Applying Genetic Algorithms to Referring Expression Generation and Aggregation

Raquel Hervás and Pablo Gervás

Departamento de Sistemas Informáticos y Programación,
Universidad Complutense de Madrid, Spain
raquelhb@fdi.ucm.es,pgervas@sip.ucm.es

1. Introduction

Natural Language Generation (NLG) includes complex tasks [2], and features such as ambiguity can make these tasks too complex in fact to be tackled with classic search algorithms in a reasonable time. Evolutionary Algorithms (EAs) are heuristic techniques [1] based on a search model which emulates evolutionary phenomena, and that can be used to speed up the search. The purpose of most EAs is to find a good solution but not necessarily the best solution, and this is enough for most natural languages processes. In this paper we present a first approach to the idea of using NLG and EAs together.

The specific problems addressed in this paper are two subtasks of the process of automatically generating a natural language text for a given content: referringf expresion generation and aggregation. The nature of these subtasks is best illustrated over an example:

```
   A princess lived in a castle. She loved a brave knight. She was
pretty. The castle had towers. It was strong. They lived in it.
```

In this text various elements of the reality that is being described are mentioned one or more times: a princess, a knight,... Different natural language expressions are used to describe them at each occurrence in the text: "*a princess*", "*she*", "*the princess*". The task of referring expression generation concers the way in which a system decides at each occurrence how to refer to a given element.

The correct use of referring expressions to compete with human generated texts involves a certain difficulty [4]. Possible simple algorithms for deciding when to use a pronoun and when to use the full name produce poor results. Two occurrences of the same concept in a paragraph can be far apart, and this may confuse the reader. Knowledge intensive approaches modelled on the way humans do it require a certain measure of content understanding that is resource hungry.

Aggregation [3] involves deciding how compact the presentation of information should be in a given text. It operates at several linguistic levels, but we only consider it here with respect to concepts and their attributes. For instance, the system must decide between generating "*The princess is blonde. She sleeps.*" and generating "*The blonde princess sleeps.*". Aggregation is generally desirable, but may result in adjective-heavy texts when the information to impart becomes dense in terms of attributes, as in "*The pretty blonde princess lived in a strong fancy castle with her* *stern rich parents.*". A good solution should find a balance between conciseness and acceptable style.

2 An Evolutionary Solution

We propose the use of a simple genetic algorithm, where the appearances of the concepts are considered as the genes. The initial population is generated randomly, using for each concept its full name or its pronoun. When using the full name, a selection of the attributes the concept has in the knowledge base is chosen. These attributes will appear just before the name of the concept, as it is usual in English. The system works over this population for a number of generations determined by the user. In each generation three genetic operators are used: crossover, mutation and aggregation. Finally, at the end of each generation each tale is evaluated and a selection of the population is passed to the next one, in such way that the tales with a higher fitness have more possibilities of being chosen.

The operators

For the *crossover operator*, two drafts are selected randomly and crossed by a random point of their structure. So, each of the sons will have part of each of the parents.

In the case of the *mutation operator*, some of the genes are chosen randomly to be mutated. If the gene is a pronoun, it will change into the correspondent full name, always associated with a subset of its possible attributes. In case the gene was a full name, there are two options: to change it into a pronoun, or to change the subset of attributes that appear with it. One of these two options is chosen randomly.

The *aggregation operator* addresses the task of deciding on the aggregation between concepts and their attributes. This involves a certain modification of the structure of the text, because sentences in the text may be deleted if the information they impart becomes part of a previous sentence. The aggregation operator acts only on genes corresponding to explicitly mentioned concepts: concepts referred by pronouns are excluded. It can act in two directions. If the reference to the concept appears with one or more attributes – as in *"A blonde princess lived in a castle."* -, the operator disggregates the attributes by eliminating their mention and adding a corresponding "X is Y" sentence – resulting in *"A princess lived in a castle. She was blonde."*. If the reference to X has no attributes – as in *"A princess lived in a castle."* -, the algorithm looks for an "X is Y" sentence – such as *"The princess was blonde."*-, adds the corresponding attributes to the reference, and deletes the "X is Y" sentence – resulting in *"A blonde princess lived in a castle."*.

The goal of this definition of the aggregation is to ensure that the attributes of a concept be mentioned in the appearance of a concept or in the correspondent "X is Y" sentences, but not in both. As the aggregation operator are used randomly, the desired result is obtained only in some cases.

The fitness function

The key to the genetic algorithm lies in the choice of fitness function. A simple approach would be to require that in each generation the user reads all the texts and gives them a fitness value. The number of generations and individuals in the population for a simple experiment makes this approach impractical. So, we analyzed the features of a human generated text from the point of view of the referring expressions, and we found five different evaluation functions, each one aimed at a different aspect.

Correct Referent. When writing a text, we cannot use a pronoun for something that we have not mentioned before, or readers would get confused. In addition, if the full name reference and the pronoun are far, the reader can also get confused and be unable to link the two occurrences of the same concept. So, this function penalizes the appearances of pronouns for a concept that has not been referred in the two previous genes with its full name.

Redundant Attributes. When describing a concept in an "X is Y" sentence, people do not use the attribute they are going to describe in the reference to the concept. This function penalizes sentences like *"The <u>blonde</u> princess was <u>blonde</u>"*.

Reference Repetition. Using always the same reference together with the same attributes results in repetitive text. For example, it is acceptable to use "the princess" every time we refer to the princess character, but it would be striking to use always "the pretty blonde princess". To avoid that, repetitive use of references is penalized.

Coherence. If we use different subsets of its attributes in different references to the same concept, the reader may mistakenly assume that we are referring to different concepts. For example, if we use "the pretty princess" and "the blonde princess" in different places, and we have not specified that the princess is both pretty and blonde, it could seem that there are two princess: a pretty one and a blonde one. With this function, we penalized the appearance of attributes that have not been mentioned in the first description of a concept or in a "X is Y" sentence.

Lack of Concept Information. When applying the genetic operators to a draft, some information about a concept may disappear from the final text. To avoid that, this function looks for all the attributes of each concept in the ontology, and checks if they are mentioned in the text. If not, the draft is penalized.

For the evaluation of each of the texts that form the population, we use formula (1), where *error* is the accumulated error of a specific text taking into account the fitness functions mentioned above.

$$\text{fitness} = 1 / (\text{error} + k). \quad \textbf{(1)}$$

In this way, the fitness would be greater if the error is smaller. The constant k is used to avoid divisions by zero. In our experiments it was set with the value 1, so the maximum possible fitness was 1.

The results

The following text is an example of system output for a very simple input concerning the initial description of the characters that take part in a tale:

```
  A princess lived in a castle. She was the daugther of parents. They
were  stern.  The  blonde  princess  loved  a  brave  handsome  knight.  The
castle had towers. The parents lived in the castle. It was strong.
```

The beginnings of stories are rich in descriptions of characters and locations and are therefore good case studies for the problems being discussed. The input to the system provides conceptual descriptions of characters and locations in terms of attributes and relations between them. The evolutionary algorithm operates over the output of previous stages that have carried out other NLG tasks such as content determination – selecting the particular concepts that are relevant – and discourse planning - organising them in an orderly fashion. These tasks are currently carried out in a traditional manner and simply provide the data for the evolutionary stages. A traditional surface realization stage transforms the output of the evolutionary stages into readable text.

4 Conclusions and future work

With respect to both of the tasks addressed, the output texts respect the strong constraints required for the text to be acceptable, while at the same time showing reasonable variation between the different options much as a human-generated text would. Due to the main difficulties in the evaluation of the population at the end of each generation, we are currently considering the possibility of using neural networks to reproduce the human evaluation of the quality of the resulting texts. We are also working on extending the system to allow the use of proper nouns to describe some concepts, as an additional option to pronouns and descriptive references, including the revision of the genetic operators and the introduction of new evaluation functions to estimate the correct application of proper nouns. In view of these results, in future work we want to apply EA techniques to other tasks of NLG, such as content determination and discourse planning. The particular advantages of evolutionary techniques, combined stage by stage in this manner, may be an extremely powerful method for solving natural language generation problems while also profiting from classic NLG techniques.

References

1. Holland, J. H., *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, Massachusetts, Second Edition, 1992.
2. Reiter, E. "Has a consensus NL generation architecture appeared, and is it psychologically plausible?", in: McDonald, D. and Meteer, M. (eds.) Proccedings of INLGW '94, pages 163-170, Kennebunkport, Maine, 1994.
3. Reape, M, Mellish, C., "Just what is aggregation anyway?", in: Proceedings of the 7th European Workshop on Natural Language Generation, Toulouse, France, 1999.

4. Reiter, E. Dale, R. “A fast algorithm for the generation of referring expressions”, in: Proceedings of the 14th conference on Computational linguistics - Volume 1, Nantes, France, 1992