MDL-based DCG Induction for NP Identification

Miles Osborne
Alfa Informatica, University of Groningen, NL
osborne@let.rug.nl

Abstract

We introduce a learner capable of automatically extending large, manually written natural language Definite Clause Grammars with missing syntactic rules. It is based upon the Minimum Description Length principle, and can be trained upon either just raw text, or else raw text additionally annotated with parsed corpora. As a demonstration of the learner, we show how *full* Noun Phrases (NPs that might contain pre or post-modifying phrases and might also be recursively nested) can be identified in raw text. Preliminary results obtained by varying the amount of syntactic information in the training set suggests that raw text is less useful than additional NP bracketing information. However, using all syntactic information in the training set does not produce a significant improvement over just bracketing information.

1 Introduction

Identification of Noun Phrases (NPs) in free text has been tackled in a number of ways (for example, [25, 9, 2]). Usually however, only relatively simple NPs, such as 'base' NPs (NPs that do not contain nested NPs or postmodifying clauses) are recovered. The motivation for this decision seems to be pragmatic, driven in part by a lack of technology capable of parsing large quantities of free text. With the advent of broad coverage grammars (for example [15] and attendant efficient parsers [11], however, we need not make this restriction: we now can identify 'full' NPs, NPs that might contain pre and/or post-modifying complements, in free text.

Full NPs are more interesting than base NPs to estimate:

- They are (at least) context free, unlike base NPs which are finite state. They can contain pre- and post-modifying phrases, and so proper identification can in the worst case imply full-scale parsing/grammar learning.
- Recursive nesting of NPs means that each nominal head needs to be associated with each NP. Base NPs simply group all potential heads together in a flat structure.

As a (partial) response to these challenges, we identify full NPs by treating the task as a special case of full-scale sentential Definite Clause Grammar (DCG) learning. Our approach is based upon the Minimum Description Length (MDL) principle. Here, we do not explain MDL, but instead refer the reader to the literature (for example, see [26, 27, 29, 12, 22]). Although a DCG learning approach to NP identification is far more computationally demanding than any other NP learning technique reported, it does provide a useful test-bed for exploring some of the (syntactic) factors involved with NP identification. By contrast, other approaches at NP identification more usually only consider lexical/part-of-speech influences.

In this paper, we consider, from an estimation perspective, how dependent NPs are upon their (surrounding) syntactic context. We varied the information content of the training set and measured the effect this had upon NP identification accuracy. Results suggest that:

- Use of any syntactic information, in addition to raw text during estimation, produces better results than estimation from raw text alone.
- NPs containing an internal annotation (nonterminals in addition to NPs) are harder to estimate than NPs that do not contain these additional nonterminals.
- Training with NP annotated sentences and training with sentences annotated with full sentential parses produce very similar results to each other.

We stress that the last finding is provisional, and further investigation is necessary to verify it.

The structure of the rest of this paper is as follows. Section 2 gives an overview of our approach, whilst section 3 goes into estimation and modelling details. We do not start induction *ab initio*, but instead base estimation upon manually written grammars. Section 4 briefly describes the particular grammar used in this research, whilst section 5 relates our work to others. Section 6 presents an experimental evaluation of our learner. The paper ends with a discussion of our findings.

2 Overview

Our learner is probabilistic, and starts with a DCG-based language model M_0. Parameters are initially estimated from parsed corpora, annotated in terms of the non-terminal set used by the DCG. It incrementally processes each sentence s_i in the list of sentences $s_0 \ldots s_k \ldots s_n$. If a sentence s_k cannot be generated (the grammar contained within the model lacks the required rules), we need to find a new model with a high, non-zero posterior probability given the sentences $s_0 \ldots s_k$ seen so far. Our (for computational reasons, necessarily suboptimal) approach selects such a model by carrying out a local search over the space of models with a non-zero posterior probability, given all sentences see so far. We use a MDL-based prior to help us compute a posterior probability. Analogously to Pereira and Schabes (P+S) [23], we also constrain the search using parsed corpora. Unlike P+S, we not only use parsed corpora to constrain parameter estimation, we also use it to constrain model selection. We replace M_0 with the newly constructed (locally) maximal a posterior model and after processing all sentences in this incremental manner, terminate with a model that generates all sentences seen in the training set.

Key aspects of our approach are:

- Incremental learning. We only construct rules necessary to parse sentences in training set. This reduces the computational burden and enables us to learn with grammars that use large (> 30) feature sets. By contrast, batch approaches that compile-out all rules that can be expressed with a fixed category set and with rules limited to some length can only deal with far smaller feature sets, thereby preventing induction of realistic grammars.

- Initialisation with a model containing manually written rules, with parameters estimated from parsed corpora. This alleviates some of the pitfalls of local search and, by definition, makes estimation faster (our initial model is already a reasonable estimate). Lari and Young demonstrated this point when they used an Hidden Markov Model as an approximation of a *Stochastic Context Free Grammar* SCFG [19]. Note that in general any manually written grammar will undergenerate, so there is still a need for new rules to be induced.

- Ability to induce 'fair' models from raw text. We do not select models solely on the basis of likelihood (when training material is limited, such models tend to overfit); instead, we select models in terms of their MDL-based prior probability and likelihood. MDL-based estimation usually reduces overfitting (since, with limited training material, we select a model that fits the training material well, but not too well) [22].

- Learning from positive-only data. We do not require negative examples, nor do we require human intervention. This enables us to transparently use grammar learning as part of standard text parsing.

- Use of parsed corpora allows us to induce models that encode semantic and pragmatic preferences. Raw text is known to underconstrain induction [13] and even with an MDL-based prior, when training upon raw text, we would be unlikely to estimate a model whose hidden variables (grammar rules) resulted in linguistically plausible derivations. Parsed corpora supplies some of this missing information.

3 Estimation Details

Estimation of a model, given a training set of sentences and an associated set of manually constructed parses, consists of four steps: probabilistic modelling of DCGs, model construction, search (constrained by parsed corpora) , and model estimation. We now explain these steps in turn.

Probabilistic Modelling of DCGs

DCGs in our approach are modelled in terms of a compression-based prior probability and a SCFG-based likelihood probability. The prior assigns high probability to compact models, and low probabilities to verbose, idiomatic models. As such, it favours simple grammars over more complex possibilities. The likelihood probability describes how well we can encode the training set in terms of the model. We now turn to the specification of likelihood and prior probabilities in our system.

Likelihood Probability To specify a likelihood probability for DCGs, we have opted to use a SCFG, which consists of a set of context free grammar rules along with an associated set of parameters [6]. Each parameter models the way we might expand non-terminals in a top-down derivation process, and within a SCFG, we associate one such parameter with each distinct context free rule. However, DCG rules are feature-based, and so not directly equivalent to simple context free rules. In order to define a SCFG over DCG rules, we need to interpret them in a context-free manner. One way to achieve this is as follows. For each category in the grammar that is distinct in terms of features, invent an atomic non-terminal symbol. With these atomic symbols, create a SCFG by mapping each category in a DCG rule to an atomic symbol, yielding a context free (backbone) grammar, and with this grammar, specify a SCFG, M_i. Naturally, this is not the most accurate probabilistic model for feature-based grammars, but for the interim, is sufficient (see Abney for a good discussion of how one might define a more accurate probabilistic model for feature-based grammars [1]).

SCFGs are standardly defined as follows. Let $P(A \rightarrow \alpha \mid A)$ be the probability of expanding (backbone) non-terminal symbol A with the (backbone) rule $A \rightarrow \alpha$ when deriving some sentence s_i. The probability of the j^{th} derivation of s_i is defined as the product of the probabilities of all backbone rules used in that derivation. That is, if derivation j followed from an application of the rules $A_1^j \rightarrow \alpha_1^j, \ldots, A_n^j \rightarrow \alpha_n^j$,

$$P^j_{\text{deriv}}(s_i \mid M_i) = \prod_{i=1}^{n} P(A_i^j \rightarrow \alpha_i^j) \qquad (1)$$

The probability of a sentence is then defined as the sum of the probabilities of all n ways we can derive it:

$$P_s(s_i \mid M_i) = \sum_{j=1}^{n} P^j_{\text{deriv}}(s_i \mid M_i) \qquad (2)$$

Having modelled DCGs as SCFGs, we can immediately specify the likelihood probability of M_i generating a sample of sentences $s_0 \ldots s_n$, as:

$$P(s_0 \ldots s_n \mid M_i) = \prod_{j=0}^{n} P_S(s_j \mid M_i) \qquad (3)$$

This treats each sentence as being independently generated from each other sentence.

Prior Probability Specifying a prior for DCGs amounts to encoding the rules and the associated parameters. We encode DCG rules in terms of an integer giving the length, in categories, of the rule (requiring $\log^*(n)$ bits, where $\log^*$ is Rissannen's encoding scheme for integers), and a list of that many encoded categories. Each category consists of a list of features, drawn from a finite set of features, and to each feature there is a value. In general, each feature will have a separate set of possible values. Within manually written DCGs, the way a feature is assigned a value is sensitive to the position, in a rule, of the category containing the feature in question. Hence, if we number the categories of a rule, we can work out the probability that a particular feature, in a given category, will take a certain value. Let $P(v \mid f_i)$ be the probability that feature f takes the value v, in category i of all rules in the grammar. Each value can now be encoded with a prefix code of $-\log(P(v \mid f_i)$ bits in length. Encoding a category simply amounts to a (fixed length) sequence of such encoded features, assuming some canonical ordering upon features. Note we do not learn lexical entries and so not not need to encode them.

To encode the model parameters, we simply use Rissannen's prefix coding scheme for integers to encode a rule's frequency. We do not directly encode probabilities, since these will be inaccurate when the frequency, used to estimate that probability, is low. Rissannen's scheme has the property that small integers are assigned shorter codes than longer integers. In our context, this will favour low frequencies over higher ones, which is undesirable, given the fact that we want, for estimation accuracy, to favour higher frequencies. Hence, instead of encoding an integer i in $\log^*(i)$ bits (as, for example, Keller and Lutz roughly do [18]), we encode it in $\log^*(Z - i)$ bits, where Z is a number larger than any frequency. This will mean that higher frequencies are assigned shorter code words, as intended.

The prior probability of a model M_i, containing a DCG G and an associated parameter set is:

$$P(M_i) = 2^{-(l_g(M_i)+l_p(M_i))} + C \qquad (4)$$

where:

$$l_g(M_i) = \sum_{r \in G} [\log^*(\mid r \mid) + \sum_{i=1}^{\mid r \mid} \sum_{f \in F} -\log(P(v \mid f_i))] \qquad (5)$$

is description length of the grammar and

$$l_p(M_i) = \sum_{r \in G} \log^*(Z - f(r)) \qquad (6)$$

is the description length of the parameters. C is a constant ensuring that the prior sums to one; F is the set of features used to describe categories: $\mid r \mid$ is the length of a DCG rule r seen $f(r)$ times.

Apart from being a prior over DCG rules, our scheme has the pleasing property that it assigns longer code words to rules containing categories in unlikely positions than to rules containing categories in expected positions. For example, our scheme would assign a longer list of code words to the categories expressing a rule such as $Det \rightarrow Det\ NP$ than to the list of categories expressing a rule such as $NP \rightarrow Det\ NP$. Also, our coding scheme favours shorter rules than longer rules, which is desirable, given the fact that, generally speaking, rules in natural language grammars tend to be short.

Posterior Probability In summary, the probability of a model, given a set of training examples is:

$$P(M_i \mid s_0 \ldots s_n) = \frac{[2^{-(l_g(M_i)+l_p(M_i))}+C]\cdot\prod_{j=0}^{n} P_S(s_j|M_i)}{P(s_0 \ldots s_n)} \tag{7}$$

Model Construction

For lack of space, we only sketch our model construction strategy. In brief, our approach is (largely) monotonic, in that we extend an old model with new rules constructed from pieces of manually written rules, such that the new and old rules result in a parse for the current sentence (and the old rules alone fail to parse the current sentence). In more detail, whenever we fail to parse a sentence with the manually written DCG, we use an optimised chart parser [10] to construct all local trees licensed by the manually written grammar. We next consider ways adjacent local trees licensed by manually written DCG rules may be combined into larger local trees (ie invent rules whose right hand side consists of categories that spell out the mother categories of these adjacent local trees; the left hand side will be one of these right-hand side categories, with the possibility of having its bar level raised). The parser packs all local trees in space polynomial with respect to sentence length. If, within self-imposed bounds upon computation (for example, limiting the number of local trees joined together), we succeed in constructing at least one tree that spans the entire sentence, we can build a new model by extracting all new rules seen in that tree and adding them to the old model.

Note that periodically, it is useful to prune (and renormalise) the model of rules seen only once in the previously encountered sequence of sentences. Such pruned rules are likely to have arisen either due to marked constructions, noise in the training material, or rules that appeared to be promising, but did not receive any subsequent support and as such have little predictive utility. Pruning the model is a non-monotonic operation and hard to formally justify, but nevertheless useful. Whitten, Cleary and Bell also comment upon the usefulness of resetting the model [3].

Our model construction approach has the following properties: rules constructed all encode a version of $\bar{X}$-Syntax, which weakly constrains the space of possible rules [8, 21]; analyses produced using manually written rules are favoured over those produced using learnt rules (by virtue of computation being resource-bounded): this mirrors the empirical fact that when extending manually written grammars, only a few rules are necessary, and those required generally 'join' together local trees generated by manually written rules.

Search

Model construction may produce an exponential number of parses for a sentence and for computational reasons, we are unable to evaluate all the models encoded within these parses. We therefore use a probabilistic unpacking strategy that efficiently selects the n most likely parses, where n is much less than the total number of parses possible for some sentence [11]. There is insufficient space here to describe how we rank parses, but the underlying parse selection model is based upon the SCFG used to evaluate models. Currently, it is not lexicalised, so parse selection performance is subject to the well-known limitations of non-lexicalised SCFGs [5]. Whilst estimating models, we simultaneously estimate the parse selection model in terms of the parse used to produce the model picked. Probabilistic unpacking is crucial to our approach, and it is this that makes our learner computationally feasible.

After extracting n parses, we can then go on to construct k[1] models, evaluate their posterior probabilities, and then select the model that maximises this term. However, as was shown by P+S, when training material consists of just (limited quantities) of raw text, classical, single model parameter estimation often results in a model that produces worse parse selection results than when the estimation process is constrained to only consider derivations compatible with the parsed corpora. In our context, we can use parsed corpora to constrain both parameter estimation and also model selection. We simply re-rank the n parses produced during model construction using a tree similarity metric that compares how 'close' an automatically constructed parse is to a manually written parse, and take the q parses that all minimise the metric and are all scored equally well [17]. From these q parses we can then build models as usual. When $q = 1$, there is no need to rely upon MDL-based model selection. Otherwise, when q is greater than one, we have a set of parses, all equally consistent with the manually created tree, and so fall-back upon the usual model selection strategy. Our use of parsed corpora differs from P+S's in that we use it as a soft constraint: we may still keep parses even if they violate constraints in the manually constructed parse tree. The reason for this decision is that we do not construct all possible parses for a sentence, and so at times may not produce a parse consistent with a manually created parse. Also, it is not clear whether parsed corpora is sufficiently reliable for it to be trusted absolutely. Clearly there will be a link between the amount of information present in the parsed corpora and the quality of the estimated model. In the experimental

[1] k may be less than or equal to n, depending upon which independence assumptions are made by the model

section of this paper, we consider this issue.

Estimation

When computing a model's posterior probability, we estimate the description length of features, $P(v \mid f_i)$, the model parameters, $P(A \rightarrow \alpha \mid A)$ and the likelihood probability, $P(s_0 \ldots s_n \mid M_i)$.

The feature description length is estimated by counting the number of times a given feature takes some value in some category position, and applying a maximal likelihood estimator to arrive at a probability. The model parameters are estimated by counting the number of times a given backbone rule was seen in the previous n parses just produced, and then again using a maximal likelihood estimator to produce a probability. We estimate, as we cannot afford to globally recompute, the likelihood probability using the following approximations. Only a fixed number of n previously seen sentences that cannot be parsed using the manually written rules are considered in the likelihood computation. We assume that the parses of these sentences remains constant across alternative models, but the derivation probabilities might vary. We also assume that the string probability of each sentence is reasonably well approximated by a single parse.

4 The Grammar

The grammar we extend with learning, (called the *Tag Sequence Grammar* [7], or TSG for short) was developed with regard to coverage, and when compiled consists of 455 object rules. It does not parse sequences of words directly, but instead assigns derivations to sequences of part-of-speech tags (using the CLAWS2 tagset [4]). The grammar is relatively shallow, (for example, it does not fully analyse unbounded dependencies) but it does make an attempt to deal with common constructions, such as dates or names, commonly found in corpora, but of little theoretical interest. Furthermore, it integrates into the syntax a text grammar, grouping utterances into units that reduce the overall ambiguity.

For the experiments reported here, we manually extended TSG with four extra rules. These extra rules dealt with obvious oversights when parsing the WSJ.

5 Related Work

Our approach is closely related to Stolcke's model merging work [29]. Apart from differences in prior and likelihood computation, the main divergence is that our work is motivated by the need to deal with undergeneration in broad-coverage, manually written natural language grammars (for example [15]). Although we do not go into the issues here, estimation of rules missing from such grammars is different from estimating grammars *ab initio*. This is because rules missing from any realistic grammar are all likely to have a low frequency in any given corpus, and so will be harder to differentiate from competing, incorrect rules purely on the basis of statistical properties alone. We know of no other work reporting automated extension of broad-coverage grammars using MDL and parsed corpora.

One of the anonymous reviewers wanted to know how our work related to Explanation-Based Learning (EBL) [20]. EBL is not concerned with induction of rules: it deals with finding more efficient ways to use existing rules. For example, in NLP, EBL has been used to reduce the time taken to parse sentences with large grammars ([28]). EBL does not extend the coverage of any given grammar, unlike our approach. In our opinion, it would be better to view our learner as an Inductive Logic Programming system specialised for DCG induction.

6 Experiments

For our experiments we used material supplied by the CoNLL99 workshop organisers. This consisted of 48,224 fully parsed training sentences, and a disjoint set of 984 testing sentences. Both sets were randomly drawn from the parsed section of the Wall Street Journal. The test set came in two versions, differing from each other in how the sentences were marked-up. The first version consisted of sentences with NP bracketings marked (results using this test set are given in table 1). The second version had NP bracketings marked, and within each marked NP, there was an internal parse (results for this version are in table 2). These parses were labelled with Penn Nonterminals. Each test sentence was trivially rooted with an S symbol (necessary for the evaluation software). To make this clearer, if an original Wall Street Journal parse, stripped of tags and nonterminal decorations was:

(S (NP (NP Bailey Controls), (VP based (PP in (NP (NP Wickliffe), (NP Ohio)))).) (VP makes (NP computerized industrial controls systems)).)

the version containing just NP bracketing would be:

(S (NP (NP Bailey Controls), based in (NP (NP Wickliffe), (NP Ohio)),) makes (NP computerized industrial controls systems) .)

whilst the version containing parsed NPs would be:

(S (NP (NP Bailey Controls), (VP based (PP in (NP (NP Wickliffe), (NP Ohio)))),) makes (NP computerized industrial controls systems).)

For computational reasons, we could not deal with all sentences in the training set, and when learning rules, we limited ourselves to sentences with a maximum length of 15 tokens. During evaluation, we used sentences with a maximum length of 30 tokens. This reduced the training set to 10,249 parsed sentences, and the test set to 739 sentences. Finally, we retagged the CoNLL99 material with the Claws2 tagset (required by TSG). Evaluation was carried out by Parseval (which reports unlabelled bracketing results: we do not report labelled results as TSG does not use the Penn Nonterminal set) [16]. Note that evaluation is based upon bracketing, and not word accuracy. For example, if we failed to include one word in a NP that contains four other words, we would have a bracketing accuracy of 0. On the other hand, a word accuracy result would be 80%.

As a comparison, we evaluated TSG upon the testing material. This is experiment 5 in tables 1 and 2. The other four experiments differed from each other in terms of what the learner was trained upon:

1. Just tagged sentences.

2. Tagged Sentences with NP bracketings marked. We reduced WSJ parses to include just their NP bracketings.

3. Tagged sentences with NPs bracketing annotated with an internal parse. Again, we mapped WSJ parses to reduced parses containing just annotated NPs.

4. Tagged sentences with a full Wall Street Journal parse.

For each experiment, we report the size of the final grammar, the percentage of testing sentences covered (assigned a full parse), crossing rates, recall and precision results with respect to testing sentences with NPs bracketed and those containing annotated NPs. For the bracketing task, we mapped full parses, produced by models, to parses just containing NP bracketing. For the annotation task. we mapped full parses to parses containing just NPs with an internal annotation. Note that within our grammatical framework, the best mapping is not clear (since parses produced by our models have categories using multiple bar levels, whilst WSJ parses only use a single level). As a guess, we treated bar 1 and bar 2 nominal categories as being NPs. This means that our precision results are lowered, since in general, we produce more NPs than would be predicted by a WSJ parse.

In each case. we evaluate the performance of a model in terms of the highest ranked parse. and secondly, in terms of the 'best' parse, out of the top 10 parses produced. Here 'best' means the parse produced that is closest, in terms of a weighted sum crossing rates, precision and recall, to the manually selected parse. This final set of results gives an indication of how well our system would perform if it had a much better parse selection mechanism. Best figures are marked in parentheses.

Figure 1 gives our results for the bracketing task, whilst figure 2 gives our results for the annotation task. Model size and coverage results were identical for both tests, so the second table omits them.

Exp	Size	% Gen	CR	R	P
1	2687	78	1.27 (1.02)	66.5 (73.9)	51.5 (55.
2	2774	91	1.34 (0.99)	64.6 (73.5)	50.8 (55.
3	2782	91	1.29 (0.97)	64.6 (73 5)	50.9 (55.
4	2592	90	1.34 (1.01)	64.8 (73.2)	50.5 (55.
5	459	63	1.2 (0.95)	68.7 (75.8)	53 2 (56.

Figure 1: NP Identification Results (Bracketing)

Exp	CR	R	P
1	2.08 (1.52)	57.99 (67.7)	48.1 (54.6)
2	2.27 (1.54)	56.5 (67.0)	46.0 (54.2)
3	2.22 (1.52)	56.5 (66.9)	50.9 (54.2)
4	2.32 (1.63)	56.5 (66.7)	45.4 (53.1)
5	1.85 (1.40)	59.2 (69.1)	51.4 (57.2)

Figure 2: NP Identification Results (Annotation)

Firstly, when compared with other work on NP recovery, our results are poor. As was mentioned in the search section, this is largely due to our system being based upon a language model that has well known limitations. Furthermore, as was argued in the introduction, full NPs are by definition harder to identify than base NPs, so we would expect our results to be worse. Secondly, we see that the bracketing task is easier than the annotation task: generally. the results in table 1 are better than the results in table 2. Given the fact that the annotation search space is larger than the bracketing space, this should come as no surprise. Turning now to the individual experiments, we see that parsed corpora (experiments 2, 3 and 4) is an informative constraint upon NP induction. Rules learnt using parsed corpora better capture regularities than do rules learnt from just raw text (experiment 1). This is shown by the increased coverage results of experiments 2. 3 and 4 over 1. In terms of crossing rates, recall and precision, no clear story has emerged. Surprisingly, there seems to be minimal difference in coverage when using either annotated NPs or full parses. This could be due to a number of reasons, such as WSJ NPs being more reliably annotated than other phrases, simple artifactual problems with the learner, the evaluation metrics being too coarse to show any real differences, etc. Further, qualitative investigation should determine whether there are any differences in the parses that TSG alone cannot assign to sentences.

Due to time constraints, we did not measure statistical significance tests between the various experiments. A later version of this paper (available from the author, osborne@let.rug.nl) will report these tests.

7 Conclusion

We presented an MDL-based incremental DCG learner. Experimental evaluation showed that estimation is possible using just raw sentences, but that better results are possible when additional parsed corpora is used. Evaluation also showed that this parsed corpora need not be that detailed, and that NP bracketing information produced similar results to using full WSJ parses. This final results seems counterintuitive, and merits further investigation.

Future work on the learner will be in three main directions:

- Abandonment of the SCFG as the basis of the language model. We are considering either Abney's random fields [1] or Goodman's Probabilistic Feature Grammmars [14] as a replacement. Apart from performance improvements, altering the model class should allow empirical investigation of the MDL claim that model classes can be evaluated in terms of compression. So, if we discover even more compact models using (say) Goodman's scheme than we could using our SCFG, we might deduce that this is the case. Naturally, lexicalisation would enter into any scheme entertained.

- Use of semantics in estimation. We have at our disposal a large grammar augmented with a compositional semantics [15]. Again, this should lead to better results.

- Prior weighting. As is well known, MDL-based learners sometimes improve from weighting the prior with respect to the likelihood. Schemes, such as Quinlan and Rivest's [24], fall outside of the coding framework and (effectively) replicate the training set. We intend to pursue encoding-based schemes that achieve the same purpose.

Acknowledgments

We would like to thank Erik Tjong Kim Sang for help with organising CoNLL99 and the two anonymous reviewers for commenting on the paper. This work was supported by the EU Project *Sparkle* LE-2111 and the TMR Project *Learning Computational Grammars.*

References

[1] Steven P. Abney. Stochastic Attribute-Value Grammars. *Computational Linguistics*, 23(4):597–618, December 1997.

[2] Shlomo Argamon, Ido Dagan, and Yuval Krymolowski. A Memory-Based Approach to Learning Shallow Natural Language Patterns. In *Proceedings of the 17th International Conference on Computational Linguistics*, 1998. http://xxx.lanl.gov/ps/cmp-lg/9806011.

[3] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression.* Advanced Reference Series. Prentice Hall, 1990.

[4] Ezra Black, Roger Garside, and Geoffrey Leech, editors. *Statistically driven computer grammars of English the IBM-Lancaster approach.* Rodopi, 1993.

[5] Ezra Black, Fred Jelinek, John Lafferty, and David M. Magerman. Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In 31st *Annual Meeting of the Association for Computational Linguistics*, pages 31–37, Ohio State University, Columbus, Ohio, USA, June 1993.

[6] T. Booth. Probabilistic representation of formal languages. In *Tenth Annual IEEE Symposium on Switching and Automata Theory*, October 1969.

[7] Ted Briscoe and John Carroll. Automatic Extraction of Subcategorization from Corpora. In *Proceedings of the 5th Conference on Applied NLP*, pages 356–363, Washington, DC, 1996.

[8] Ted Briscoe and Nick Waegner. Robust Stochastic Parsing Using the Inside-Outside Algorithm. In *Proceedings of the AAAI Workshop on Statistically-based Techniques in Natural Language Processing*, 1992.

[9] Claire Cardie and David Pierce. Error-Driven Pruning of Treebank Grammars for Base Noun Phrase Identificatio. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 218–224, 1998.

[10] John Carroll. *Practical Unification-based Parsing of Natural Language.* PhD thesis, University of Cambridge, March 1993.

[11] John Carroll and Ted Briscoe. Probabilistic normalisation and unpacking of packed parse forests for unification-based grammars. In *Proceedings of*

the AAAI Fall Symposium on Probabilistic Approaches to Natural Language, pages 33–38, Cambridge, MA, 1992.

[12] Carl de Marcken. *Unsupervised Language Acquisition.* PhD thesis, MIT, 1996.

[13] E. M. Gold. Language Identification to the Limit. *Information and Control*, 10:447–474, 1967.

[14] Joshua Goodman. Probabilistic Feature Grammars. In 5th *International Workshop on Parsing Technologies*, MIT, Cambridge, Massachusetts, USA, September 1997.

[15] Claire Grover, Ted Briscoe, John Carroll, and Bran Boguraev. *The Alvey Natural Language Tools Grammar (4th Release).* Technical report, University of Cambridge Computer Laboratory, 1993.

[16] Philip Harrison, Steven Abney, Ezra Black, Dan Flickinger, Ralph Grishman Claudia Gdaniec, Donald Hindle, Robert Ingria, Mitch Marcus, Beatrice Santorini, and Tomek Strzalkowski. Evaluating Syntax Performance of Parser/Grammars of English. In Jeannette G. Neal and Sharon M. Walter, editors, *Natural Language Processing Systems Evaluation Workshop, Technical Report RL-TR-91-362*, 1991.

[17] Eirik Hektoen. Probabilistic Parse Selection Based on Semantic Cooccurrences. In 5th *International Workshop on Parsing Technologies*. pages 113–122, MIT, Cambridge, Massachusetts. USA. September 1997.

[18] Bill Keller and Rüdi Lutz. Evolving Stochastic Context-Free Grammars from Examples Using a Minimum Description Length Principle. In *Worksop on Automata Induction, Grammatical Inference and Language Acquisition*, Nashville, Tennessee, USA, July 1997. ICML097.

[19] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[20] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1.1:47–80, 1986.

[21] Miles Osborne and Derek Bridge. Learning unification-based grammars using the Spoken English Corpus. In *Grammatical Inference and Applications*, pages 260–270. Springer Verlag. 1994.

[22] Miles Osborne and Ted Briscoe. Learning Stochastic Categorial Grammars. In T. Mark Ellison, editor, *CoNLL97*, pages 80–87. ACL, July 1997.

[23] Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the* 30th *ACL. University of Delaware, Newark, Delaware*, pages 128–135, 1992.

[24] J. Ross Quinlan and Ronald L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[25] Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking Using Transformation-Based Learning,. In *Proceedings of the* 3rd *ACL Workshop on Very Large Corpora*, pages 82–94, June 1995.

[26] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry.* Series in Computer Science -Volume 15. World Scientific, 1989.

[27] Jorma Rissanen and Eric Sven Ristad. Language Acquisition in the MDL Framework. In Eric Sven Ristad, editor, *Language Computation.* American Mathemtatical Society, Philedelphia, 1994.

[28] Christer Samuelsson and Manny Rayner. Quantiative Evaluation of Explanation-Based Learning as an Optimisation Tool for a Large-Scale Natural Language System. In *Proceedings of the* 12th *International Joint Conference on Artificial Intelligence*, pages 609–615, 1991.

[29] Andreas Stolcke and Stephen Omohundro. Inducing Probabilistic Grammars by Bayesian Model Merging. In *Grammatical Inference and Applications*, pages 106–118. Springer Verlag. 1994.