Non-directionality and Self-Assessment in an Example-based System Using Genetic Algorithms

Yves Lepage
Bahagian Sains Komputer, PP Sains Matematik & Sains Komputer, Universiti Sains Malaysia, 11800 Penang, Malaysia
e-mail: lepage@cs.usm.jp
research done at ATR Interpreting Telecommunications Research Labs, Hikari-dai 2-2, Saika-cho, Soraku-gun, 619-02 Kyoto, Japan

Abstract
We show the application of an optimisation technique to natural language processing: genetic algorithms, thanks to the definition of a data structure called board and a formal distance. The system has two interesting features: non-directionality, which is more than bi-directionality, and self-assessment, independently of the inner knowledge. Results of experiments are presented and discussed.

Topical paper: Software for NLP

Introduction
The purpose of this article is to show that an engine based on an optimisation technique, namely genetic algorithms, can perform NLP tasks: analysis and generation in the frame of example-based approaches. But more than that, the system we have built has interesting properties:

- it is truly non-directional, i.e. it performs more than bi-directional tasks;
- it evaluates its results relatively to the input, and not relatively to its internal knowledge.

Two original facts make this possible:

- the definition of a data structure, called board which is in essence bidirectional. It is the association of a sentence pattern and its linguistic structure;
- the definition of a distance on this data structure.

1 Motivations
1.1 Non-directionality
Our main motivation was to design a system where not only the formalism but also the system engine is bi-directional. In [Lepage 91], we sketched such an engine for a rule-based approach.

From a theoretical point of view, such a system is also more interesting than a system where, although the formalism would be bi-directional, analysis and generation would be separate modules resulting from different compilations.

In our sketch, a more general property than bi-directionality emerged: non-directionality. Bi-directionality is just the property of executing analysis and generation with the same set of specifications, whereas non-directionality allows more: a complete sentence and its complete analysis can be built from a partial specification of the sentence and a partial description of the associated structure.

1.2 Self-assessment
A second motivation lies in a flaw of rule-based systems using context-free parsers, which is that they often fail to deliver a solution for trivial reasons such as a word missing in a dictionary. On the contrary, our system always delivers an output for any input, would the solution be "bad". Of course, this would be of no meaning if the quality of outputs would not be evaluated. Hence, when delivering a solution, the system scores it.

Some machine translation systems viewed as expert systems may return an evaluation of their work in terms of their knowledge (grammar) [Tong 89], some other may evaluate the result according to thesaurus classification and statistical frequencies [Furuse and Iida 92b], but all these methods are specific. Here, on the contrary, the system delivers a score which is a formal distance between the input and the output. Thus, it is independent of the linguistic representation chosen (dependency or constituency). This is not the case of a proposal such as [Harrison et al. 91].

This score is a possible answer to the serious lack of assessment in natural language processing, as it may apply to any other system, leading to reliable comparisons of intrinsic performances.

2 Realisation
Genetic algorithms constitute a possible answer to the previous motivations. They are a collection of techniques for approaching the solution of optimisation problems [Goldberg 89].

On the contrary to usual programming techniques which handle only one object at a time, genetic algorithms deal with a collection of individuals, called a population. For each individual, one can compute a function, called the fitness function. Those individuals for which the fitness function is optimum are the best individuals.

From two individuals, one can derive two new individuals by cutting them into two pieces and gluing the
pieces back in the way illustrated in Figure 1. This is crossover. Some random modification of the children may occur, accounting for mutation to complete the genetic metaphor.

![Figure 1: Principle of crossover](image)

The previous operation can be repeated over a population a number of times so that populations follow one another. In the last generation, the best individuals are hopefully solutions of the optimisation problem at hand. In order to apply genetic algorithms to natural language processing, one has to determine:

- which data has to play the role of individuals;
- consequently, what a population will be;
- for an individual, what its fitness is;
- in a population, how individuals are selected for crossover and how it is performed;
- how analysis and generation can be viewed as optimisation problems on a population.

The meeting of our research interests with genetic algorithms is a consequence of various available results.

Firstly, the need for linguistic specification of computer grammars led to the proposal of an original data structure called board. This data structure is neutral with respect to analysis and generation. It will play the role of individuals. Boards rely on the data structure of forests, for which it is possible to define a crossover method.

Secondly, the stream of example-based machine translation is now well-established and justifies considering a collection of already parsed sentences as a population [Sadler and Vendelman 90], [Sato and Nagao 90].

Finally, our recent work on distances, and especially the fine-grained correspondences between different parts of the texts and of the structure [Boitet and Zaharin 88], [Zaharin and Lepage 92], as a particular case, projective constituency boards such as Figure 2 verify these constraints.

Boards would be of little use if they would not allow the description of patterns. Hence, Figure 3 is also a valid board. It is similar to Figure 2, except that portions of the string and the tree have been replaced by variables (prefixed by a $ sign). These variables stand for forests, not only for trees - the point is important. Because it is always better to look for a unified view of objects, the string part and the tree part are considered to be of the same data type, that of forest. As a matter of fact, a string is a forest with only one level, and a tree is a forest with only one node on the highest level. Now, as forests are the underlying data type, variables stand naturally for subforests. On the string side, considering variables as forests is far more interesting than if they would instantiate with one word only.

An interesting property about the board data structure, and it is exactly why it has been devised, is that, because it is the association of a string (the text) and a (linguistic) tree, it is neutral with respect to the main natural language processing operations:

- analysis (input: string, output: tree);
- generation (input: tree, output: string).

### 2.1.2 Population = Data base of boards

Our database of sentences is that of ATR telephone conversations. These dialogues are telephone conversations in English. This represents 255 sentences of which 150 are different.

- Hello.
- This is the Conference Office.
- Could you tell me about the attendance fee for the conference? If I apply for the conference now, how much is the attendance fee?
- Yes. At present the attendance fee is $35,000 yen per person. If you apply next month, it will be $40,000 yen.

![Figure 4: An excerpt from the ATR dialogues](image)
The linguistic structures corresponding to the previous sentences have been drawn by hand and scrupulously reviewed to ensure consistency. They are syntactic constituency trees and are exactly projective, which means that each leaf in the tree corresponds to a word in the sentence in the same order.

As for illustration, all the trees and sentences in this paper are extracted from our database of boards. Some representational choices have been made to limit the number of morpho-syntactic categories to 14 (and phrase types to 7) and to keep projectivity by all means.

2.2 Functions

2.2.1 Fitness = Distance between forests

We define the fitness of an element in a population (set of boards) as the distance to a given input (a board) to the system. In other words, we have to define a distance between boards. A simple idea is to take the sum of the distances between the strings on the one hand, and the trees on the other hand. As strings and trees are forests, a distance on forests is required.

The definition of a distance on forests is given below, with a, b being nodes, u, u', v, v' being forests and denoting concatenation of forests.

\[
\begin{align*}
\text{dist}(a(u'), u, b(v'), v) &= \min \left( \text{dist}(a(u'), b(v'), v), \text{dist}(a(u'), u, b(v'), v) \right) \\
\text{dist}(a(u'), u, b(v')) &= \text{dist}(a(u'), u) + \text{dist}(u, b(v')) \\
\text{dist}(a(u'), b(v')) &= \text{dist}(a(u'), b) + \text{dist}(u', v') \\
\text{dist}(a, b) &= 0 \quad \text{if } a = b \\
\text{dist}(a, e) &= 1 \quad \text{insertion} \\
\text{dist}(a, e) &= 1 \quad \text{deletion}
\end{align*}
\]

It is a direct generalisation of two classical distances on strings [Wagner & Fischer 74] and trees [Selkow 77]. Both distances answer the correction problem: what is the minimal number of typing operations needed to transform one object into the other one? In both distances and their generalisation to forests, the typing operations are insertion, deletion and replacement.

An extension of the previous distance to forest patterns (i.e. forests containing variables) has been presented in [Lepage et al. 92]. It is no longer a metric, so we call it a proximity score. With this score, the distance between a variable and a constant object is zero by definition. Figure 5 gives an illustration (the unit is a one word or node difference).

2.2.2 Crossover = Exchange of subforests

We turn now to crossover. The first question is how boards are selected in a population for crossover.

It seems reasonable that those individuals with better fitness value should intervene more in the production of the next generation. Along this line, the simple following law gives the probability of a board i with fitness \( f_i \) (some reciprocal of distance) to be selected for crossover:

\[
p_i = \frac{f_i}{\sum_i f_i}
\]

As for crossover itself, it has to be defined on strings and on trees.

On strings, be they chromosomes or sequences of bits, crossover is generally performed as illustrated in Figure 1. We could translate crossover sentences following this simple principle (see Figure 6).

But we insist on keeping the unity of data structure between strings and trees. So, we translate string crossover into forest terms: it is the exchange of the sister forests of the crossover points. This can be applied directly to trees, see Figure 7. This technique is different from the exchange of subtrees as proposed in [Koza 92].

Now, by keeping projectivity during crossover, only corresponding parts of strings and trees will be exchanged. As a consequence, string crossover will allow exchange of inner substrings. To sum up, a board obtained by crossover will give a partially valid description of a possibly ungrammatical sentence (see Figure 8).

2.2.3 Optimisation problem = Closest board in database

The system built for the experiment implements a simple genetic algorithm. The starting population is a set of example boards, i.e. complete sentences with their complete associated linguistic structures.
If an input board is given to the system, each board in the data base of examples can be assigned a fitness score: its distance to the input board.

- When the input is a board where the linguistic tree is unknown (a variable), the output will be the closest board containing the closest sentence with its associated tree. This is a kind of analysis.

- When the input is a board where the string is unknown (a variable), the output will be the closest board containing the closest tree with its associated string. This is a kind of generation.

- When the input is a board where both the sentence and the linguistic tree are partially specified (they contain variables), the output will be the closest board containing a complete sentence and its complete associated linguistic structure.

We call the last operation non-directional completion. In fact, analysis and generation are only particular cases of this operation. For instance, analysis is non-directional completion for a board with no variable in the string part, and a variable as the tree part.

For each operation above, the external behaviour of the system may be considered different, although the internal behaviour is exactly the same. In any case, the output is a board, built from pieces of the data base boards, and minimising the distance to the input. It is important to stress the point that the input never enters the data base of boards. It is only used to compute the fitness of each board in the data base in each generation. Figure 9 summarises the system and its functioning.

3 Experimentation

3.1 Experiments

We tested the performance of the system for analysis, generation and non-directional completion.

For analysis, a board is extracted from the data base (call it reference board). A new board is built by associating the string part of the reference board with a variable as its tree part. It becomes the input to the system. Of course, the reference board is eliminated from the database.

A first measure is given by the system itself: it is the fitness of the output, which is the distance between the output and the input. A second measure is the distance between the output and the reference board, which reflects the absolute quality of the output. Moreover, runtimes have been measured.

This procedure was carried out for each board of the data base so that average values could be computed. There were 225 boards in the data base.
For generation, the same procedure was applied, but, of course, the tree part is kept in building the input board. For non-directional completion, an incomplete board is automatically built by inserting variables at random positions in the string and tree parts of the reference board.

3.2 Results

Analysis Analysis gives an average error of about 9.2 elements relatively to the exact output after thirty generations. The average number of elements (nodes and words) in a board is 24.5, hence, the error rate is 38%, not a very good result. The fitness gives the average number of words wrong in the average string output by the system: around 3.2 words for a 8.5 word-long sentence.

| generation | fitness | quality | time (in sec.) |
|------------|---------|---------|----------------|
| 3          | 4.8     | 11.97   | 8.53           |
| 6          | 4.48    | 13.69   | 8.64           |
| 9          | 4.52    | 13.52   | 10.89          |
| 12         | 4.66    | 13.50   | 13.26          |
| 15         | 4.66    | 13.52   | 15.67          |
| 18         | 4.13    | 12.97   | 15.82          |
| 21         | 3.97    | 11.39   | 16.79          |
| 24         | 3.98    | 10.62   | 17.50          |
| 27         | 3.23    | 9.24    | 16.63          |

Generation Generation is performed with better results than analysis. The average error in the tree only is 1.1 node for 16 node-heavy trees and the absolute error rate falls to 12%. However, as expected, generation is slower than analysis: because more tree distance computations are performed.

| generation | fitness | quality | time (in sec.) |
|------------|---------|---------|----------------|
| 3          | 4.8     | 9.49    | 8.12           |
| 6          | 4.73    | 9.39    | 14.86          |
| 9          | 3.92    | 8.35    | 16.90          |
| 12         | 3.56    | 7.35    | 15.56          |
| 15         | 2.89    | 6.35    | 18.95          |
| 18         | 2.81    | 5.35    | 18.99          |
| 21         | 1.94    | 4.35    | 18.71          |
| 24         | 1.48    | 3.35    | 19.54          |
| 27         | 1.39    | 3.21    | 20.94          |
| 30         | 1.14    | 2.26    | 22.61          |

Non-directional completion The following results must be considered as purely illustrative, because the form of boards for non-directional completion is unrestricted. As could be expected, because no pre-complete in the input, quality is worse than for analysis and generation, although fitness appears to be quite good.

| generation | fitness | quality | time (in sec.) |
|------------|---------|---------|----------------|
| 3          | 4.8     | 14.69   | 7.49           |
| 6          | 5.81    | 14.64   | 12.69          |
| 9          | 4.32    | 13.77   | 15.37          |
| 12         | 4.92    | 13.32   | 17.00          |
| 15         | 3.68    | 13.95   | 17.33          |
| 18         | 2.72    | 12.98   | 18.97          |
| 21         | 1.50    | 13.96   | 18.95          |
| 24         | 1.39    | 13.25   | 20.93          |
| 27         | 1.24    | 13.01   | 21.47          |
| 30         | 1.10    | 12.68   | 22.30          |

3.3 Discussion

We will now discuss the advantages and drawbacks of our system.

3.3.1 Non-directionality

The general function of the system is to build a complete sentence and its complete associated syntactic tree from a partially specified sentence and a partially specified tree. Hence, analysis and generation turn out to be only particular cases of this general operation. This feature is what we called non-directionality. It is more general than bi-directionality. Until now, we are not aware of any natural language processing system having this property.

From the applications point of view, non-directionality allows one to envisage linguistically founded editing operations. For example, suppose we would like to replace refund the fee by pay the fee back all over a text. We would like the operation to apply for any tense of the verb. The following board could be used to retrieve all possible candidates. It says that we want a verbal phrase (structural constraint) and that the substring for must appear (string constraint). Of course, to perform such an operation, we would not advise the use of genetic algorithms...

3.3.2 Assessment

Because parts of the input may be modified in the output, assessment is necessary. The system delivers a score which is not directly connected to the knowledge of the system. It is the distance between the input and the output. Minimising this distance is precisely the task of the system. As this score is a theoretical metric between structures, it is not stuck to a particular representation. It could be applied to evaluate similar systems using different representations, for example dependency structures.

3.3.3 Normalising effect

Despite the previous points, important criticisms can still be addressed to the current system.

Experiments carried out with input sentences from outside the data base have shown that the system has a normalising effect: outputs are cast to resemble sentences and trees from the database. This is a negative effect if a free-input system is wanted. But, if a large enough data base is built and if standardisation is required, as is the case with technical documents in many companies, this may be seen as a positive feature.

3.3.4 Computational limits

A classical criticism of genetic algorithm concerns heavy computation time. Here, it is proportional to the number of examples in the data base! This prevents us from using a big enough data base for any reasonable-size application.

As for space, our implementation of the system relies on a toolbox which makes extensive use of dynamic programming (storing intermediate results to increase speed). Memory size limits are rapidly reached. In this
implementation, the space-time trade-off is a sensitive issue.

To remedy both these problems we are envisaging porting our programs on a parallel machine. This does not add anything from the theoretical point of view, but genetic algorithms are obvious candidates for parallelisation.

**Conclusion**

This paper has described the application of an optimisation technique to natural language processing tasks, i.e. analysis and generation. The system has been shown to have two interesting properties: non-directionally, which is more than bi-directionality and self-assessment, independently of its internal knowledge.

**References**

[Boitet and Zaharin 88] Christian Boitet and Zaharin Yusoff

Representation trees and string-tree correspondences

*Proceedings of COLING-88*, pp 59-64, Budapest, 1988.

[Goldberg 89] David E. Goldberg

*Genetic Algorithms in Search, Optimisation, and Machine Learning*

Addison Wesley Publishing Company, 1989.

[Harrison et al. 91] P. Harrison, S. Abney, E. Black, D. Flickenger, C. Gdaniec, R. Grishman, D. Hindle, R. Ingrina, M. Marcus, B. Santorini, T. Strzalkowski

Evaluating Syntax Performance of Parser / Grammars of English

*Proceedings of the Workshop on Evaluating Natural Language Processing Systems, ACL*, 1991, pp. 7-7.

[Furuse and Iida 92b] Furuse Osamu and Iida Hitoshi

An Example-based Method for Transfer-driven Machine Translation

*Proceedings of the fourth International Conference on Theoretical and Methodological Issues in Machine Translation, TMI-92*, pp 139-156, Montréal, 1992.

[Koza 92] John R. Koza

*Genetic Programming - On the Programming of Computers by Means of Natural Selection*

MIT Press, 1992.

[Lepage et al. 92] Yves Lepage

*Generating Context-Sensitive Languages with Correspondence Identification Grammars*

*Proceedings of the Pacific Rim Symposium on Natural Language Processing, Singapore*, November 1991, pp. 258-263.

[Sadler and Vendelhams 90] Victor Sadler and Ronald Vendelhams

Pilot implementation of a bilingual knowledge bank

*Proceedings of Coling-90, Helsinki*, 1990, vol 3, pp. 449-451.

[Sato and Nagao 90] Sato Satoshi and Nagao Makoto

Toward Memory-based Translation

*Proceedings of Coling-90, Helsinki*, 1990, vol 3, pp. 247-253.

[Selkow 77] Stanley M. Selkow

The Tree-to-Tree Editing Problem

*Information Processing Letters*, Vol. 6, No. 6, December 1977, pp. 184-186.

[Tong 89] Tong Loong Cheong

A data-driven control strategy for grammar writing systems

*Machine Translation, 4(1)*, December 1989, pp. 177-193.

[Vauquois and Chappuy 85] Bernard Vauquois and Sylviane Chappuy

Static grammars: a formalism for the description of linguistic models

*Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation*, Calgate University, New York, August 1985, pp 298-322.

[Wagner & Fischer 74] Robert A. Wagner and Michael J. Fischer

The String-to-String Correction Problem

*Journal for the Association of Computing Machinery*, Vol. 21, No. 1, January 1974, pp. 168-173.

[Zaharin and Lepage 92] Zaharin Yusoff and Yves Lepage

On the specification of abstract linguistic structures in formalisms for Machine Translation

*Proceedings of the International Symposium on Natural Language Understanding and AI*, pp 145-153, Izuka, July 1992.