An Evolutionary Approach 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

Abstract

The work presented here is intended as an evolutionary task-specific module for referring expression generation and aggregation to be enclosed in a generic flexible architecture. Appearances of concepts are considered as genes, each one encoding the type of reference used. Three genetic operators are used: classic crossover and mutation, plus a specific operator dealing with aggregation. Fitness functions are defined to achieve elementary coherence and stylistic validity. Experiments are described and discussed.

1 Introduction

In this paper we present a first approach to the idea of using Natural Language Generation (NLG) and Evolutionary Algorithms (EAs) together.

To test the feasibility of our idea, we decided to select only some particular features of the text on which to put it to the test. Given the complexity of all the changes that are possible to a text, at the levels of syntax, semantics, discourse structure and pragmatics, it seemed impractical to tackle them all at once. For the purpose of illustration, we decided that the problems of the referring expressions and the aggregation were the most suitable to be solved using EAs. Referring Expression Generation involves deciding how each element ocurring in the input is described in the output text. Aggregation 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.*". It is necessary to find the balance between the use of compound or single sentences, or in the case of the modifiers of a concept between the description of the attributes of the concept using only a phrase or various.

We analysed the features of a human generated text from the point of view of the referring expressions, and we found five different features of simple texts that might be susceptible of easy treatment by means of evolutionary techniques. They are described below.

Correct Referent.
When writing a text, we cannot use a pronoun for something that we have not mentioned before, or readers would get confused. An example could be:

> *She lived in a castle. A princess was the daughter of parents.*

In addition, if the full noun reference and the pronoun are far, the reader can also get confused and be unable to link the two occurrences of the same concept, as we can see in the following text:

> *A princess lived in a castle. She was the daughter of parents. She loved a knight. She was pretty. She was blonde. It had towers. It was strong. They lived in it.*

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. Sentences such as the one below are incorrect:

> *The blonde princess was blonde.*

Reference Repetition.
Using always the same reference together with the same set of 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 princess"*, as in this example:

> *A pretty princess lived in a castle. The pretty princess was the daughter of parents. The pretty princess loved a knight. The pretty princess was blonde.*

To avoid that, repetitive use of references is penalized.

Coherence.
If we use different subsets of 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:

A princess lived in a castle. The pretty princess was the daughter of parents. The blonde princess loved a knight.

Overlooked Information.
When processing the conceptual representation of a given input, some information about a concept may disappear from the final text. This should be avoided.

This paper describe an evolutionary solution that guarantees the satisfaction of these restrictions in the conceptual rendition of a given input by means of shallow techniques that rely on very little knowledge about the domain and no reasoning or common sense capabilities.

2 Natural Language Generation Tasks and Evolutionary Algorithms

This section outlines the elementary requirements of the two generation tasks addressed in this paper, and sketches the basic principles of the evolutionary techniques that are used.

2.1 Referring Expression Generation and Aggregation

The correct use of referring expressions to compete with human generated texts involves a certain difficulty. Possible simple algorithms for deciding when to use a pronoun and when to use the full noun 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.

As shown in [Reiter and Dale, 1992], a referring expression must communicate enough information to be able to uniquely identify the intended referent in the current discourse context, but avoiding the presence of redundant or otherwise unnecessary modifiers. Therefore, it is essential to choose a reference which matches these constraints. Taking into account these features, Reiter and Dale proposed an algorithm to generate definite noun phrases to identify objects in the current focus of attention of the reader or the hearer. However, Krahmer and Theune [Krahmer and Theune, 2000] argue that due to the original motivation of the work of Reiter and Dale of making distinguishing descriptions, various other aspects of the generation of definites remained somewhat underdeveloped. In particular they focus on the role of context-sensitivity for referring expression generation.

Kibble and Power [Kibble and Power, 2000] propose a system which uses Centering Theory [Walker *et al.*, 1998] for planning of coherent texts and choice of referring expressions. They argue that text and sentence planning need to be driven in part by the goal of maintaining referential continuity: obtaining a favourable ordering of clauses, and of arguments within clauses, is likely to increase opportunities for non-ambiguous pronoun use.

Aggregation can be seen as the NLG task that involves deciding how compact the presentation of information should be in a given text, although there is no exact definition in the literature about what aggregation is [Reape and Mellish, 1999]. It operates at several linguistic levels, and due to that Reape and Mellish make a classification of the different types of aggregation: conceptual, discourse, semantic, syntactic, lexical and referential. However, the line between them is very narrow, and in some cases a specific example could be classified as different types of aggregation.

2.2 Evolutionary Algorithms

We propose the use of evolutionary algorithms (EAs) [Holland, 1992] to deal with the referring expression generation and aggregation tasks. Evolutionary algorithms are an extended set of problem resolution techniques inspired by evolutionary phenomena and natural evolution. They work on a population of individuals (representations of possible solutions for the problem we are solving) that evolve according to selection rules and genetic operators like crossover and mutation. The fitness function is a metric which allows the evaluation of each of the possible solutions, in such way that the average adaptation of the population would increase in each generation. Repeating this process hundreds or thousands of times it is possible to find very good solutions for the problem.

Evolutionary algorithms combine random search, because the genetic operators are applied randomly, with oriented search, given by the fitness values. These algorithms find generally good solutions, but not always the best ones. However, this is enough for simple applications. In the case under consideration, the main advantage we can find in evolutionary algorithms is that they do not need specific rules to build a solution, only measurements of its goodness.

Evolutionary techniques have been shown in the past to be particularly well suited for the generation of verse. The work of Manurung [Manurung, 2003] and Levy [Levy, 2001] proposed different computational models of the composition of verse based on evolutionary approaches. In both cases, the main difficulty lay in the choice of a fitness function to guide the process. Although Levy only addressed a simple model concerned with syllabic information, his overall description of the architecture in terms of a population of poem drafts that evolve, with priority given to those drafts that are evaluated more highly, is an important insight. Levy uses a neural network, trained with examples of valid verse, to evaluate these drafts. The work of Manurung addresses the complete task, and it presents a set of evaluators that grade the candidates solutions according to particular heuristics.

Evolutionary algorithms have been also used in text planning. In [Duboue and McKeown, 2002] the authors present a technique to learn a tree-like structure for a content planner from an aligned corpus of semantic inputs and corresponding, human produced, outputs. They apply a stochastic search mechanism with a two-level fitness function to create the structure of the planner. Genetic algorithms are also used in [Mellish *et al.*, 1998] where the authors state the problem of given a set of facts to convey and a set of rhetorical relations that can be used to link them together, how one can arrange this material so as to yield the best possible text.

An important conclusion to draw from these efforts is the suitability of evolutionary techniques for natural language generation tasks in which the form plays a significant role, to the extent of sometimes interfering with the intended content, such as is the case for lyrics generation.

3 An Evolutionary Submodule for a Simple Generator

The work presented here is intended to be a module for the tasks of referring expressions generation and aggregation enclosed in the architecture of cFROGS [García *et al.*, 2004]. cFROGS is a framework-like library of architectural classes intended to facilitate the development of NLG applications. cFROGS identifies three basic design decisions: what set of modules to use, how control should flow between them, and what data structures are used to communicate between the modules.

We have tested the implementation of the module in an existing application: ProtoPropp [Gervás *et al.*, 2004]. This is a system for automatic story generation. The natural language generator module of ProtoPropp – implemented as a pipeline architecture of cFROGS modules – perform 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. In the previous prototype of ProtoPropp the referring expression to use for a concrete concept was determined using a very simple heuristic: the first time that the concept appears in the paragraph, the generator uses its full noun, in all other cases it uses a pronoun. When using a full noun reference, it is indefinite for the first appearance of the concept in the text and definite for the rest.

The input of the evolutionary algorithm is a basic discourse structure where each phrase is a message about a relation between two concepts or a description of some attribute of an element. Additionally, this submodule has access to a knowledge base of conceptual information about the discourse elements that appear in the input (characters, locations, attributes, relations).

In this simple evolutionary algorithm, the appearances of the concepts are considered as the genes. The initial population is generated randomly, using for each concept its full noun or its pronoun. When using the full noun, a selection of the attributes the concept has in the knowledge base is chosen. These attributes will appear just before the noun 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.

3.1 Data Representation and Genes

Within the context of the larger cFROGS architecture, data are represented as complex data structures with generic interfaces to ensure easy connectivity between different modules [García *et al.*, 2004]. These data follow ideas from the RAGS [Cahill *et al.*, 2001] generic architecture. However, the notation described here corresponds to a representation internal to the module intended to facilitate the operation of the evolutionary techniques.

Characters, *locations* and *attributes* are represented as simple facts containing an unique identifier (to distinguish each specific character and location from the others) and their names. The identifier in attributes corresponds to the concept that holds the attribute, and the name corresponds to the attribute itself. The current prototype operates over simple linguistic constructs: the *description of a concept* using an attribute, or a *relation* between two concepts. *Pronominal reference* is indicated by changing the name of the concept for 'pron', and *definite and indefinite reference* is indicated by adding a fact 'ref' indicating if the reference is definite or indefinite. Finally, the concepts may go along with some *attributes preceding the name of the concept*, as in "the pretty blonde princess". This list of attributes is represented between -> and <-.

A sample part of a draft for the evolutionary algorithm would be the following:

```
[character(ch26,princess),
    ref(ind),
    ->attribute(ch26,pretty)<-,
    relation(ch26,l14,live),
    location(l14,castle),
    ref(ind)]
[character(ch26,pron),
    relation(ch26,ch25,love),
    character(ch25,knight),
    ref(ind)]
[character(ch26,princess),
    ref(def),
    isa(),
    attribute(ch26,blonde)]
```

In this example, the set of genes would be this:

```
  Genes:
 0: character(ch26,princess),
    ref(ind),
    ->attribute(ch26,pretty)<-
 1: location(l14,castle),
    ref(ind)
 2: character(ch26,pron)
 3: character(ch25,knight),
    ref(ind)
 4: character(ch26,princess),
    ref(def)
```

3.2 The Genetic Operators

Three genetic operators are used: crossover, mutation and aggregation.

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 - as in *"she lived in a castle"* -, it will change into the corresponding full noun, always associated with a subset of its possible attributes - for example *"the princess lived in a castle"* or *"the pretty princess lived in a castle"* -. In case the

Correct Referent	$error_1 = \sum$ pronominal references to a concept not referred in full in the two previous genes
Redundant Attributes	$error_2 = \sum$ "<adj> X is <adj>" sentences
Reference Repetition	$error_3 = \sum$ repeated use of same set of attributes – $att(gen_i)$ – to refer to the concept in gen_i
Coherence	$error_4 = \sum_{i=1}^{N}(att(gen_i) - I)$ with I the set of attributes used before for the concept in gen_i
Overlooked Information	$error_5 = \sum$ subset of attributes of concept i in the ontology not mentioned in the text

Table 1: Definition of fitness functions

gene was a full noun - as in *"the pretty princess"* -, there are two options: to change it into a pronoun - in this case *"she"* -, or to change the subset of attributes that appear with it - for example *"the princess"* or *"the pretty blonde princess"* -. 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 disaggregates 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 are mentioned in the appearance of a concept or in the correspondent "X is Y" sentences, but not in both. As the aggregation operator is used randomly, the desired result is obtained only in some cases.

3.3 The Fitness Function

The key to the evolutionary 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.

We have defined five different fitness functions as shown in Table 1. This definitions are the results of the analysis of the features of human-generated text.

For the evaluation of each of the drafts that form the population, we use the following formula:

$$fitness = 1/(\sum_i error_i + k)$$

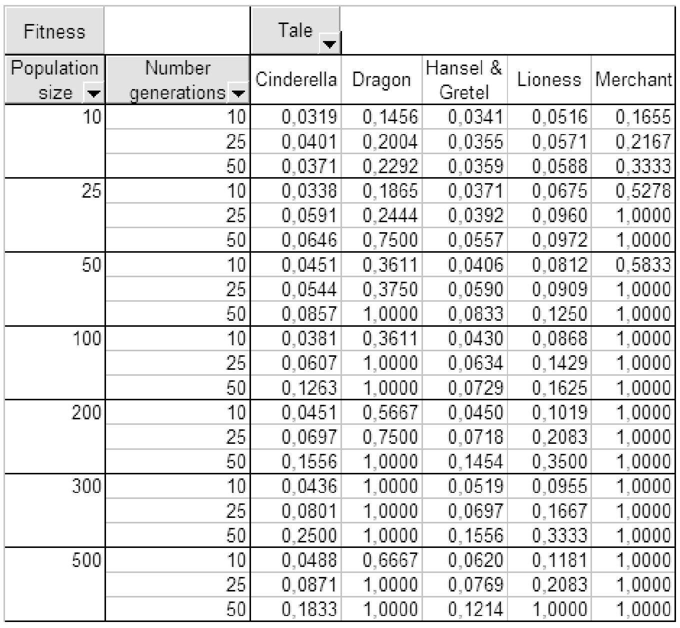
In this way, the fitness would be greater when the error is smaller. The constant k is used to avoid divisions by zero. In our experiments it was set with the value 1, so the maximum possible fitness was 1.

4 Experiments and Results

To test the feasibility of the idea of using together NLG and EAs, we have formalized five different fairy tales, mainly differentiated by their lengths in number of genes, that is, in appearances of concepts. We must take into account that the number of genes shown below are not completely exact, because the aggregation operator can erase or add new sentences to the tale. These are the tales formalized and used to do the experiments:

- Cinderella: 102 genes
- Hansel and Gretel: 90 genes
- The Lioness: 50 genes
- The Dragon: 32 genes
- The Merchant: 31 genes

For each of these tales we have made several experiments using different population sizes (10, 25, 50, 100, 200, 300, 500) and number of generations (10, 25, 50). The three genetic operators mentioned before (crossover, mutation and aggregation) are applied, and the five fitness functions used for the evaluation of the tales.



Fitness		Tale				
Population size	Number generations	Cinderella	Dragon	Hansel & Gretel	Lioness	Merchant
10	10	0,0319	0,1456	0,0341	0,0516	0,1655
	25	0,0401	0,2004	0,0355	0,0571	0,2167
	50	0,0371	0,2292	0,0359	0,0588	0,3333
25	10	0,0338	0,1865	0,0371	0,0675	0,5278
	25	0,0591	0,2444	0,0392	0,0960	1,0000
	50	0,0646	0,7500	0,0557	0,0972	1,0000
50	10	0,0451	0,3611	0,0406	0,0812	0,5833
	25	0,0544	0,3750	0,0590	0,0909	1,0000
	50	0,0857	1,0000	0,0833	0,1250	1,0000
100	10	0,0381	0,3611	0,0430	0,0868	1,0000
	25	0,0607	1,0000	0,0634	0,1429	1,0000
	50	0,1263	1,0000	0,0729	0,1625	1,0000
200	10	0,0451	0,5667	0,0450	0,1019	1,0000
	25	0,0697	0,7500	0,0718	0,2083	1,0000
	50	0,1556	1,0000	0,1454	0,3500	1,0000
300	10	0,0436	1,0000	0,0519	0,0955	1,0000
	25	0,0801	1,0000	0,0697	0,1667	1,0000
	50	0,2500	1,0000	0,1556	0,3333	1,0000
500	10	0,0488	0,6667	0,0620	0,1181	1,0000
	25	0,0871	1,0000	0,0769	0,2083	1,0000
	50	0,1833	1,0000	0,1214	1,0000	1,0000

Table 2: Table of numerical results

In Table 2 we can see the numerical results of the experiments. For each combination of population size and number

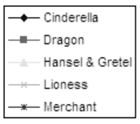


Figure 1: Legend for the tales

of generations results shown have been averaged over a number of runs.

We can analyse these results taking into account the three different number of generations used. The legend for the following graphics is shown in Figure 1.

4.1 10 Generations

As we can see in Figure 2, only 10 generations are not enough for the bigger tales. However, in the case of the smaller ones, the fitness values increase with the size of the population, and at certain point they achieve the maximum value of 1.

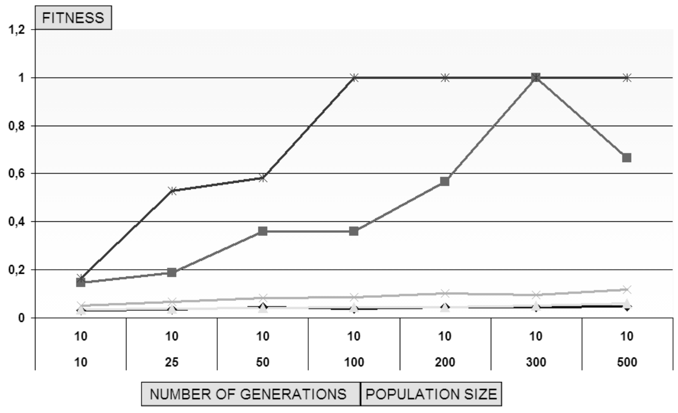


Figure 2: Fitness values of the tales with 10 generations

4.2 25 Generations

In Figure 3 the fitness values for the bigger tales are higher than in the case of 10 generations, but still not good enough. For the smaller tales we achieve the maximum fitness value of 1 quicker than with only 10 generations.

4.3 50 Generations

We can see in Figure 4 the best values achieved in the experiments. For the smaller tales, we get the maximum fitness value of 1 very quickly. In the case of the bigger ones, the fitness values are higher than in the previous experiments, but not very good yet, except in the case of “The Lioness”, where the maximum value of 1 is achieved with 50 generations and 500 individuals in the population.

5 Discussion

To start with, EAs seem to be a good approach to solve the tasks addressed, and in all the experiments the results obtained are better than the ones achieved using previous heuristics. An example of generated text with the initial simple heuristic is:

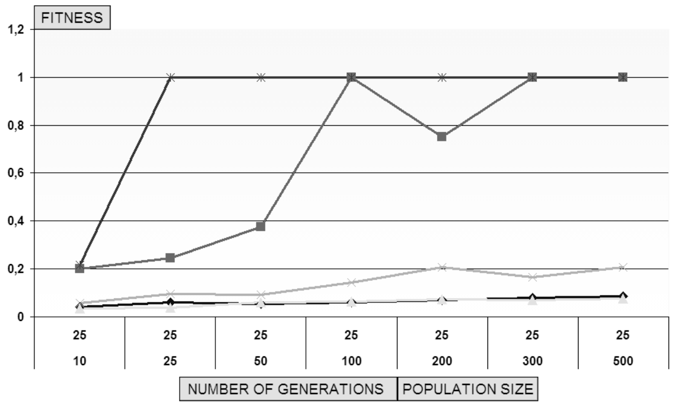


Figure 3: Fitness values of the tales with 25 generations

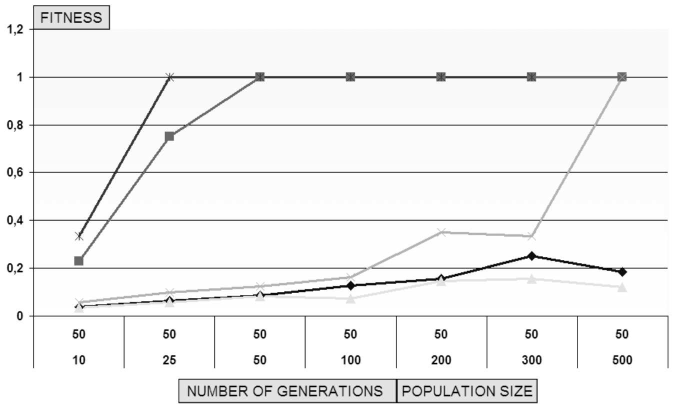


Figure 4: Fitness values of the tales with 50 generations

> *A princess lived in a castle. She loved a knight. She was pretty. She was blonde. It had towers. It was strong.*

Using the evolutionary module the same piece of tale is generated as follows:

> *A pretty princess lived in a strong castle. She was blonde. The princess loved a brave knight. The castle had towers.*

The second example shows that the texts generated by the evolutionary module are richer from the point of view of adjectives and structure.

Note that depending on the number of genes you need a certain number of individuals and generations to achieve a good fitness value. For example, “The Lioness”, with 50 genes, gets the maximum fitness with 50 generations and 500 individuals, as long as “Hansel and Gretel” and “Cinderella” would need more generations and individuals to get the maximum fitness.

Another important point is that in a specific tale, with a specific number of genes, you can achieve the same results increasing the number of generations or the size of the population. For instance, “The Merchant”, with 31 genes, gets the maximum fitness with both 25 or 50 generations with small populations or 10 generations with populations of more than 100 individuals.

Finally, it is important to note that our approach presents some differences respect to the one of Reiter and Dale [Reiter and Dale, 1992]. As we have already mentioned, we are working in the field of the fairy tales, with the specific requirements of story generation. An important point is that these are not informative texts, and therefore we can relax some constraints taken into account in other works in the area of referring expressions.

6 Conclusions and future work

With respect to both of the tasks addressed, the output texts respect the specific 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. We are 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.

It would be also interesting to compare our solution with different approaches found in the literature, as for example [Reiter and Dale, 1992] or [Krahmer and Theune, 2000] for the referring expression generation, and the one of Dalianis and Hovy [Dalianis and Hovy, 1996] for the aggregation.

Finally, an evaluation as the one proposed in [Callaway and Lester, 2001] would be useful to estimate the goodness of the generated texts. The authors describe the evaluation of STORYBOOK, a narrative prose generation system that produces original fairy tales in the Little Red Riding Hood domain. They pretend to evaluate multiple versions of a single story assuring that the content is identical across them. Five versions of two separate stories are produced, a pool of twenty students in English compare them, and at last they are analyzed with an ANOVA test.

References

[Cahill *et al.*, 2001] L. Cahill, R. Evans, C. Mellish, D. Paiva, M. Reape, and D. Scott. The RAGS reference manual. Technical Report ITRI-01-07, Information Technology Research Institute, University of Brighton, 2001.

[Callaway and Lester, 2001] C. Callaway and J. Lester. Evaluating the effects of natural language generation techniques on reader satisfaction. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, Edinburgh, UK, 2001.

[Dalianis and Hovy, 1996] H. Dalianis and E. Hovy. Aggregation in natural language generation. In G. Ardoni and M. Zock, editors, *Trends in Natural Language Generation: an Artificial Intelligence Perspective,EWNLG'93*, pages 88–105. Springer Verlag, 1996.

[Duboue and McKeown, 2002] P.A. Duboue and K.R. McKeown. Content planner construction via evolutionary algorithms and a corpus-based fitness function. In *Proceedings of the Second International Natural Language Generation Conference (INLG 2002)*, Ramapo Mountains, NY, 2002.

[García *et al.*, 2004] C. García, R. Hervás, and P. Gervás. Una arquitectura software para el desarrollo de aplicaciones de generación de lenguaje natural. *Sociedad Española para el Procesamiento del Lenguaje Natural, Procesamiento de Lenguaje Natural*, 33:111–118, 2004.

[Gervás *et al.*, 2004] P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás. Story plot generation based on CBR. In Anne Macintosh, Richard Ellis, and Tony Allen, editors, *12th Conference on Applications and Innovations in Intelligent Systems*, Cambridge, UK, 2004. Springer, WICS series.

[Holland, 1992] J.H. Holland. *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, Second Edition, 1992.

[Kibble and Power, 2000] R. Kibble and R. Power. An integrated framework for text planning and pronominalization. In *Proc. of the International Conference on Natural Language Generation (INLG), Israel*, 2000.

[Krahmer and Theune, 2000] E. Krahmer and M. Theune. Efficient context-sensitive generation of referring expressions, 2000.

[Levy, 2001] R. P. Levy. A computational model of poetic creativity with neural network as measure of adaptive fitness. In *Proccedings of the ICCBR-01 Workshop on Creative Systems*, 2001.

[Manurung, 2003] H.M. Manurung. *An evolutionary algorithm approach to poetry generation*. PhD thesis, School of Informatics, University of Edinburgh, 2003.

[Mellish *et al.*, 1998] C. Mellish, A. Knott, J. Oberlander, and M. O'Donnell. Experiments using stochastic search for text planning. In Eduard Hovy, editor, *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 98–107. Association for Computational Linguistics, New Brunswick, New Jersey, 1998.

[Reape and Mellish, 1999] M. Reape and C. Mellish. Just what is aggregation anyway? In *Proceedings of the 7th European Workshop on Natural Language Generation*, Toulouse, France, 1999.

[Reiter and Dale, 1992] E. Reiter and R. Dale. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th conference on Computational linguistics*, Nantes, France, 1992.

[Walker *et al.*, 1998] M.A. Walker, A.K. Joshi, and E.F. Prince. *Centering Theory in Discourse*. Clarendon Press, Oxford, 1998.