EVOLUTIONARY METHODS FOR DESIGN, OPTIMISATION AND CONTROL
K. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou and T. Fogarty (Eds.)

LEARNING WEIGHTS FOR A NATURAL LANGUAGE GRAMMAR USING GENETIC ALGORITHMS

Ingo Schröder[(a)], Horia F. Pop[(a,b)⋆]
(a) Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Strasse 30
22527 Hamburg, Germany
e-mail: schroeder|hfpop|menzel|foth@informatik.uni-hamburg.de

Wolfgang Menzel[(a)], Kilian A. Foth[(a)]
(b) Department of Computer Science
Faculty of Mathematics and Computer Science
"Babeş-Bolyai" University
1, M. Kogălniceanu Street
3400 Cluj-Napoca, Romania
e-mail: hfpop@cs.ubbcluj.ro

Abstract. Experiments in learning the weights of a natural language grammar by means of genetic algorithms are described. The results suggest that a manually prepared grammar can be improved, both in terms of quality and parsing time. Additionally, a good grammar can be derived from an almost random initial assignment.

Key words: Constraint Dependency Grammar, Constraints Satisfaction, Genetic Algorithms

1 Introduction

The application of soft computing techniques to natural language processing has a number of advantages, namely the increased robustness against non-standard utterances and the ability to accomodate conflicting information such as preferences or defaults in the model. On the other hand, they introduce the problem of finding appropriate values for the free parameters by means of a suitably designed training procedure. Probabilistic approaches typically compute their parameters by means of well-established estimation procedures independently of the probabilistic processing component itself, often using frequency counts from a set of training data. Other approaches that are based on alternative measures require qualitatively different training methods to optimize their behaviour on a given data sample. In particular, since no independent measure of model quality is available, they must allow the inclusion of the processing model into the optimization procedure.

The current paper presents an effort to improve a Weighted Constraint Dependency Grammar[1] using an evolutionary approach aimed at optimizing the parsing performance on a given training corpus by learning the constraint weights.

2 Constraint-based grammar formalism

The formalism of Weighted Constraint Dependency Grammars[1] (WCDG) maps natural language utterances to tree-like configurations that describe the underlying structures of a sentence, e. g., its hierarchical organization and aspects of its meaning. A WCDG consists of constraints that describe what a valid analysis should look like. Associated weights determine how important a grammatical condition represented by a constraint is. So far, not only the constraints have been designed manually; the weights were also assigned by a grammar writer, which is a labor-intensive and error-prone process.

The constraints are divided into hard constraints, which may not be violated, and soft constraints, characterized by non-zero weights, which may be violated (if unavoidable). These soft constraints allow the grammar to accept utterances that are not completely grammatical while ranking those parses that satisfy the hard constraints, thus facilitating a complete disambiguation.

CDGs have been used for different tasks: interactive machine translation,[2] speech recognition,[3] robust parsing and error diagnosis.[4,5] Additionally, WCDGs facilitate incremental parsing,[6] and have great potential for time-adaptive behavior.

3 Evolutionary-based grammar learning

A procedure which finds the weights for the constraints automatically is highly desirable because the interactions among different constraints are barely foreseeable and, therefore, human grammar writers cannot be expected to find near-optimal weights.

With several hundred free parameters, the space of possible sets of weights is too large to be explored systematically by traditional search methods. Evolutionary techniques, on the other hand, visit only a small portion of the search space by selecting only a small number of candidates from which new candidates are derived. At the same time, random jumps allow for the exploration of different parts of the space. Therefore, evolutionary algorithms seem to be promising methods for the problem at hand.

The genetic algorithms used in our experiments have the following general setup: The fitness function is defined as a weighted mean of the f-measure (80%) and a time index (20%). The f-measure is an aggregated measure for the quality of a parsing solution, while the time index is a measure for parsing speed, with values between zero (no time used) and one (the maximum amount of time used). Each genetic individual represents the set of constraint weights corresponding to one particular grammar, with an individual weight corresponding to a single gene. Different genetic operators have been used: multipoint and arithmetic crossover, relative and absolute mutation and mutation-to-zero, with probabilities of p_{mc}, p_{ac}, p_r, p_a, p_z respectively.

For each pair of individuals, the multipoint crossover operator generates two new combined individuals. After generating a random number of crossover points, the new individuals are generated by keeping genes from the parents and switching to the other parent when reaching a crossover point.

The arithmetic crossover operator generates a single individual by computing its genes as the weighted arithmetic mean of the genes of its parents, with the weight of the mean being randomly generated.

The mutation operators work at gene level. The relative mutation operator increases or decreases the value of a gene based on a randomly generated value. The absolute mutation operator replaces the current gene value with a randomly generated value. The mutation-to-zero operator simply sets the current weight to zero. Recall that a weight of zero means that the associated constraint must not be violated in any case. We added this special mutation-to-zero operator in order to take this special meaning into account.

The selection scheme for the next generation works as follows: Include the five fittest individuals, discard the five worst individuals and fill up the new population by randomly selecting individuals based on their fitness measure.

4 Experiments

We have used this evolutionary computing scheme with the aim to improve a manually designed grammar and to generate a grammar from a semi-random initial assignment. Figure 1a presents the results of improving the weights of a carefully designed manual grammar. The initial generation was formed by random variations of the manual grammar, and the following experimental parameters were used: $p_{ac} = 0.25$, $p_{mc} = 0.25$, $p_r = 0.02$, $p_a = 0.002$, $p_z = 0.002$.

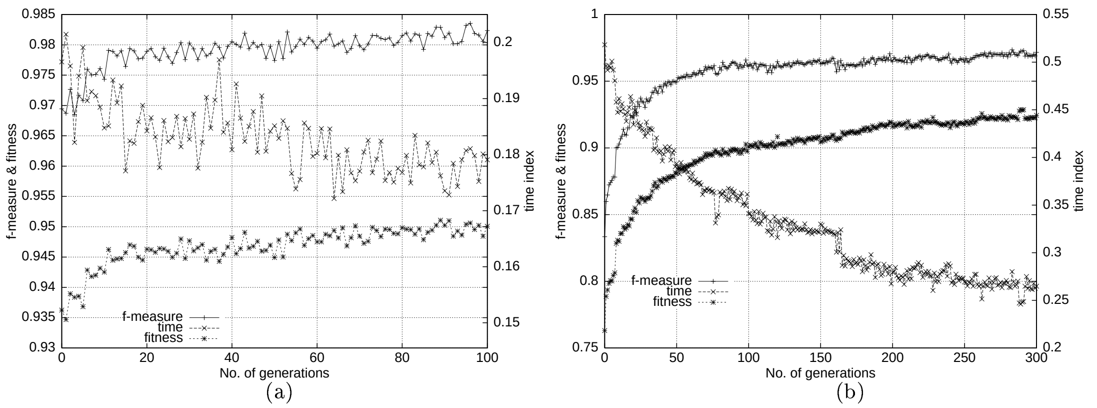


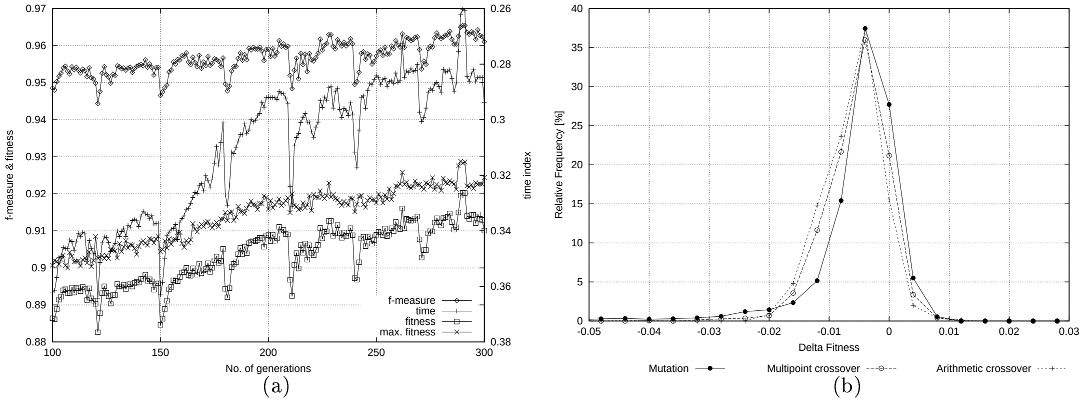
Figure 1: F-measure, time and fitness of the best individual per generation for (a) the first experiment; (b) the second experiment

It can be observed that small but noticable improvements could be achieved. The fitness (f-measure) increased from 93.6% (96.9%) to 95.1% (98.4%). The time needed slightly decreased.

Figure 1b presents the results for the second experiment where the constraints that were soft in the manual grammar got a random initial weight while the originally hard ones were kept at zero. The following experimental parameters were used: $p_{ac} = 0.25$, $p_{mc} = 0.25$, $p_r = 0.02$, $p_a = 0.0$, $p_z = 0.0$. In this experiment a new perturbation scheme has been used in order to escape from long periods of inactivity. In every 30th generation starting at generation 120, the population has completely been replaced by a new population built by random variations of the best individual.

Similar, but (due to the random starting point) clearer observations can be made. The fitness (f-measure) increased by 16.5 (11.1) percent points and the processing time decreased by 52.5%.

Figure 2a presents the average f-measure, time and fitness for the individuals produced at each generation, in the last part of the second experiment. The trend of the average fitness is increasing in a similar manner as the trend of the maximum fitness (see Figure 1b) which confirms that the results are not just a coincidence of the single run. The alternative of re-running the same experiment multiple times



(a) (b)

Figure 2: (a) Average f-measure, time and fitness, and (b) relative frequencies of the differences in fitness between parent(s) and new individual depending on applied operator

was not available due to time constraints.

Figure 2b presents the relative frequencies of the differences between the fitness values of the parent(s) and the produced individual (partitioned into classes) depending on the applied operator. It can be seen that the mutation operator is more effective than both types of crossover and that multipoint crossover (slightly) outperforms arithmetic crossover. This effect shows even clearer during the experiment where most (but not all) best new individuals result from mutation not only because mutation is superior to the other operators but also because many more individuals are generated by mutation (7941) than by arithmetic crossover (748) or multipoint crossover (1426). This ranking of effectiveness has also proven stable over time.

5 Results validation

The produced grammars have been tested on an extra set of 90 utterances. On this testing set, the manual grammar has a fitness of 91.5% and the optimized grammar produced by the first experiment has a fitness of 92.3%. The second experiment started with a grammar having the fitness (f-measure) of 77.2% (84.3%) and produced a grammar with a fitness of 91.5% (96.0%) (see Figure 3). It can be concluded that similar although slightly worse (compared to the training set) results have been obtained.

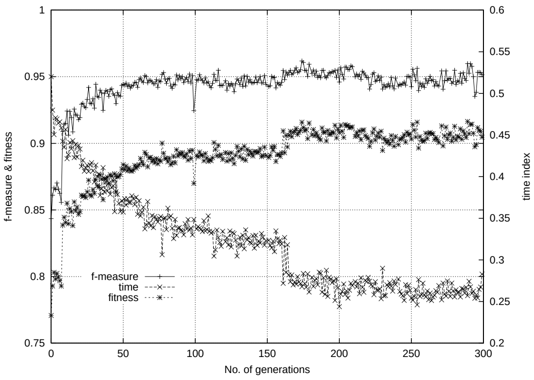


Figure 3: F-measure, time and fitness of best individuals from each generation of second experiment, evaluated on the test set

A different validation refered to test set variations: the test set was randomly split into two subsets. Both subsets showed exactly the same tendencies with variations within a three percent margin. Even if a verification of these results on a larger test set is desirable, this validation confirms the robustness of the procedure.

6 Conclusions

Genetic algorithms, based on the principle of "survival of the fittest" have been used to learn the weights of a constraint dependency grammar from a corpus of annotated utterances. The experiments have demonstrated a remarkable potential of evolutionary techniques for improving existing grammars, both in terms of parsing quality and speed-up in processing time. These results have been obtained not only for a manually weighted initial grammar but also for an initial grammar with randomly assigned soft constraint weights. Although the initial grammar represented a considerable amount of human work, a small but noticeable increase of quality and time has been achieved for the first case. In the second case, a level of performance was reached close to the initial hand-tuned grammar.

These results, confirmed on an independent test set, show that genetic algorithms may be used as a fully automatic substitute for this difficult subtask of grammar development. However, more experiments will be necessary to further improve the behaviour of the algorithm, such that, ideally, it would be possible to converge to the same level of quality, irrespective of the weights' initial assignment.

REFERENCES

[1] Johannes Heinecke, Jürgen Kunze, Wolfgang Menzel, and Ingo Schröder. Eliminative parsing with graded constraints. In *Proceedings of the Joint Conference COLING/ACL-98*, pages 526–530, Montréal, Canada. (1998).

[2] Hiroshi Maruyama. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Annual Meeting of the Association of Computational Linguistics (ACL-1990)*, pages 31–38, Pittsburgh, PA, USA. (1990).

[3] Mary P. Harper, C. M. White, Wen Wang, Michael T. Johnson, and Randall A. Helzerman. The effectiveness of corpus-induced dependency grammars for post-processing speech. In *Proceedings of the First Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pages 102–109, Seattle, WA, USA. (2000).

[4] Wolfgang Menzel and Ingo Schröder. Model-based diagnosis under structural uncertainty. In Henry Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-1998)*, pages 284–288, Brighton, UK. (1998).

[5] Wolfgang Menzel and Ingo Schröder. Error diagnosis in a language tutoring system. *Special ReCALL publication: Language Processing in CALL*, , 20–30, (1999).

[6] Kilian Foth, Wolfgang Menzel, Horia F. Pop, and Ingo Schröder. An experiment on incremental analysis using robust parsing techniques. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, pages 1026–1030, Saarbrücken, Germany. (2000).