Probabilistic Unification Grammars

Tony C. Smith and John G. Cleary

Department of Computer Science, University of Waikato, Hamilton, New Zealand
{tcs,jcleary}@waikato.ac.NZ; phone: +64 (7) 838–4453; fax: +64 (7) 838–4155

Abstract
Recent research has shown that unification grammars can be adapted to incorporate statistical information, thus preserving the processing benefits of stochastic context-free grammars while offering an efficient mechanism for handling dependencies. While complexity studies show that a probabilistic unification grammar achieves an appropriately lower entropy estimate than an equivalent PCFG, the problem of parameter estimation prevents results from reflecting the empirical distribution. This paper describes how a PUG can be implemented as a Prolog DCG annotated with weights, and how the weights can be interpretted to give accurate entropy estimates. An algorithm for learning correct weights is provided, along with results from some complexity analyses.

Introduction

Probabilistic context-free grammars are a popular formalism for modeling language because they preserve the simple CFG framework for characterising syntactic structure while incorporating the predictive benefits of statistical methods[3]. However, their lack of a mechanism for handling constraint dependencies means that rules must be multiplied out for each feature.

Unification grammars, on the other hand, have a clean denotational semantics for representing grammatical dependencies, and can be implemented efficiently as logic programs, such as Prolog DCGs[6]. Additional feature contraints are accommodated by simply adding variables to the appropriate clauses. A unification grammar can also be extended to include stochastic information and, in principle, thereby achieve better entropy estimates than an equivalent PCFG.

Two major problems arise when one attempts to include probabilities in a unification grammar. First, it is not clear where statistical information should be assigned—should it reflect estimated rule frequency (as it does for PCFGs), the probability of feature structures[5] or type hierarchies[2], or some combination of these. Second, it is difficult to formulate a coherent probabilistic basis as to how this information should be interpretted.

There have been several recent attempts at creating stochastic unification grammars. Eisele [4] proposes assigning probabilities to the Horn clauses of a logic program, and estimating grades of grammaticality by calculating probabilities for SLD proof trees. The probability of an expression is estimated as the sum of the product of probabilities for clauses occurring along the path of each successful proof. Eisele follows the PCFG principle of maintaining the probability mass across grammar-rule analogues—so that when the probability of a successful clause is increased by some amount, the probabilities of unsuccessful alternatives are uniformly decreased to maintain balance. Redistribution of the probability mass in this manner results in weights which do not take into account feature values when a clause succeeds, and entropy estimations consequently fail to reflect the empirical distribution.

Brew [2] argues that a PUG should model the accessibility relationships between nodes of the tree, and describes a means for probabilistic interpretation of typed-feature structures. The right hand side of each rule is a multiset of maximal and non-maximal types, where non-maximal leaves are successively expanded beginning from a non-maximal start node. Each expansion is formulated from a set of possible introduction relationships defined for a given non-maximal type. Each introduction relationship is associated with a probability, and the probability of an expression is the product of the probabilities for refinements used in the expansion. Because the same probability distribution is used for a given type whenever it is expanded, irrespective of the enclosing context, entropy estimates are inaccurate. An additional distortion occurs because part of the probability mass is spent on impossible feature structures.

This paper outlines a PUG formalism in which rule probabilities are interpretted differently depending upon feature values known when a set of rule analogues is invoked. We begin with a description of PCFGs, and the combinatorial explosion that results when they are adjusted to maintain feature agreement. We show how this situation can be avoided by using a PUG, and outline a PUG formalism implemented as a Prolog DCG annotated with weights. We explain how the weights can be used to give accurate entropy estimates, and outline an algorithm for estimating them correctly. Results from applying the learning algorithm to a sample corpus are also included. Finally, we describe how the formalism can be extended so that the grammar will converge (in the limit) to the optimum probabilities that would be given to a fully expanded PCFG.

Probabilistic context-free grammars

A PCFG is a four-tuple (V, N, N_1, R), where V is a set of terminal symbols, N is a set of nonterminals, N_1 is the start symbol and R is the set of production rules of the form $\beta_j N_i^k \Rightarrow \zeta_j$ where ζ_j is the rewrite of N_i^k and β_j is the probability of N_i^k (the k-th production for N_i). The probability of any sentence covered by the PCFG is given as the product of the probabilities for the rules used in its parse (or, in the case of an ambiguous grammar, the sum of the probabilities for each possible parse). Two essential properties of a PCFG are that the probabilities for all sentences generable from the model sum to one, and that the probabilities for all production rules for each nonterminal also sum to one.

Modeling the empirical distribution

Consider the following twelve sentence training corpus:

a dog chases	the dog chases	the dogs chase
a cat chases	the cat chases	the cats chase
a dog chased	the dog chased	the dogs chased
a cat chased	the cat chased	the cats chased

These sentences can be tersely covered by the following PCFG weighted according to the observed word frequencies (terminal symbols are enclosed in square brackets):

1	s	→	d	n	v	1/3	n	→	[dog]	1/3	v	→	[chases]
1/3	d	→	[a]			1/3	n	→	[cat]	1/6	v	→	[chase]
2/3	d	→	[the]			1/6	n	→	[dogs]	1/2	v	→	[chased]
						1/6	n	→	[cats]				

This grammar is an unsatisfactory model of the training corpus because it does not yield probabilities in accordance with the empirical distribution. For example, the probabilty of "the dog chased" is the product of the probabilities for $s \rightarrow d\ n\ v$, $d \rightarrow [the]$, $n \rightarrow [dog]$ and $v \rightarrow [chased]$, or

$1*2/3*1/3*1/2 = 1/9$, while a similar calculation for the probability of "the dogs chased" gives $1*2/3*1/6*1/2 = 1/18$, yet the training corpus shows both expressions as equiprobable.

The reason for this discrepancy is that the grammar admits twelve sentences not in the training corpus. If the corpus is assumed to be a subset of english, we might subjectively claim that the additional sentences are ill-formed due to contraventions in number agreement. This can be corrected by introducing separate s-rules for singular and plural subject forms and by making a few adjustments to some of the terminal categories, as with:

2/3	s	$\rightarrow$	d1	n1	v1	1/2	n1	$\rightarrow$	[dog]	1/2	v1	$\rightarrow$	[chases]
1/3	s	$\rightarrow$	[the]	n2	v2	1/2	n1	$\rightarrow$	[cat]	1/2	v1	$\rightarrow$	[chased]
1/2	d1	$\rightarrow$	[a]			1/2	n2	$\rightarrow$	[dogs]	1/2	v2	$\rightarrow$	[chase]
1/2	d1	$\rightarrow$	[the]			1/2	n2	$\rightarrow$	[cats]	1/2	v2	$\rightarrow$	[chased]

While this model gives correct probability estimates for individual sentences, there are two obvious drawbacks to accommodating grammatical dependencies in this way. First, rule set expansion is (in general) combinatorial with the number of feature values and, second, each adjustment to the rule set comes with a loss of descriptive power in the characterisation—a better model at the price of a weaker grammatical theory.

Probabilistic unification grammars

A unification grammar is a formal description where grammatical units are associated with sets of feature values. Grammatical units may combine only when their common features unify. A context-free grammar is a zero-order unification grammar: one where grammatical constituents have no features. As we have seen, individual atoms and terms in a CFG are constrained only by the ordering imposed by the right-hand side of phrase structure rules, thus dependencies can only be captured with individual rules. Higher-order unification grammars allow context information to be carried through feature values, and thus provide a means for preserving descriptive power within a more general theory.

Perhaps the simplest and most common formalism for unification grammars is as Prolog DCGs. Grammatical units are combined through the uniform operation of merging and checking feature values contained in clause variables [6]. Like CFGs, UGs can be annotated with probabilities as a means of grading grammaticality. For example, the following probabilistic DCG covers the training corpus given above.

1	s	$\rightarrow$	d(X), n(X), v(X).	1/3	n(s)	$\rightarrow$	[dog].	1/3	v(s)	$\rightarrow$	[chases].
1/3	d(s)	$\rightarrow$	[a].	1/3	n(s)	$\rightarrow$	[cat].	1/6	v(p)	$\rightarrow$	[chase].
2/3	d(_)	$\rightarrow$	[the].	1/6	n(p)	$\rightarrow$	[dogs].	1/2	v(_)	$\rightarrow$	[chased].
				1/6	n(p)	$\rightarrow$	[cats].				

This grammar defines a sentence as any combination of the sub-trees $d(X)$, $n(X)$ and $v(X)$ that unifies with respect to the single common feature X (representing number agreement). Unification is possible whenever X has the same value for all three sub-trees, or when the feature is ungrounded. In this case, the possible values for X are s for singular (which should not be confused with the start symbol) or p for plural. The anonymous variable _ indicates that the clause will unify with any feature value; thus $d(\_) \rightarrow [the]$ is a valid sub-tree for either $d(s)$ or $d(p)$.

Distribution of the probability mass

One might argue (in light of the empirical distribution) that the probability for, say, $n(s){\rightarrow}[dog]$ should be $1/2$ instead of $1/3$ (in the DCG above), given that it is one of two equiprobable re-writes for $n(s)$. It is important to remember, however, that a DCG clause is an abbreviated notation for an equivalent Prolog clause, so that $n(s){\rightarrow}[dog]$ could be translated to

$$n(\_X, \_L1, \_L2) :- \_X = s, \_L1 = [dog|\_L2].$$

In this respect the probability mass should be distributed appropriately over the complete set of rule-analogues for $n(X)$.

Probabilities in context

As with a PCFG, the probability of any sentence covered by a PUG is the product of the probabilities for the rules used in its parse (or, in the case of an ambiguous PUG, the sum of the probabilities for each possible parse). The important difference, however, is that probabilities for alternatives at a given node are not statically determined, but depend upon the contextual information known or assumed when an estimate is made.

For example, imagine we want to know the probability of "the dog chased" given the PUG defined in the previous section. If we interpret rule weights under the same independence assumptions that we make for PCFGs then the probability would be calculated as the product of the probabilities for each clause forming part of the successful parse; or $p(s{\rightarrow}d(X), n(X), v(X)) * p(d(X){\rightarrow}[the]) * p(n(X){\rightarrow}[dog]) * p(v(X){\rightarrow}[chased]) = 1 * 2/3 * 1/3 * 1/2 = 1/9$. But the probability of each clause used in the parse is not independent and must be interpretted in light of the contextual information specified by the value of feature X.

Consider the problem of fixing the probability for $d(X){\rightarrow}[the]$. Though we can be sure that we will use the rule $d(\_){\rightarrow}[the]$, its precise probability depends upon what we know about X. There are three possible cases:

1. if X is grounded to s then we want to know the probability that $d(s){\rightarrow}[the]$, which is $(2/3)/1 = 2/3$;
2. if X is grounded to p then we are asking about the probability that $d(p){\rightarrow}[the]$, which is $(2/3)/(2/3) = 1$;
3. if X is not grounded then the probability for $d(\_){\rightarrow}[the]$ is simply $2/3$.

In the first case, we are asking what the probability is that $d(X){\rightarrow}[the]$ given that $X = s$. As there are two possibilities for $d(s)$, with a total probability mass of 1, then $p(d(s){\rightarrow}[the]) = 2/3$. In the second case, we are asking what the probability is that $d(X){\rightarrow}[the]$ given that $X = p$. There is only one possibility for $d(X)$ given $X = p$, with a total probability mass of $2/3$, making $p(d(p){\rightarrow}[the]) = (2/3)/(2/3) = 1$. In the third case, we are asking what the probability is that $d(X){\rightarrow}[the]$ given that X can be anything.

Now consider finding the probability for "the dog chased" once more. We know the probability for $d(X){\rightarrow}[the]$ is $2/3$ because X is ungrounded. When we come to process the symbol dog, we find that the only applicable rule is $n(s){\rightarrow}[dog]$. Recall that this particular DCG clause is equivalent to the Prolog predicate $n(\_X, \_L1, \_L2) :- \_X = s, \_L1 = [dog|\_L2]$, thus what we really want to know is $p(\_X = s \wedge \_L1 = [dog|\_L2])$, which is simply $p(n(X){\rightarrow}[dog]) = 1/3$. We note that using this clause now grounds X to s, so when we come to calculate the probability for $v(X){\rightarrow}[chased]$ we are actually asking for $p(v(s){\rightarrow}[chased])$. Given there are two possible rules for $v(s)$, with a total probability mass of $(1/3)+(1/2) = 5/6$, then $p(v(s)\rightarrow[chased]) = (1/2)/(5/6) = 3/5$. The probability for "the dog chased" is thus $(2/3) * (1/3) * (3/5) = 2/15$. Probabilities for the complete training set, given this PUG, are as follows:

a dog chases/a cat chases	$1/3 * 1/2 * 2/5$	$=$	$1/15$
the dog chases/the cat chases	$2/3 * 1/3 * 2/5$	$=$	$4/45$
the dogs chase/the cats chase	$2/3 * 1/3 * 1/4$	$=$	$1/36$
a dog chased/a cat chased	$1/3 * 1/2 * 3/5$	$=$	$1/10$
the dog chased/the cat chased	$2/3 * 1/3 * 3/5$	$=$	$4/15$
the dogs chased/the cats chased	$2/3 * 1/6 * 3/4$	$=$	$1/12$

Distributions for anonymous variables

Any PUG rule expressed using an anonymous variable in the lefthand side can be rewritten once for each possible instantiation of that singleton. For example, $d(\_)\rightarrow[the]$ in the grammar above could be expressed using $d(s)\rightarrow[the]$ and $d(p)\rightarrow[the]$ instead. The expanded form is an equivalent grammatical theory in the sense that it covers the same language, but it is not an equivalent grammatical model because the anonymous variable does not allow us to know the distribution of the probability mass over the possible contexts in which the clause might be invoked. The likelihood of $d(s)\rightarrow[the]$ and $d(p)\rightarrow[the]$ are assumed to be the same, though this need not be the case at all.

We could calculate approximate probabilities for $d(s)\rightarrow[the]$ and $d(p)\rightarrow[the]$ by dividing the total probability mass for $d(\_)$ equally over the number of possible instantiations for the anonymous variable. However, the fact that it may be grounded to anything, including nothing at all, makes it impossible to determine exactly how large a divisor we should use.

This inability to determine the distribution for ungrounded features implies that it is impossible to calculate precise entropy mesures from PUGs that employ singleton variables in rule heads. This is an instance of the problem of calculating universal priors implied by Kolmogoroff complexity, and is too lengthy a topic to be discussed in detail here. It is important to note, however, that while the incorporation of anonymous variables gives a grammar more descriptive power as a syntactic theory, its distortion of complexity estimates make it less desirable as a grammatical model, and they should be avoided whenever accurate entropy estimates are an objective.

Training the PUG

The principal problem for stochastic grammars in general is that good approximations of the probability distribution cannot be determined *a priori*. It is usual instead to train the grammar on a sample corpus and then test it against some unseen expressions.

Anytime a probability is changed during training, the remainder of the probability mass must be adjusted across the alternative clauses. But this approach will not work for PUGs because, as we have seen, the probabilities themselves depend upon the context in which a clause is invoked. This problem is avoided by using weights instead of probabilities—counts of how many times a clause is used successfully in parsing. When a probability estimate is needed, the calculation is made using whichever weights are appropriate for the given context.

The training algorithm is implemented in Eclipse Prolog. Annotated rules of the DCG are translated into Prolog clauses, and the weights for each rule are maintained in a global array. Weights are incremented for clauses used in a successful proof.

The wrong way

It is tempting to increment the weight of a given clause after all of its conditional predicates have succeeded, which could be achieved by adding an appropriate predicate at the end of the rule, so that the clause for $v(p) \rightarrow [chase]$ might be translated to something like

$$v(\_X, \_13, \_14) : -\_X = p, \_13 = [chase|\_14], inc\_weight(v(\_X, \_13, \_14)).$$

where inc_weight increments the rule's weight after all conditions have been satisfied. Though the rule itself may be successful, however, subsequent failures may preclude it from forming part of any successful proof tree. Thus a mechanism would be needed to decrement the weight upon backtracking from a failure.

The right way

An alternative method which we use is to simply count how many times a clause is invoked and how many times it is re-entered upon backtracking from a failed parse, and then simply add the difference to the weight when no further parse trees are detected. This difference will be zero for any clauses that do not appear in a successful proof, and non-zero for all those that do. This technique allows the grammar to converge on the optimal weights when the context in which the production is invoked is ignored. The problem of re-entrancies is discussed later.

Example

Training allows the PUG to converge on the optimal weights, thus the accuracy of an entropy estimate at any given point depends upon three things

1. the initial weights before training,
2. the number of training expressions that have been processed, and
3. the representative nature of the training set with respect to the language being modeled.

The actual prior weights become insignificant as the size of the training corpus increases and the burden of evidence moves the weights (in the limit) towards their optimum. Consider, as an example, what happens to the PUG given above when we start with an arbitrarily chosen initial weight of 1 for all clauses. If we train the PUG using the set of sample expressions given earlier, each iteration of training should move the model closer and closer to the empirical distribution.

First, for comparison purposes, we calculate prior probabilities for the most likely and most unlikely expressions in the training set (based on the figures in the table given earlier) with respect to the initial grammar. To find $p($a dog chased$)$, we first calculate $p(d(X) \rightarrow [a]) = 1/2$. This grounds X to s giving $p(n(s) \rightarrow [dog]) = 1/2$ and $p(v(s) \rightarrow [chased]) = 1/2$—thus $p($a dog chased$) = (1/2) * (1/2) * (1/2) = 1/8$. For $p($the cats chase$)$ we find that $p(d(X) \rightarrow [the]) = 1/2$ and $p(n(X) \rightarrow [cats]) = 1/4$, where the latter grounds X to p giving $p(v(p) \rightarrow [chase]) = 1/2$ and a final probability of $(1/2) * (1/4) * (1/2) = 1/16$.

Next we train the model (an arbitrarily chosen) ten times processing all eight expressions on each iteration, producing the following model:

121	s	$\rightarrow$	d(X), n(X), v(X).	41	n(s)	$\rightarrow$	[dog].	41	v(s)	$\rightarrow$	[chases].
41	d(s)	$\rightarrow$	[a].	41	n(s)	$\rightarrow$	[cat].	21	v(p)	$\rightarrow$	[chase].
81	d(_)	$\rightarrow$	[the].	21	n(p)	$\rightarrow$	[dogs].	61	v(_)	$\rightarrow$	[chased].
				21	n(p)	$\rightarrow$	[cats].				

As expected, we see the weight of $d(\_)\rightarrow[the]$ at nearly twice the weight of $d(s)\rightarrow[a]$ (plus one from the original weights), and similarly $v(\_)\rightarrow[chased]$ at nearly three times the weight of $v(p)\rightarrow[chase]$. To estimate the probability of *a dog chased* given this model, we find $p(d(X)\rightarrow[a]) = 41/122$, which grounds X to s giving $p(n(s)\rightarrow[dog]) = 41/82$ and $p(v(s)\rightarrow[chased]) = 61/102$—thus $p($a dog chased$) = (41/122) * (41/82) * (61/102) = 102541/1020408 \simeq 1/10$. For $p($the cats chase$)$ we find that $p(d(X)\rightarrow[the]) = 81/122$ and $p(n(X)\rightarrow[cats]) = 21/124$, where the latter grounds X to p giving $p(v(p)\rightarrow[chase]) = 21/82$ and a final probability of $(81/122) * (21/124) * (21/82) = 35721/1240496 \simeq 1/35$. Continued iteration of the training would see the probabilities move ever closer to those predicted by the empirical distribution.

Re-entrancies

We have shown that probabilistic unification grammars can be very useful for modeling feature-value dependencies. But other types of context should also be considered in entropy estimates, not all of which map easily into a feature unification paradigm. Certain types of re-entrancies can be incorporated into the PUG formalism without much trouble. For example, it may be useful to know whether a nounphrase clause is being invoked in the subject or object position, which could be modeled simply with an additional feature. But if depth of nesting in a center-embedded sentence, such as *the mouse the cat chased died*, had an impact on the probability of a sentence (which is almost certainly true) then this introduces the possibility of an infinite set of possible feature-values, giving rise to the same problem discussed earlier with regard to universal priors.

One possible approach would be to use feature-vectors with variable characteristics. Instead of calculating probabilities in terms of feature-values, some form of partial matching techniques could be applied on the contexts described by the feature-vectors.

In a recent paper, Abney[1] proposes a similar method using random fields, where a probability distribution is defined over a set of labelled graphs (i.e. configurations) with varying topologies. The weight assigned to a configuration is the product of the weights assigned to the configuration properties. Like a partial match approach (as is used in arithmetic encoding, and so forth) the empirical distribution can be recreated by using just enough properties to distinguish among trees. We refer readers to Abney's paper for details about how properties are selected and how their weights are determined.

Remarks

We have presented a formalism for a probabilistic context-sensitive grammar based on feature unification; and demonstrated a learning algorithm that will allow such a grammar to converge (in the limit) to the optimum probabilities that would be given to a fully-expanded PCFG. Perhaps the most important observation to come from this work is that entropy measures should reflect not just the likelihood of structures, but the likelihood of structures in a given context. Because of this, it is essential to affix weights rather than probabilities to elements of grammar, to allow the measure of likelihood to be calculated in light of the known context.

Now that computational linguistics has become entrenched in the vernacular and practice of entropy estimation as a means for evaluating grammars, statistical models of syntax will continue as a "hot" area of research for some time. The increasing interest in stochastic attribute-value grammars is a clear indication that many sound, stochastic methods for building context-sensitive models of language are soon to come.

References

[1] Steven Abney. Stochastic attribute-value grammars. *The Computation and Language E-Print Archive*, October 1996. 9610003.

[2] Chris Brew. Stochastic HPSG. In *Proceedings of EACL-95*, 1995.

[3] Eugene Charniak. *Statistical language learning*. MIT Press, Massachusetts, 1993.

[4] Andreas Eisele. Towards probabilistic extensions of constraint-based grammars. Deliverable r1.2.b, DYANA-2, September 1994.

[5] Albert Kim. Graded unification: A framework for interactive processing. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 313–315. Association for Computational Linguistics, June 1994.

[6] Hans Uszkoreit and Annie Zaenen. Grammar formalisms. In Ron Cole, editor, *A Survey of the State of the Art in Human Language Technology*, chapter 3.3. Center for Spoken Language Understanding, University of Pisa, Italy, 1996.