An Implementation of a Flexible Author-Reviewer Model of Generation using Genetic Algorithms*

Ruli Manurung[a], Graeme Ritchie[b], and Henry Thompson[c]

[a]Faculty of Computer Science, University of Indonesia, Depok 16424, Indonesia
maruli@cs.ui.ac.id

[b]Dept. of Computing Science, University of Aberdeen, King's College, Aberdeen AB24 3UE, UK
g.ritchie@abdn.ac.uk

[c]HCRC, University of Edinburgh, Informatics Forum, 10 Crichton St., Edinburgh EH8 9AB, UK
ht@inf.ed.ac.uk

Abstract. This paper proposes performing natural language generation using genetic algorithms to address two issues: the difficulty in controlling global textual features which arise from a large number of interdependent local decisions, and the difficulty in applying conventional NLG wisdom in domains where the communicative goal lacks sufficient detail. It presents details of an implemented system that embodies the aforementioned proposal, and discusses the results of an empirical study conducted using the system.

Keywords: natural language generation, genetic algorithms, poetry, creative language.

1. Background

This paper proposes an approach to natural language generation (NLG) using the genetic algorithm (GA), a widely used stochastic search method.

In this section we discuss two well known issues in NLG, and in Section 2 we discuss why and how genetic algorithms can address these issues. In Sections 3 to 5 we present an implementation of an NLG system that embodies the proposed approach. Our own interest is in developing a generator that conveys a given semantics as a text that simultaneously exhibits a certain metre, i.e. regular patterns in the rhythm of the text. Consequently, some of the design decisions, particularly concerning the evaluation functions, are domain-specific. However, we believe the architecture as a whole is of general-purpose interest. The paper concludes with some examples and discussion in Section 6.

1.1.Achieving fidelity and fluency

Oberlander and Brew (2000) argue that NLG systems must achieve fidelity and fluency goals, where fidelity is the faithful representation of the relevant knowledge contained within the communicative goal, and fluency is the ability to do it in a natural-sounding way such that it engenders a positive evaluation of the system by the user. In practice, applied NLG systems can often sidestep the fluency goal given a very restricted domain of output, with a limited style that may be just enough to serve the purpose of the application.

Unfortunately, fluency may be controlled by global textual features which arise from a large number of local decisions, few of which are based on stylistic considerations. This does not suit

* The work reported in this paper was carried out while all authors were at the University of Edinburgh.

the prevalent paradigm of NLG as a top-down, goal-driven process, decomposed into the stages of content determination, text planning, and surface realisation, typically implemented within a *pipeline architecture* (Reiter, 1994) (cf. the "generation gap" problem in Meteer (1991)).

Oberlander and Brew propose an architecture which consists of two collaborating modules: an author and a reviewer. The author faces the task of generating a text that conveys the correct propositional content, i.e. achieving fidelity, whereas the reviewer must ensure that the author's output satisfies whatever macroscopic properties have been imposed on it, i.e. achieving fluency. Recent corpus-based NLG systems (Langkilde and Knight, 1998) essentially embody this architecture: a symbolic generator acts as the author, and a language model acts as the reviewer.

1.2.Vague communicative goals

Most NLG systems make two basic assumptions: that text generation is communicative goal-driven, and that these goals are sufficient to dictate a top-down approach for the planning of the text's structure and decomposition of goals. However, Mellish et al. (1998b) claim there is a class of NLG problems for which these basic assumptions do not apply. In the case of the ILEX system, this is due to two factors. Firstly, as ILEX produces explanation labels of jewelry items on display, there is often no clear plan or goal to be conveyed beyond "*say something coherent and interesting about this artifact within the space available*". Secondly, ILEX can not plan far in advance, as it has to generate text in real-time based on the user's choices.

The alternative approach they adopt is *opportunistic planning*, whose key elements are interleaving of planning and execution, flexible choice of tasks from an agenda, expanding "sketchy plans" as needed, taking into account the current state of the world, and recognition of opportunities through detection of reference features.

2. Using genetic algorithms to do NLG

In addressing the above issues, we advocate treating the NLG process as a constraint satisfaction problem, where a solution is a text that satisfies multiple interdependent constraints relating to various levels of linguistic representation, e.g. semantic, syntactic, pragmatic, stylistic. Finding such a solution requires searching a space that is undoubtedly immense. Our proposed solution is to employ the *genetic algorithm* (GA), a widely-used heuristic search strategy that relies on random traversal of a search space with a bias towards more promising solutions. Specifically, it evolves a population of individuals over time, through an iterative process of evaluation, selection, and evolution. Upon termination, the fittest individual is hoped to be an optimal, or near-optimal, solution (Bäck et al., 1997).

Using GAs to do NLG has been done before, e.g. Mellish et al. (1998a). However, these previous attempts employed GAs as optimisation functions for specific subtasks of NLG. We believe that handling the entire NLG process through GAs opens up the potential for various flexible approaches, which we discuss in Section 2.1. In particular, employing GAs allows a measure of opportunistic planning (Section 1.2), where the evolutionary cycle enables fitness functions to recognize opportunities and provide feedback to the executor, i.e. genetic operators.

Finally, we note that using genetic algorithms for NLG also reflects a *discriminative* model of generation, where domain knowledge is stated declaratively, i.e. what a good text should look like instead of how to write one (cf. the corpus-based systems in Section 1.1).

2.1.Representing linguistic constraints and the encoding of domain knowledge

When using GAs for NLG, a solution is a text that must achieve fluency and fidelity goals. These goals can be expanded as a set of constraints to be satisfied by a text, e.g. it must be grammatical, it must convey some given meaning, it must be readable, etc.

There are two ways constraints can be implemented: ensuring that all possibly evolvable solutions never violate the constraint, or imposing penalties on individuals that violate a constraint. There is a trade-off: the former approach is obviously ideal, but its intractability is often the very reason GAs are employed in the first place. On the other hand, imposing

excessively heavy penalties often leads to premature convergence on the first found well-formed solution, whereas if the penalties are too light, the GA may continue to evolve ill-formed solutions that score better than well-formed ones.

Within this framework, there is large scope for flexibility in terms of where domain-specific knowledge is encoded to help satisfy these constraints. For NLG, it seems reasonable to assume that candidate solutions must at least be grammatically well-formed.

Oberlander & Brew's author-reviewer model specifies that the author focuses on achieving fidelity, whereas the reviewer focuses on maximizing fluency goals. In GAs, this suggests devising genetic operators that explicitly work towards realising some input semantics, and fitness functions that measure fluency factors such as readability, length, coherence, etc.

However, other setups are possible. For instance, one could envisage an author (genetic operator) concentrating on fluency whilst a separate reviewer (fitness function) assessed the output for fidelity, or a pair of reviewers assessing a document of grammatical nonsense[1], each concentrating on fluency and fidelity respectively. Such approaches may seem unnecessarily awkward for conventional NLG tasks, but may provide a more suitable platform for NLG systems without well-defined communicative goals (see Section1.2).

Note that the various components, i.e. the ensemble of authors and reviewers, can more or less be defined independently of each other, modulo the need for a common representation of a candidate text. This addresses the "engineering argument", one of the main arguments supporting the pipeline architecture as opposed to an integrated architecture (Reiter, 1994), i.e. it enables a modular decomposition of an NLG system, thus resulting in a more manageable implementation.

3. Linguistic representation

Our system represents candidate texts as lexicalized tree adjoining grammar (LTAG) derivation trees, augmented with the use of feature structures (Vijay-Shanker and Joshi, 1988). A derivation tree can be seen as the basic formal object that is constructed during the course of sentence generation from a semantic representation (Joshi, 1987). However, derivation trees are also the ideal data structure within our system for another reason, i.e. the non-monotonic structure building nature of GAs. Since the genetic operators may involve randomly altering subtrees through subtree deletion and swapping, we must somehow undo the unification of certain feature structures. Using the derivation tree as our primary data structure, we are able to store all local feature structures in their respective elementary trees (cf. Kilger (1992)). When required, e.g. to evaluate certain properties of the resulting text, the derived tree is rebuilt. Redundant computation is minimized by reusing a cached derived tree if it has not been modified between iterations.

Within evolutionary theory, the LTAG derivation tree can be viewed as the *genotypic* representation of candidate solutions, from which we can compute the *phenotypic* information of semantic (Section 5.2) and prosodic (Section 5.1) features via the derived tree.

We adopt a simple 'flat' semantic representation that is often used in NLG (Koller and Striegnitz, 2002). A *semantic expression* is a set of first order logic literals, which is logically interpreted as a conjunction of all its members. The arguments of these literals represent domain concepts such as objects and events, while the functors state relations between these concepts. See Section 5.2 for some examples.

The semantic form of a tree is the union of the semantic expressions of its constituent elementary trees, with binding of variables during substitution and adjunction to control predicate-argument structure; cf. Stone et al. (2001).

Finally, since our system requires information on prosody, each word is associated with its phonetic spelling, taken from the CMU pronouncing dictionary[2].

[1] As produced by a statistical language model, or by some combination of monkeys and typewriters.
[2] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

4. Genetic operators for NLG

Genetic operators are functions that are stochastically applied to candidate solutions to explore alternative solutions. In essence, they define the search space. When designing operators for our system, the following desiderata were considered:

1. **Grammaticality:** The operators should ensure that syntactic well-formedness of the candidate texts be preserved. This suggests that genetic operators be based on the derivational rules of the underlying grammatical formalism (LTAG).
2. **Non-monotonicity:** As operators are stochastically applied, it is highly improbable that optimal texts can be constructed using monotonic structure-building operators alone. Rather, they are typically built through the trial and error design mechanism that evolution affords. Therefore, one or more non-monotonic operators must facilitate this, such as deletion, replacement, and swapping of substructures.
3. **Incrementality:** Constructing texts in an incremental fashion enables the generation process to benefit from the guiding hand of evolution. The appropriate granularity of operator incrementality is an open question. Furthermore, incomplete derivations may conflict with requirements of grammaticality.

Within our framework, several sets of operators were implemented.

4.1.Baseline operators

Existing work in *genetic programming* defines genetic operators on tree data structures, such as *grow*, which randomly selects a leaf from a tree and replaces it with a randomly generated new subtree, *shrink*, which does the opposite, and *switch*, which randomly selects two nodes and swaps their position.

Within the context of NLG, although it seems obvious to perform such structural manipulations on phrase structure trees, we argue that they are most appropriately applied to the LTAG derivation tree instead. This maintains the syntactic principle of well-formed LTAG structures being constructed through valid compositions of elementary trees using the operations of substitution and adjunction. The BLINDADD operator adds a node in the derivation tree; variants exist for both substitution and adjunction. The BLINDDELETE operator removes a node in the derivation tree, along with the subtree that it dominates. Finally, the BLINDSWAP operator swaps the positions of two subtrees, either belonging to the same derivation tree, resulting in *mutation*, or to another derivation tree, resulting in *crossover*.

These baseline operators, while grammatically sound (in particular, all involved feature structures must license the operation), are oblivious to fidelity and fluency goals. Consequently, they indiscriminately add, delete, and swap both good and bad content, delegating judgments of quality to the fitness evaluation functions.

4.2.Semantically motivated operators

Given the task of generating the sentence "*John loves Mary*", it seems absurd that an NLG system would attempt to add content concerning, say, Greek archaeological artefacts or medical conditions, yet this is an entirely possible scenario given the baseline operators above.

Accordingly, we implemented a set of operators that deliberately attempts to bring the semantics of a candidate text closer to that of some pre-defined input semantics. This is precisely the task of surface realisation in NLG, and our approach is reminiscent of Nicolov (1998) and Stone et al. (2001).

Our "semantically smart" operator, SMARTADD, explicitly tries to realise some portion of the input semantics *S*, specifically that which has not yet been realised, while simultaneously maintaining syntactic well-formedness. Nicolov calls this gradual process the *consumption* of semantics. Conversely, SMARTDELETE will only consider removing elementary trees whose lexical semantics are extraneous with respect to *S*, and SMARTSWAP will only consider swapping subtrees that preserve the predicate argument structure represented by *S*.

Such operators require a way of reasoning about the relationship between the input semantics and the semantics conveyed by the candidate text, i.e. which portion of the input semantics has been realised, and which portion of the candidate text semantics indeed realises the input, or is extraneous. This is achieved using the semantic mapping algorithm discussed in Section 5.2.

5. Fitness functions for NLG

In GAs, the fitness function is where the bulk of domain-specific knowledge and heuristics is typically encoded. Specifically for NLG, the fitness function serves as a metric, or more precisely a set of metrics, that measure whether a candidate text achieves the goals of fidelity and fluency.

For our implemented system, we measured fidelity in terms of how well a candidate text realised a given propositional input, and fluency in terms of how closely the rhythmic stress patterns of a text matched a given poetic metre.

5.1.Metre similarity

Our system is tasked with conveying a given semantics as a text that exhibits a given metre. For example, Fig. 1 shows the metre of Hillaire Belloc's "*The Lion*", with stressed syllables in bold type, unstressed syllables in normal type, syllables extraneous to the underlying metre in italics, and • indicating a 'missing' syllable.

The	**Li**on, the	**Li**on, he	**dwells** in the	**waste**,
He	**has** a big	**head** and a	**ve**ry small	**waist**;
But his	**shoul**ders are	**stark**, and his	**jaws** they are	**grim**,
And a	**good** little	**child** • will	**not** play with	**him**.

Figure 1: Metre pattern of Belloc's "*The Lion*"

Our system represents metre patterns as a list of *stress syllables* notated as follows: **w** ('*weak*') is an unstressed syllable, **s** ('*strong*') is a stressed syllable, **x** ('*wildcard*') is any syllable, and **b** indicates a linebreak. Fig. 2 shows example notations for (a) a limerick, and (b) "*The Lion*" (formatted into lines for readability purposes).

```
[w,s,w,w,s,w,w,s,b,
 w,s,w,w,s,w,w,s,b,
    w,s,w,w,s,b,
    w,s,w,w,s,b,
 w,s,w,w,s,w,w,s,b]
```

(a)

```
[w,s,w,w,s,w,w,s,w,w,s,b,
 w,s,w,w,s,w,w,s,w,w,s,b,
 w,s,w,w,s,w,w,s,w,w,s,b,
 w,s,w,w,s,w,w,s,w,w,s,b]
```

(b)

Figure 2: Encoding for (a) a limerick and (b) "*The Lion*"

Our metre evaluation function measures the degree of similarity between a given metre pattern and the metre exhibited by a candidate text. To compute this, we use the well-known *minimum edit distance*, in which the distance between two strings is the minimal sum of costs of operations (symbol insertion, deletion, and substitution) that transform one string into another. We have devised a suitable cost function that reflects our intuitions of metre. Since the edit distance only accounts for context-free operations, we implemented a metre compensation function to account for the fact that context can affect lexical stress, particularly in poetry.

Our metre evaluation function, $\mathcal{F}_{metre}$, takes the value computed by the minimum edit distance algorithm, adjusts it using our context-sensitive compensation scheme, and normalizes it to the interval [0,1]. Table 1 shows $\mathcal{F}_{metre}$ values for various candidate texts, against the target form in Fig. 2(b). The first is Belloc's actual poem, which itself contains some metrical imperfections; the second is a limerick by Edward Lear; the third is an extract from an academic text, containing roughly the correct number of syllables; the last is chosen for its inappropriateness. The $\mathcal{F}_{metre}$ scores do not conflict with our intuitions of poetic metre.

Table 1: Metre fitness for various texts

Candidate text	$\mathcal{F}_{metre}$
The Lion, the Lion, he dwells in the waste. He has a big head and a very small waist. But his shoulders are stark, and his jaws they are grim, and a good little child will not play with him.	0.787
There was an old man with a beard, who said, "it is just as i feared! two owls and a hen, four larks and a wren, have all built their nests in my beard!"	0.686
Poetry is a unique artifact of the human language faculty, with its defining feature being a strong unity between content and form.	0.539
John loves Mary.	0.264

5.2.Semantic similarity

Following Love (2000), we propose two factors that must be considered: *structural similarity* and *conceptual similarity*. Structural similarity measures the degree of isomorphism between two semantic expressions. Conceptual similarity is a measure of relatedness between two concepts (logical literals). We simply use the following: two concepts are the same if and only if they share the same literal functor. However, one could envisage a refined approach using an underlying ontology such as WordNet, or using statistical models of lexical semantics, e.g. LSA.

Computing a structural similarity mapping between two expressions is an instance of the NP-complete maximal common subgraph problem. However, we have implemented a greedy algorithm that serves our purposes and runs in $O(N^2)$, based on Gentner's structure mapping theory (Falkenhainer et al., 1989). It takes two sets of logical literals, S_{target} and $S_{candidate}$, and attempts to 'align' the literals. We then apply a function $\mathcal{F}_{sem}$, normalised to [0,1], to compute a score based on various aspects of the alignment; this is based on Love's computational model of similarity Love (2000).

Table 2 shows an example of computing semantic similarity for a selection of candidate texts against a target semantics that represents the second line of Belloc's "*The Lion*", i.e. "[The lion] has a big head and a very small waist". The target semantic expression is as follows:

$$S_{target} = \{lion(\_,L), own(\_,L,H), head(\_,H), big(\_,H), own(\_,L,W), waist(\_,W), small(S,W), very(\_,S)\}$$

The first two texts convey a subset of the target; the third text conveys an altogether different fact about the lion; the fourth text is purposely inappropriate; and the last text, conveys the semantics of the first text in its object to the verb '*love*'. As with our metre similarity function, we believe that the $\mathcal{F}_{sem}$ scores roughly approximate human intuitions.

Table 2: Semantic fitness for various texts

Candidate text	**Candidate semantics**	$\mathcal{F}_{sem}$
The lion has a big head	*{lion(_,L), own(_,L,H), head(_,H), big(_,H)}*	0.525
The lion has a head and a waist	*{lion(_,L), own(_,L,H), head(_,H), own(_,L,W), waist(_,W)}*	0.598
The lion dwells in the waste	*{lion(_,L), dwell(D,L), inside(_,D,W), waste(_,W)}*	0.078
John loves Mary	*{john(_,J), love(_,J,M), mary(_,M)}*	0.0451
John and Mary love the lion's big head	*{john(_,J), love(_,J,H), mary(_,M), love(_,M,H), lion(_,L), own(_,L,H), head(_,H), big(_,H)}*	0.389

6. Testing and discussion

Throughout our testing, we employed *proportionate selection*, which assigns a distribution that accords parents a probability to reproduce that is proportional to its fitness. Individuals are sampled from this distribution using *stochastic universal sampling*, which minimises chance fluctuations in sampling. To reduce the chances of premature convergence or stagnation, we

used an *elitist* population of 20% of the entire population (the latter being 40). See Bäck et al. (1997) for a review of these issues. Each test was run five times, and each run lasted for 500 iterations. The three mutation operators used, along with their probabilities, were substitution (0.5), adjunction (0.3), and deletion (0.2). For crossover, the subtree swapping operator was used. The probabilities of applying genetic operators were $p_{mutation} = 0.6$, $p_{crossover} = 0.4$, for both the "blind" and "smart" variants. A small handcrafted grammar and lexicon was used, with 33 elementary trees and 134 lexical items, 28 of which were closed class words. Most of the content words were taken from Belloc's "*The Bad Child's Book of Beasts*".

6.1.Fluency and fidelity generation

In this test, we measured the ability of our system to generate texts that simultaneously achieve fidelity and fluency goals. We took a very simple approach to combining the metre similarity and semantic similarity functions – the arithmetic mean of their scores, i.e.

$$\mathcal{F}_{fitness} = \frac{\mathcal{F}_{metre} + \mathcal{F}_{sem}}{2}$$

The target metre was that of a limerick, as in Fig. 2(a). The target semantics was a representation of the first two lines of "*The Lion*" (Fig. 1), with a slight alteration where the original opening noun phrase "*The lion, the lion*" was replaced with "*The african lion*". The target semantic expression is as follows:

$$\mathcal{S}_{target} = \{lion(\_, l), african(\_, l), dwell(d, l), inside(\_, d, was), waste(\_, was), own(\_, l, h), head(\_, h), big(\_, h), own(\_, l, wat), waist(\_, wat), small(s, wat), very(\_, s)\}$$

Two variants of the test were conducted: one with the baseline 'blind' operators and one with the semantically-aware 'smart' operators.

Table 3 shows the highest-scoring candidate from the blind operator test. The text is metrically perfect. However, the unmapped $\mathcal{S}_{target}$ literals show that the text fails to convey three concepts, i.e. that the lion is *african*, that its head is *big*, and that the waist is *very* small.

Table 3: Solution for blind operators tests

Fitness score: 0.81
Text: A **li**on, it **dwells** in a **waste**. A **li**on, it **dwells** in a **waste**. A **waste** will be **rare**. Its **head** will be **rare**. Its **waist**, that is **small**, will be **rare**.
Unmapped $\mathcal{S}_{target}$: *{african(_1, l), big(_6,h), very(_9, s)}*
Unmapped $\mathcal{S}_{candidate}$: *{rare(_33,_34), will(_35,_36), waste(_37,_34), dwell(_38,_39), lion(_40,_39), inside(_41,_38,_42), waste(_43,_42), rare(_44,_45), will(_49,_50), rare(_51,_52), will(_55,_56)}*

Table 4 shows the highest-scoring candidate from the smart operators test. Although the fitness score is very similar to the one in Table 3, the characteristics of the text are markedly different. The smart operators, which increase bias towards semantics, have a detrimental effect on the metre. Unlike the metrically perfect limerick in Table 3, this text requires several edit operations: 2 insertions and 2 deletions (even Belloc's original poem contains similar rhythmic imperfections – see Section 5.1). However, it does a better job of conveying $\mathcal{S}_{target}$, only failing to convey the fact that the waist is *very* small, whilst also conveying fewer extraneous semantics (most of which are repetitions of correct semantics).

Table 4: Solution for smart operators test

Fitness score: 0.83
Text: A **ve**ry • **af**rican **li***on*, *who* is **af**rican, **dwells** in a **waste**. Its **head**, that is **big**, is **ve**ry • **big**. A **waist**, *that* is its **waist**, it is **small**.
Unmapped S_{target}: *{very(_9, s)}*
Unmapped $S_{candidate}$: *{very(_137,_135), big(_141,_136), waist(_145,_143), very(_153,_152), african(_154,_140)}*

6.2.Line by line generation

In this test, we had our system generate each line of a limerick individually. The purpose is to see whether the system can perform better given a simpler task. We also based this test on a different limerick to show the flexibility of the system.The new input is shown in Table 5. Note that the target metres represent an *ideal* limerick. The "gold standard" limerick itself is metrically imperfect.

Table 5: Buller's original limerick as individual lines.

Line 1: *There was a young lady called Bright.* $S = \{lady(\_, l), young(\_, l), name(\_, l, b), bright(\_, b)\}$ **[w,s,w,w,s,w,w,s,b]**
Line 2: *She could travel much faster than light.* $S = \{travel(t, l), faster(f, t, li), light(\_, li), much(\_, f), can(\_, t)\}$ **[w,s,w,w,s,w,w,s,b]**
Line 3: *She set out one day in a relative way.* $S = \{leave(le, l), relative(\_, le), oneday(\_, le)\}$ **[w,s,w,w,s,w,s,w,w,s,b]**
Line 4: *She returned on the previous night.* $S = \{return(r, l), on(\_, r, n), night(\_, n), previous(\_, n)\}$ **[w,s,w,w,s,w,w,s,b]**

Table 6 shows the best solution obtained by trying to generate the whole limerick at once, as in previous tests, whereas Table 7 collects the results of generating each individual line. In the latter case, the resulting limerick is metrically much better than the former, as there are only two edits compared to five. Both of these generated texts are metrically superior to the original.

Table 6: Solution for entire limerick.

Fitness score: 0.69
Text: A **la**dy could **<u>be</u>** on an **eve***ning*, that **<u>could</u>** be pre**ce**ding, one **day**. A *young* **la**dy <u>called</u> **Bright**, who **set** out one **day**, **tra**velled much **fast**er than **light**.
Unmapped S_{target}: *{can(_6, t), relative(_7, le), return(r, l)}*
Unmapped $S_{candidate}$: *{oneday(_199,_210), lady(_201,_197),can(_221,_207),can(_228,_213)}*

Table 7: Collected solution for individual lines.

Fitness score: Line 1: 0.82, Line 2: 0.66, Line 3: 0.78, Line 4: 0.86
Text: A **la**dy called **Bright** could be **young**. She **tra**velled. The **light** could be **light**. She **set** out one **day**. She **set** out one **day**. She **is** on a **pre**vious **night**.
Unmapped S_{target}: *{travel(t, l), faster(f, t, li), much(_1, f), relative(_0, le), {return(r, l)}*
Unmapped $S_{candidate}$: *{can(_57,_81), {light(_527,_524), travel(_603,_536), {leave(_359,_323), oneday(_333,_359)}*

However, the system fails to satisfy the semantics, and in fact does worse in the latter case, as there are five unmapped S_{target} literals as opposed to three. This suggests that, particularly given the smaller task of individual line generation, our evaluation function is not guiding the GA to optimize semantics as well as it is for metre. We attempt to address this in the final test.

6.3.Evaluation weighting

The results in the preceding test suggest that the evaluation function is biased towards metre optimisation. In our final test, we simply modified the linear combination by doubling the weight of semantic fitness as follows:

$$F_{fitness} = \frac{F_{metre} + 2 * F_{sem}}{3}$$

Table 8 collects the results of generating each individual line using the modified evaluation function. We believe this text is definitely an improvement over the ones in Tables 6 and 7, and most closely resembles the original limerick in Table 5. Note that semantically it only lacks 2 target literals and only has 1 extraneously conveyed literal. Metrically, it requires 9 edits. Subjectively, however, we believe it still scans reasonably well as a limerick.

This suggests that semantic fitness should carry more weight than metre fitness, perhaps reflecting the intuition that fidelity is more of a 'harder' constraint than fluency is.

Table 8: Collected solution, modified fitness

Fitness score: Line 1: 0.78, Line 2: 0.79, Line 3: 0.95, Line 4: 0.76
Text: There **is** a young **la**dy called **Bright**. She *will* **tra**vel much **fast**er than **light**. She **set** out one **day** * **rel**ative**ly**. She **is** on *a* pre**ce**ding * **night**.
Unmapped S_{target}: *{can(_2, t), return(r, l)}*
Unmapped $S_{candidate}$: *{will(_409,_415)}*

7. A (Speculative) Summary

We have proposed a flexible author-reviewer model for performing NLG that is based on GAs to address the two issues presented in Section 1.

We then presented details of an implemented instance of this model, which specifically aims to convey a given semantics as a text that satisfies a given metre pattern. Through a series of small tests, we showed that it has the potential to satisfy the interdependent goals of fidelity and fluency (compare in particular, the output in Table 8 with the gold standard in Table 5).

As implemented, our system does not really address the difficulty of generation when the communicative goal is vague: the semantic similarity function (Section 5.2) still requires an existing propositional input. However, one can envisage other measures of fidelity that account for notions of coherence, interestingness, consistency. As for fluency, one could replace our very specific metre similarity function with a declarative model of, for instance, readability, document length, personality, and language complexity.

References

Back, T., D. Fogel, and Z. Michalewicz, editors. Handbook of Evolutionary Computation. Oxford University Press and Institute of Physics Publishing, 1997.

Falkenhainer, B., K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. Artificial Intelligence, 41:1–63, 1989.

Joshi, A. K. The relevance of tree adjoining grammars to generation. In G. Kempen, editor, Natural Language Generation: New Results in Artificial Intellligence, pages 233–252. Martinus Nijhoff Press, Dordrecht, The Netherlands, 1987.

Kilger, A. Realization of tree adjoining grammars with unification. Technical Report TM-92-08, DFKI, Saarbr¨ucken, Germany, 1992.

Koller, A. and K. Striegnitz. Generation as dependency parsing. In Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics, Philadelphia, USA, July 2002.

Langkilde, I. and K. Knight. Generation that exploits corpus-based statistical knowledge. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, pages 704–710, Montreal, Canada, August 1998.

Love, B. C. A computational level theory of similarity. In Proceedings of the 22nd Annual Meeting of the Cognitive Science Society, pages 316–321, Philadelphia, USA, August 2000.

Mellish, C., A. Knott, J. Oberlander, and M. O'Donnell. Experiments using stochastic search for text planning. In Proceedings of the Ninth International Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada, 1998a.

Mellish, C., M. O'Donnell, J. Oberlander, and A. Knott. An architecture for opportunistic text generation. In Proceedings of the Ninth International Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada, 1998b.

Meteer, M. Bridging the generation gap between text planning and linguistic realisation. Computational Intelligence, 7(4):296–304, 1991.

Nicolov, N. Approximate Text Generation from Non-Hierarchical Representations in a Declarative Framework. PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK, 1998.

Oberlander, J. and C. Brew. Stochastic text generation. Philosophical Transactions of the Royal Society of London, Series A, 358:1373–1385, 2000.

Reiter, E. Has a consensus on NL generation appeared? and is it psycholinguistically plausible? In Proceedings of the Seventh International Natural Language Generation Workshop, pages 163– 170, Kennebunkport, USA, April 1994. Springer-Verlag.

Stone, M., C. Doran, B. Webber, T. Bleam, and M. Palmer. Microplanning with communicative intentions: The SPUD system. Technical Report TR-65, Rutgers University Center for Cognitive Science, New Jersey, USA, 2001.

Vijay-Shanker, K. and A. K. Joshi. Feature structure based tree adjoining grammars. In Proceedings of 12th International Conference of Computational Linguistics, pages 714–720, Budapest, Hungary, August 1988.Blackburn, P. and J. Bos. 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Publications.