of the lexical head. Since that part of speech can usually be inferred from the name of the rule dominating a phrase, it is perhaps unsurprising that this decoration yielded essentially no improvement over the baseline.

2.2.6 Head Lexicalization

The decoration is the name of the type of the value of the HEAD feature of the lexical entry representing the lexical head. The abridged decorated example tree is:



The score was 40.90%.

This decoration is almost identical to the previous one; it can differ in certain coordination constructions, but the details are boring and unimportant since neither model performed well!

2.3 Other Decorations

2.3.1 Punctuation Decorations

Three types of punctuation were considered. In the first experiment, the decoration used was the sentence final punctuation (i.e. the same for all nodes of all candidates for a particular sentence). The score was 40.44%.

For the next experiment, the decoration was the punctuation mark at the beginning of the string dominated by the node (typically none present; sometimes an open quote or a parenthesis). The score was 40.41%.

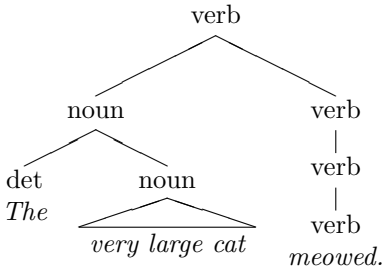
The final experiment of this type used the punctuation mark at the end of the string dominated by the node as the decoration (typically none present; sometimes a comma, period, close quote, close parenthesis, etc). The score was 41.29%.

2.3.2 Local Head Decoration

This decoration was the name of the type of the value of the HEAD feature of the AVM corresponding to the local node. In almost all cases, this is identical to the head lexicalization decoration from above. The differences are rare, subtle and, it turns out, unimportant. The score was 40.50%.

2.4 "Pure" Decorations

In these experiments, each of the decoration types above was used as described, with the addition of baseline-style features extracted from a *third* tree, whose nodes eschew their original labels and are clad *only* with the decorations, e.g. this "pure" head lexicalization tree:



These experiments generally outperformed their "unpure" counterparts, but only by very small amounts (less than 0.5%).

2.5 Semantic Features

In addition to a syntax tree, the ERG produces an *MRS*, which is essentially a first-order logic representation of the meaning of the sentence. The following two experiments used features derived from the MRS output rather than the syntax tree. The MRS for our example analysis can be approximated by the following conjunction of predications:[3]

$$\text{cat}_n(x_2) \wedge \text{meow}_v(e_1, x_2) \wedge \text{very}_{\text{deg}}(e_7, e_8) \wedge \text{large}_a(e_8, x_2)$$

Note that some aspects of the meaning of the sentence (e.g. the cardinality of nouns and the tense of verbs) are represented as internal properties of the logical variables rather than as part of the predicate logic. For the purposes of these experiments, we will ignore those properties.

[3]Actually, MRS represents the meaning of a sentence as a multiset of individual predications combined with constraints on how to assemble them into a logical formula. In sentences with multiple quantifiers, there can be multiple non-equivalent logical formulas described by the MRS, and the meaning of the sentence can crucially depend on which formula was intended by the speaker. Since the Redwoods corpus does not annotate a prefered logical formula, these details are irrelevant to this investigation.

2.5.1 MRS Variable-centric

The features are (unordered) 2-sets and 3-sets of (predicate, argument-slot) tuples which refer to the same logical variable. An example such 3-set is:

$$\{(\text{large}_a, ARG1), (\text{cat}_n, ARG0), (\text{meow}_v, ARG1)\}$$

The score was 41.36%.

2.5.2 MRS Predication-centric

The features are individual predications, with the argument logical variables replaced by the names of the predications of which they are the ARG0. The ARG0 slot of the original predication is omitted to reduce redundancy. Two examples are:

$$\text{meow}_v(\text{cat}_n)$$

$$\text{very}_{\text{deg}}(\text{large}_a)$$

The score was 42.64%.

2.6 Compound Experiments Results

Naively, one might expect that if features X delivered an x% boost over the baseline and features Y delivered a y% boost over the baseline, then a model containing both X and Y ought to deliver an $x + y$% boost over the baseline. In practice, this is rarely the case. For most of the compound experiments performed herein, the combined model outperformed both constituents, but fell significantly short of the hoped for arithmetic gain. It is possible (though uncommon) for the combination model to deliver a boost in *excess* of the sum of the constituents, or to perform worse than the constituent models.

One problem is that the information contained in the two feature sets can overlap. Another issue that can arise is that the generalizations derived from the two sets independently can contradict each other. There is no general rule for predicting the performance of a compound model; the only way to find out is to conduct an experiment.

Due to adverse combinatorics, testing all combinations independently is infeasible. Therefore, it is necessary to fall back on intuition in selecting which combinations of features to test. The combination experiments reported below were hand-designed, roughly according to the following plan: the experiments above were divided into classes in such a way that the experiments within a class could intuitively be expected to capture similar information (e.g. stem lexicalization and lexme name lexicalization), and experiments in different classes were expected to capture unrelated information (e.g. GP[2] versus MRS variable-centric). Each combination experiment consists of selecting a single good performer from each of a handful of classes, in a more-or-less ad-hoc way; in fact, the classes were not even made formal themselves.

Experiment	Score (%)	Boost (%)	Relevant Feature Types	Relevant Feature Instances
baseline	40.37	-	44k	56M
syntactic				
unparenting	40.54	0.17	63k	104M
fragments	41.19	0.82	50k	99M
GP[1]	43.22	2.85	188k	126M
GP[2]	44.66	4.29	616k	214M
GP[3]	44.95	4.58	1.7M	316M
GP[4]	44.62	4.25	3.8M	424M
uncles	43.38	3.01	306k	116M
dependencies	42.93	2.56	365k	79M
balance	40.80	0.43	77k	208M
lexeme names	40.82	0.45	77k	55M
n-grams				
surface bigrams	41.02	0.65	75k	71M
surface trigrams	40.97	0.60	228k	90M
leaf path projection unigrams	41.55	1.18	105k	237M
leaf path projection bigrams	41.63	1.26	380k	320M
leaf path projection trigrams	41.50	1.13	1.5M	678M
decorations				
form lexicalization	43.78	3.41	627k	111M
stem lexicalization	43.85	3.48	544k	112M
lexeme name lexicalization	44.15	3.78	622k	114M
lexical type lexicalization	41.87	1.50	301k	115M
simple type lexicalization	40.58	0.21	102k	113M
head lexicalization	40.90	0.53	110k	113M
local head	40.50	0.13	97k	113M
punctuation (sentence final)	40.44	0.07	86k	112M
punctuation (constituent initial)	40.41	0.04	92k	112M
punctuation (constituent final)	41.29	0.92	120k	118M
MRS-based				
predications	42.64	2.27	283k	73M
variables	41.36	0.99	169k	71M

Table 1: Scores of Individual Experiments

Relevant features are features whose values are nonconstant for at least one treebank sentence. Relevant feature instances are feature values which are different from the most common value observed value for that feature on the sentence in which they appear.

Note that when combining a *decoration* feature with a *syntax* feature, the feature set of the combination experiment is not simply the union of the basic feature sets. Rather, each syntactic feature template is allowed to extract features both from the basic syntax tree and from each decorated syntax tree.

Table 2 shows the components and results of each compound experiment.

Feature Components	Score
baseline	40.37
fragments + surface bigrams + LPP unigrams	42.95
fragments + surface bigrams + LPP unigrams + balance	42.66
lexeme name lexicalization + GP[1]	45.48
lexeme name lexicalization + GP[2]	45.87
lexeme name lexicalization + lexical type lexicalization	44.28
lexeme name lexicalization + local head + punct [CF]	44.95
dependencies + GP[1]	44.13
dependencies + GP[2]	44.83
dependencies + GP[3]	44.91
dependencies + uncles	44.11
uncles + GP[1]	43.72
uncles + GP[2]	45.14
uncles + GP[3]	45.09
The remaining experiments all include MRS predications *and* MRS variables *features:*	
MRS predications + MRS variables	42.72
+ local head	43.29
+ local head + GP[1]	44.70
+ local head + GP[2]	45.77
+ local head + GP[3]	45.70
+ local head + balance	43.60
+ lexeme name lexicalization	44.76
+ lexeme name lexicalization + GP[2]	46.11
+ lexeme name lexicalization + fragments + surface bigrams + LPP unigrams + GP[2]	47.50
+ lexeme name lexicalization + fragments + surface bigrams + LPP unigrams + uncles + dependencies + punct [CF] + local head + GP[3]	???

Table 2: Scores of Compound Experiments