Inducing Combinatory Categorial Grammars with Genetic Algorithms

Elias Ponvert
Department of Linguistics
University of Texas at Austin
1 University Station B5100
Austin, TX 78712-0198 USA
ponvert@mail.utexas.edu

Abstract

This paper proposes a novel approach to the induction of Combinatory Categorial Grammars (CCGs) by their potential affinity with the Genetic Algorithms (GAs). Specifically, CCGs utilize a rich yet compact notation for lexical categories, which combine with relatively few grammatical rules, presumed universal. Thus, the search for a CCG consists in large part in a search for the appropriate categories for the data-set's lexical items. We present and evaluates a system utilizing a simple GA to successively search and improve on such assignments. The fitness of categorial-assignments is approximated by the coverage of the resulting grammar on the data-set itself, and candidate solutions are updated via the standard GA techniques of reproduction, crossover and mutation.

1 Introduction

The discovery of grammars from unannotated material is an important problem which has received much recent research. We propose a novel approach to this effort by leveraging the theoretical insights of Combinatory Categorial Grammars (CCG) (Steedman, 2000), and their potential affinity with Genetic Algorithms (GA) (Goldberg, 1989). Specifically, CCGs utilize an extremely small set of grammatical rules, presumed near-universal, which operate over a rich set of grammatical categories, which are themselves simple and straightforward data structures. A search for a CCG grammar for a language can be construed as a search for accurate category assignments to the words of that language, albeit over a large landscape of potential solutions. GAs are biologically-inspired general purpose search/optimization methods that have succeeded in these kinds of environments: wherein solutions are straightforwardly coded, yet nevertheless the solution space is complex and difficult.

We evaluate a system that uses a GA to successively refine a population of categorial lexicons given a collection of unannotated training material.

This is an important problem for several reasons. First of all, the development of annotated training material is expensive and difficult, and so schemes to discover linguistic patterns from unannotated text may help cut down the cost of corpora development. Also, this project is closely related to the problem of resolving lexical gaps in parsing, which is a dogged problem for statistical parsing systems in CCG, even trained in a supervised manner. Carrying over techniques from this project to that could help solve a major problem in CCG parsing technology.

Statistical parsing with CCGs is an active area of research. The development of CCGbank (Hockenmaier and Steedman, 2005) based on the Penn Treebank has allowed for the development of wide-coverage statistical parsers. In particular, Hockenmaier and Steedman (2001) report a generative model for CCG parsing roughly akin to the Collins parser (Collins, 1997) specific to CCG. Whereas Hockenmaier's parser is trained on (normal-form) CCG derivations, Clark and Curran (2003) present a CCG parser trained on the dependency structures within parsed sentences, as well as the possible derivations for them, using a log-linear (Maximum-Entropy) model. This is one of the most accurate parsers for producing deep dependencies currently available. Both systems, however, suffer from gaps

Proceedings of the ACL 2007 Student Research Workshop, pages 7–12,
Prague, June 2007.

in lexical coverage.

The system proposed here was evaluated against a small corpus of unannotated English with the goal of inducing a categorial lexicon for the fragment. The system is not ultimately successful and fails to achieve the baseline category assignment accuracy, however it does suggest directions for improvement.

2 Background

2.1 Genetic Algorithms

The basic insight of a GA is that, given a problem domain for which solutions can be straightforwardly encoded as *chromosomes*, and for which candidate solutions can be evaluated using a faithful *fitness function*, then the biologically inspired operations of *reproduction*, *crossover* and *mutation* can in certain cases be applied to multisets or *populations* of candidate solutions toward the discovery of true or approximate solutions.

Among the applications of GA to computational linguistics, (Smith and Witten, 1995) and (Korkmaz and Üçoluk, 2001) each present GAs for the induction of phrase structure grammars, applied successfully over small data-sets. Similarly, (Losee, 2000) presents a system that uses a GA to learn part-of-speech tagging and syntax rules from a collection of documents. Other proposals related specifically to the acquisition of categorial grammars are cited in §2.3.

2.2 Combinatory Categorial Grammar

CCG is a mildly context sensitive grammatical formalism. The principal design features of CCG is that it posits a small set of grammatical rules that operate over rich grammatical categories. The categories are, in the simplest case, formed by the atomic categories s (for *sentence*), np (*noun phrase*), n (*common noun*), etc., closed under the slash operators /, \. There is not a substantive distinction between lexical and phrasal categories. The intuitive interpretation of non-atomic categories is as follows: a word for phrase of type A/B is looking for an item of type B on the right, to form an item of type A. Likewise, an item of type $A\backslash B$ is looking for an item of type B on the left. type A. For example, in the derivation in Figure 1, "scores" combines with the np "another goal" to form the verb phrase "scores another goal". This, in turn, combines with the np "Ronaldinho" to form a sentence.

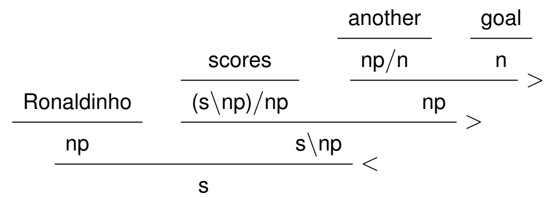

Figure 1: Example CCG derivation

Application					
$A/B\ B$	$\Rightarrow_{>}$	A	$B\ A\backslash B$	$\Rightarrow_{<}$	A
Composition					
$A/B\ B/C$	$\Rightarrow_{>\mathsf{B}}$	A/C	$B\backslash C\ A\backslash B$	$\Rightarrow_{<\mathsf{B}}$	$A\backslash C$
Crossed-Composition					
$A/B\ B\backslash C$	$\Rightarrow_{>\mathsf{B}_\times}$	$A\backslash C$	$B/C\ A\backslash B$	$\Rightarrow_{<\mathsf{B}_\times}$	A/C

Figure 2: CCG Rules

The example illustrates the rule of Application, denoted with $<$ and $>$ in derivations. The schemata for this rule, along with the Composition rule (B) and the Crossed-Composition rule ($\mathsf{B}_\times$), are given in Figure 2. The rules of CCG are taken as universals, thus the acquisition of a CCG grammar can be seen as the acquisition of a categorial lexicon.

2.3 Related Work

In addition to the supervised grammar systems outlined in §1, the following proposals have been put forward toward the induction of categorial grammars.

Watkinson and Mandahar (2000) report a Categorial Grammar induction system related to that proposed here. They generate a Categorial Grammar using a fixed and limited set of categories and, utilizing an unannotated corpus, successively refine the lexicon by testing it against the corpus sentences one at a time. Using a constructed corpus, their strategy worked extremely well: 100% accuracy on lexical category selection as well as 100% parsing accuracy with the resulting statistical CG parser. With naturally occurring text, however, their system does not perform as well: approximately 77% lexical accuracy and 37% parsing accuracy.

One fundamental difference between the strategy proposed here and that of Watkinson and Mandahar is that we propose to successively generate and evaluate *populations* of candidate solutions, rather than refining a single solution. Also, while Watkinson and Mandahar use logical methods to construct a probabilistic parser, the present system uses approximate methods and yet derives symbolic parsing systems. Finally, Watkinson and Mandahar utilize an extremely small set of known categories, smaller than the set used here.

Clark (1996) outlines a strategy for the acquisition of Tree-Adjoining Grammars (Joshi, 1985) similar to the one proposed here: specifically, he outlines a learning model based on the *co-evolution* of a parser, which builds parse trees given an input string and a set of category-assignments, and a *shredder*, which chooses/discovers category-assignments from parse-trees. The proposed strategy is not implemented and tested, however.

Briscoe (2000) models the acquisition of categorial grammars using evolutionary techniques from a different perspective. In his experiments, language agents induced parameters for languages from other language agents generating training material. The acquisition of languages is not induced using GA per se, but the evolutionary development of languages is modeled using GA techniques.

Also closely related to the present proposal is the work of Villavicencio (2002). Villavicencio presents a system that learns a unification-based categorial grammar from a semantically-annotated corpus of child-directed speech. The learning algorithm is based on a Principles-and-Parameters language acquisition scheme, making use of logical forms and word order to induce possible categories within a typed feature-structure hierarchy. Her system has the advantage of not having to pre-compile a list of known categories, as did Watkinson and Mandahar as well as the present proposal. However, Villavicencio does make extensive use of the semantics of the corpus examples, which the current proposal does not. This is related to the divergent motivations of two proposals: Villavicencio aims to present a psychologically realistic language learner and takes it as psychologically plausible that logical forms are accessible to the language learner; the current proposal is preoccupied with grammar induction from unannotated text, and assumes (sentence-level) logical forms to be inaccessible.

n is the size of the population
A are candidate category assignments
F are fitness scores
E are example sentences
m is the likelihood of mutation

Initialize:
  for $i \leftarrow 1$ to n :
    $A[i] \leftarrow$ RANDOMASSIGNMENT()
Loop:
  for $i \leftarrow 1$ to $length[A]$:
    $F[i] \leftarrow 0$
    $P \leftarrow$ NEWPARSER($A[i]$)
    for $j \leftarrow 1$ to $length[E]$:
      $F[i] \leftarrow F[i] +$ SCORE(P.PARSE($E[i]$))
  $A \leftarrow$ REPRODUCE(A, F)
  ▷ Crossover:
  for $i \leftarrow 1$ to $n-1$:
    CROSSOVER($A[i], A[i+1]$)
  ▷ Mutate:
  for $i \leftarrow 1$ to n :
    if RANDOM() $< m$:
      MUTATE($A[i]$)
Until: End conditions are met

Figure 3: Pseudo-code for CCG induction GA.

3 System

As stated, the task is to choose the correct CCG categories for a set of lexical items given a collection of unannotated or minimally annotated strings. A candidate solution *genotype* is an assignment of CCG categories to the lexical items (types rather than tokens) contained in the textual material. A candidate *phenotype* is a CCG parser initialized with these category assignments. The fitness of each candidate solution is evaluated by how well its phenotype (parser) parses the strings of the training material.

Pseudo-code for the algorithm is given in Fig. 3. For the most part, very simple GA techniques were used; specifically:

- REPRODUCE The reproduction scheme utilizes roulette wheel technique: initialize a weighted roulette wheel, where the sections of the wheel correspond to the candidates and the weights of the sections correspond to the fitness of the candidate. The likelihood that a candidate is selected in a roulette wheel spin is directly proportionate to the fitness of the candidate.

- CROSSOVER The crossover strategy is a simple partition scheme. Given two candidates C and

D, choose a center point $0 \leq i \leq n$ where n the number of genes (category-assignments), swap C[0, i] ← D[0, i] and D[i, n] ← C[i, n].

- MUTATE The mutation strategy simply swaps a certain number of individual assignments in a candidate solution with others. For the experiments reported here, if a given candidate is chosen to be mutated, 25% of its genes are modified. The probability a candidate was selected is 10%.

In the implementation of this strategy, the following simplifying assumptions were made:

- A given candidate solution only posits a single CCG category for each lexical item.
- The CCG categories to assign to the lexical items are known a priori.
- The parser only used a subset of CCG – pure CCG (Eisner, 1996) – consisting of the Application and Composition rules.

3.1 Chromosome Encodings

A candidate solution is a simplified assignment of categories to lexical items, in the following manner. The system creates a candidate solution by assigning lexical items a random category selection, as in:

Ronaldinho	(s\np)/np
Barcelona	pp
kicks	(s\np)/(s\np)
⋮	

Given the fixed vocabulary, and the fixed category list, the representation can be simplified to lists of indices to categories, indexed to the full vocabulary list:

0	Ronaldinho
1	Barcelona
2	kicks
⋮	

⋮	
15	(s\np)/np
⋮	
37	(s\np)/(s\np)
⋮	

Then the category assignment can be construed as a finite function from word-indices to category-indices $\{0 \mapsto 15, 1 \mapsto 42, 2 \mapsto 37, ...\}$ or simply the vector $\langle 15, 42, 37, ...\rangle$. The chromosome encodings for the GA scheme described here are just this: vectors of integer category indices.

3.2 Fitness

The parser used is straightforward implementation of the normal-form CCG parser presented by Eisner (1996). The fitness of the parser is evaluated on its parsing coverage on the individual strings, which is a score based on the chart output. Several chart fitness scores were evaluated, including:

- SPANS The number of spans parsed
- RELATIVE The number of spans the string parsed divided by the string length
- WEIGHTED The sum of the lengths of the spans parsed

See §5.1 for a comparison of these fitness metrics.

Additionally, the following also factored into scoring parses:

- S-BONUS Add an additional bonus to candidates for each sentence they parse completely.
- PSEUDO-SMOOTHING Assign all parses at least a small score, to help avoid premature convergence. The metrics that count singleton spans do this informally.

4 Evaluation

The system was evaluated on a small data-set of examples taken from the World Cup test-bed included with the OpenCCG grammar development system[1] and simplified considerably. This included 19 example sentences with a total of 105 word-types and 613 tokens from (Baldridge, 2002).

In spite of the simplifying assumption that an individual candidate only assigns a single category to a lexical item, one can derive a multi-assignment of categories to lexemes from the population by choosing the top category elected by the candidates. It is on the basis of these derived assignments that the system was evaluated. The examples chosen require only 1-to-1 category assignment, hence the relevant category from the test-bed constitutes the gold standard (minus Baldridge (2002)'s modalities). The baseline for this dataset, assigning np to all lexical items, was 28.6%. The hypothesis is that optimizing

[1] http://openccg.sf.net

Fitness Metric	Accuracy
COUNT	18.5
RELATIVE	22.0
WEIGHTED	20.4

Table 1: Final accuracy of the metrics

parsing coverage with a GA scheme would correlate with improved category-accuracy.

The end-conditions apply if the parsing coverage for the derived grammar exceeds 90%. Such end-conditions generally were not met; otherwise, experiments ran for 100 generations, with a population of 50 candidates. Because of the heavy reliance of GAs on pseudo-random number generation, individual experiments can show idiosyncratic success or failure. To control for this, the experiments were replicated 100 times each. The results presented here are averages over the runs.

5 Results

5.1 Fitness Metrics

The various fitness metrics were each evaluated, and their final accuracies are reported in Table 1. The results were negative, as category accuracy did not approach the baseline. Examining the average system accuracy over time helps illustrate some of the issues involved. Figure 4 shows the growth of category accuracy for each of the metrics. Pathologically, the random assignments at the start of each experiment have better accuracy than after the application of GA techniques.

Figure 5 compares the accuracy of the category assignments to the GA's internal measure of its fitness, using the Count Spans metric as a point of reference. (The fitness metric is scaled for comparison with the accuracy.) While fitness, in the average case, steadily increases, accuracy does not increase with such steadiness and degrades significantly in the early generations.

The intuitive reason for this is that, initially, the random assignment of categories succeeds by chance in many cases, however the likelihood of accurate or even compatible assignments to words that occur adjacent in the examples is fairly low. The GA promotes these assignments over others, apparently committing the candidates to incorrect assignments early on and not recovering from these commitments. The WEIGHTED and RELATIVE metrics are designed to try to overcome these effects by promoting grammars that parse longer spans, but they do not succeed. Perhaps exponential rather than linear bonus for parsing spans of length greater than two would be effective.

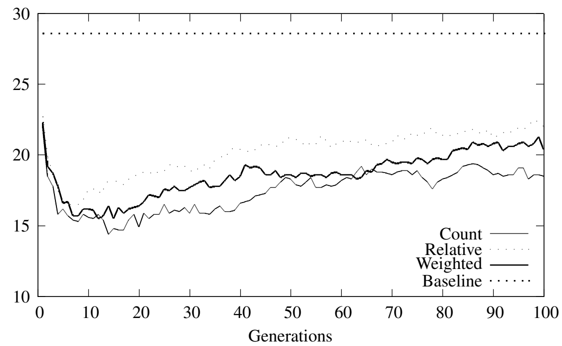

Figure 4: Comparison of fitness metrics

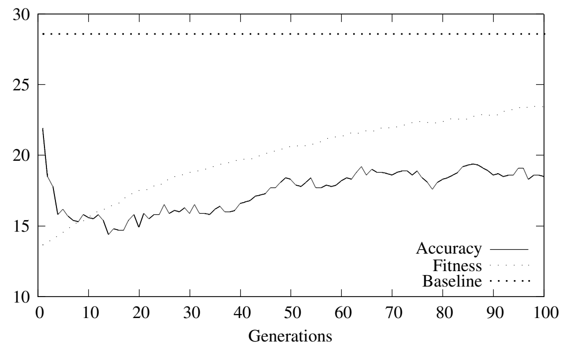

Figure 5: Fitness and accuracy: COUNT

6 Conclusions

This project attempts to induce a grammar from unannotated material, which is an extremely difficult problem for computational linguistics. Without access to training material, logical forms, or other relevant features to aid in the induction, the system attempts to learn from string patterns alone. Using GAs may aid in this process, but, in general, induction from string patterns alone takes much larger data-sets than the one discussed here.

The GA presented here takes a global perspective on the progress of the candidates, in that the individual categories assigned to the individual words are not evaluated directly, but rather as members of candidates that are scored. For a system such as this to take advantage of the patterns that arise out of the text itself, a much more fine-grained perspective is necessary, since the performance of individual category-assignments to words being the focus of the task.

7 Acknowledgements

I would like to thank Jason Baldridge, Greg Kobele, Mark Steedman, and the anonymous reviewers for the ACL Student Research Workshop for valuable feedback and discussion.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Ted Briscoe. 2000. Grammatical acquisition: Inductive bias and coevolution of language and the language acquisition device. *Language*, 76:245–296.

Stephen Clark and James R Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of EMNLP-03*, pages 97–105, Sapporo, Japan.

Robin Clark. 1996. Complexity and the induction of Tree Adjoining Grammars. Unpublished manuscript, University of Pennsylvania.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL-97*, pages 16–23, Madrid, Spain.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of ACL-96*, pages 79–86, Santa Cruz, USA.

David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Julia Hockenmaier and Mark Steedman. 2001. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of ACL*, pages 335–342, Philadelphia, USA.

Julia Hockenmaier and Mark Steedman. 2005. CCG-bank: User's manual. Technical Report MC-SIC-05-09, Department of Computer and Information Science, University of Pennsylvania.

Aravind Joshi. 1985. An introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins.

Emin Erkan Korkmaz and Göktürk Üçoluk. 2001. Genetic programming for grammar induction. In *2001 Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 245–251, San Francisco, USA.

Rober M. Losee. 2000. Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. *Information Processing and Management*, 32:185–197.

Tony C. Smith and Ian H. Witten. 1995. A genetic algorithm for the induction of natural language grammars. In *Proc. of IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*, pages 17–24, Montreal, Canada.

Mark Steedman. 2000. *The Syntactic Process*. MIT, Cambridge, Mass.

Aline Villavicencio. 2002. *The Acquisition of a Unification-Based Generalised Categorial Grammar*. Ph.D. thesis, University of Cambridge.

Stephen Watkinson and Suresh Manandhar. 2000. Unsupervised lexical learning with categorial grammars using the LLL corpus. In James Cussens and Sašo Džeroski, editors, *Language Learning in Logic*, pages 16–27, Berlin. Springer.