Multiobjective Optimization for Meaningful Metrical Poetry

Fahrurrozi Rahman and Ruli Manurung
Faculty of Computer Science
Universitas Indonesia
Depok 16424, Indonesia
rahman10@ui.ac.id, maruli@cs.ui.ac.id

Abstract

This paper reports our experiments to properly handle the multiobjective optimization nature of poetry generation — as defined in Manurung (2003) — as a stochastic search that seeks to produce a text that simultaneously satisfies the properties of grammaticality, meaningfulness, and poeticness. In particular, we employ the SPEA2 Algorithm (Zitzler, Laumanns, and Thiele 2001). Various results show that it consistently outperforms our previous system in its ability to generate a meaningful metrical text according to given semantic and metre specifications, and in some cases is able to generate the intended text, whereas our previous system fails to do so. However, it is still unable to reach the goal of generating an entire poem. We conclude with suggestions for further work to address this shortcoming.

Introduction

Manurung (2003) presents a model of poetry generation as a stochastic search process that seeks to produce a text that simultaneously satisfies various properties, as well as McGONAGALL, an implementation of the model. It lays out the representational framework, and defines evaluation functions that independently assess the ability of a text to (i) convey a given semantics whilst (ii) conforming to a given rhythmic pattern. However, it fails to account for the multiobjective optimization nature of this model of poetry generation: form and function in poetry are highly interdependent, and as such, it is incorrect to optimize for both by optimizing a simple linear combination of the separate evaluation functions.

In this paper we report our efforts to properly handle the multiobjective optimization nature of poetry as stochastic search. In particular, the Strength Pareto Evolutionary Algorithm (Zitzler, Laumanns, and Thiele 2001), one of the top-performing multiobjective optimization algorithms, is used. We start by introducing the model of poetry as a text that embodies *meaningfulness*, *grammaticality*, and *poeticness*, and a model of its generation as a multiobjective optimization stochastic search process. We then describe McGONAGALL, an implemented system that adopts this model and uses genetic algorithms (Mitchell 1996) to generate texts that are syntactically well-formed, meet certain prespecified patterns of metre, and broadly convey some given meaning. Finally, we present results of some experiments we conducted given various inputs.

Poetry writing as multiobjective optimization

Despite the vast number of different definitions of poetry, one can argue that a common characteristic is the presence of a strong interaction, or *unity*, between the form and content of a poem. The diction and grammatical construction of a poem affects the message that it conveys to the reader over and above its obvious denotative meaning (Levin 1962). As such, Manurung (2003) argues that poetry generation is much harder than conventional NLG (natural language generation), which typically operates on the assumption that a text serves as a medium for conveying its semantic content.

To account for all this, Manurung proposes a general model of a poem as a natural language artifact that simultaneously satisfies the constraints of grammaticality, meaningfulness, and poeticness. A *grammatical* poem must be syntactically well-formed. This might seem obvious for natural language texts, but it must be stated explicitly in the context of poetry. Syntactic well-formedness in poetry may well be different from that of ordinary texts (cf. figurative language), but it is still governed by rules. A *meaningful* poem must intentionally convey some conceptual message that is meaningful under some interpretation. Finally, *poeticness* states that a poem must exhibit features that distinguishes it from non-poetic text. We follow Manurung (2003) in concentrating on the concretely observable aspect of rhythmic patterns, or *metre*.

This characterization suggests a 'classical' account of poetry where adherence to regular patterns in form is essential. This avoids some of the complications of imagery or interpretation that are central to assessing more free forms of verse, and ensures that many of the important aspects of a text can be defined formally. Figure 1 shows a prototypical example of this genre — and one that we will revisit throughout this paper, i.e. a limerick by Arthur H.R. Buller, first published in Punch magazine on the 19th December 1923 (Knowles 2009).

Evolving poetry

Poetry generation can be viewed as a specialized instance of *natural language generation*, or NLG, i.e. the development of computer systems that can produce understandable

There	**was** a young	**la**dy called	**Bright**
who could	**tra**vel much	**fas**ter than	**light**.
She	**set** out one	**day**	
in a	**re**lative	**way**	
and re	**turned** on the	**pre** vious	**night**.

Figure 1: Arthur Buller's 'relativity' limerick

texts in a human language, starting from some nonlinguistic representation of a *communicative goal* as input (Reiter and Dale 2000). A considerable amount of research has been done in NLG on the so-called "generation gap" problem, where interdependent decisions must be made across various levels of linguistic representation. This problem is exacerbated by the unity of poetry. For example, with regards to metre, every single linguistic decision potentially determines the success of a poem in fitting a regular defined pattern.

Poetry generation can be viewed as a state space search problem, where a state in the search space is a possible text with all its underlying representation, from semantics all the way down to phonetics. A goal state satisfies the three constraints of meaningfulness, grammaticality, and poeticness. Such a search space is undoubtedly immense, even — given the recursive structures in natural language — infinite. Stochastic search is a heuristic search strategy that relies on the random traversal of a search space with a bias towards more promising solutions. It has become increasingly popular for solving computationally hard combinatorial problems such as constraint satisfaction, and has been shown to outperform deterministic approaches in a number of domains.

The *genetic algorithm*, or GA, is a particularly well-known instance of such a strategy, which is essentially an iteration of two phases, *evaluation* and *evolution*, applied to an ordered set, called the *population*, of candidate solutions (Mitchell 1996).

Multiobjective optimization

There are two ways that constraints can be implemented in a GA: either invalid solutions are totally excluded from the search space, or they are admitted to the space but with a bias against them. In MCGONAGALL, grammaticality is treated as a prerequisite for meaningfulness and poeticness; that is, grammaticality is implemented as an absolute constraint on the items admitted to the search space, through the design of representation and genetic operators. The remaining constraints of meaningfulness and poeticnes will be implemented as preferences via the evaluation functions of the genetic algorithm.

Given the two constraints to be optimized, poetry generation is thus an instance of multiobjective optimization. The Strength Pareto Evolutionary Algorithm (SPEA) 2 is an algorithm which handles multiobjective optimization. It is based on its predecessor, SPEA, which still has some weaknesses such as dealing with fitness assignment, density estimation and archive truncation (Zitzler, Laumanns, and Thiele 2001).

SPEA originally uses a population and archive set that is maintained per iteration. Initially the archive set is empty and shall contain any nondominated individuals from the previous population and archive in the next iteration. Furthermore, if the size of the archive exceeds a predefined limit, the archive will be truncated without destroying its characteristics. SPEA2 overcomes the problem of fitness assignment and density estimation in SPEA by taking account of both dominating and dominated values for each individual.

An individual $a \in X$ is said to dominate another individual $b \in X$ (also written as $a \succ b$) if and only if

$$\begin{array}{l} \forall i \in \{1, \ldots, n\} : f_i(a) \geq f_i(b) \quad \text{and} \\ \exists j \in \{1, \ldots, n\} : f_j(a) > f_j(b) \end{array}$$

for each objective function $(f_1, \ldots, f_n)$.

From the definition above, the strength (dominating) value, $S(i)$, for individuals in the population P_t and the archive A_t can be defined by:

$$S(i) =\mid \{j \mid j \in P_t + A_t \wedge i \succ j \mid$$

where $\mid . \mid$ indicates the cardinality of a set and the symbol + denotes the multiset union.

Based on the strength value, the raw (dominated) value of an individual, $R(i)$, is calculated as:

$$R(i) = \sum_{j \in P_t + A_t, j \succ i} S(j)$$

The smaller the raw value the better it is, i.e. $R(i) = 0$ means the individual is a nondominated one.

The density is obtained using an adaptation of the *k*-th nearest neighbor method, where the density estimation is the inverse of the distance to the *k*-th nearest neighbor :

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

Here σ_i^k denotes the *k*-th nearest neighbor for individual *i* to all individuals *j* in the population and archive set. As commonly used, *k* equals to the square root of the population and archive size, or $k = \sqrt{\mid P \mid + \mid A \mid}$.

Adding the density value to the raw value of an individual yields its fitness value :

$$F(i) = D(i) + R(i)$$

Unlike SPEA, SPEA2 preserves the number of individuals in the archive to be constant. If the number of nondominated individuals is less than the archive size, the best dominated individuals in the previous archive and population are copied to the archive until the archive is full. If the opposite happens, each individual in the archive is sorted based on its *k*-th nearest neighbor value and the one with the minimum distance will repeatedly be removed from the archive. This method prevents boundary solutions being removed. This archive set will be the solution set when the stopping criterion is satisfied.

MCGONAGALL: an evolutionary poet

In this section we will briefly describe how MCGONAGALL ensures grammaticality through its linguistic representation and genetic operators, and how it optimizes poeticness and meaningfulness.

Firstly, linguistic structures are represented using *lexicalized tree-adjoining grammar*, or LTAG. These grammars are based on the composition of *elementary trees*, of which there are two types, i.e. initial trees and auxiliary trees. These trees represent minimal linguistic structures in the sense that they account for all and only the arguments of the head of the syntactic constituent they represent, e.g. a sentential structure would contain a verb and all its complements.

In LTAG, the *derivation tree* is a data structure that records the composition of elementary trees using substitution and adjunction. The *derived tree*, on the other hand, is the phrase structure tree created by performing all the operations specified within the deriviation tree; in a sense, the derived tree is the result of generation, whereas the derivation tree is a trace of how it was built up. The derivation tree can therefore be seen as the basic formal object that is constructed during sentence generation from a semantic representation (Joshi 1987), and is the appropriate data structure on which to perform nonmonotonic operations in a stochastic generation framework. Essentially, the LTAG derivation tree forms the *genotypic* representation of a candidate solution, from which one can compute the *phenotypic* information of semantic and prosodic features via the derived tree.

A simple 'flat' semantic representation (Hobbs 1985) is used. 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 concepts in the domain such as objects and events, while the functors state relations between these concepts. For example, the representation of the semantics of the sentence *"John loves Mary"* is $\{john(j), mary(m), love(l, j, m)\}$, where l is the event of j, who has the property of 'being John', loving m, who has the property of 'being Mary'. The semantic expression of a tree is the union of the semantic expressions of its constituent elementary trees, with appropriate binding of variables during substitution and adjunction to control predicate-argument structure.

Finally, each word is associated with its phonetic spelling, taken from the CMU pronouncing dictionary. Vowels are marked for lexical stress, with `0` for no stress, `1` for primary stress, and `2` for secondary stress. For example, the spelling of *'dictionary'* is `[D,IH1,K,SH,AH0,N,EH2,R,IY0]`. For monosyllables, closed class words (e.g. *the*) receive no stress, and open class words (e.g. *cat*) primary stress.

Genetic mutation operators that randomly *add* or *delete* subtrees of a derivation tree have been introduced to move through the search space. In addition, a subtree *swapping* operator that randomly swaps two derivation tree subtrees is also implemented. When these subtrees belong to the same derivation tree, it behaves as a mutator, otherwise, it is used as a crossover operator.

```
[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]
```

Figure 2: Target form for a limerick

Evaluating poeticness

In this paper, poeticness is taken to be the well-defined and objectively observable *metre*, i.e. regular patterns in the rhythm of the lines. Figure 1 shows the metre of Buller's limerick, with stressed syllables in bold type, unstressed syllables in normal type, and syllables extraneous to the underlying metre in italics. The first, second, and fifth lines have the same number of stressed syllables, with a regular pattern of 'beats' at intervals of two unstressed syllables, and likewise for the third and fourth lines.

The system tries to maximize the similarity between the *target form*, a specification of the required metrical constraints, and the *candidate form*, the stress pattern of a candidate solution. The target form is encoded as a list of *target syllables*, notated as follows: `w` (*'weak'*) is an unstressed syllable, `s` (*'strong'*) is a stressed syllable, and `b` indicates a linebreak. Figure 2 shows the example target form for a limerick(formatted into lines for readability purposes).

The candidate form, a representation of the metre exhibited by a candidate solution, is encoded as a list of *candidate syllables* obtained from a derived tree by concatenating the phonetic spellings at its leaf nodes.

To compute the similarity between the target and candidate forms, we use the 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. The minimum edit distance can be efficiently computed in a way that produces a pairwise syllable alignment between candidate and target, thus indicating the operations that yield the minimum cost. The operation costs for substitution, insertion, and deletion of syllables have been assigned to reflect our intuitions in perceiving poetic metre. Our candidate forms indicate lexical stress patterns as if the words were pronounced in isolation. Within poetic text, context can affect stress. To compensate for this, the system iterates over the minimum edit distance alignment, detecting certain patterns and adjusting the similarity value. Two types of patterns are implemented: two consecutive deletions, or two consecutive insertions, of non-linebreaks, increases the cost by 1; the destressing of a stressed candidate syllable adjacent to a stressed target syllable, or the stressing of an unstressed candidate syllable adjacent to an unstressed target syllable, decreases the cost by 1.

The metre evaluation function, $\mathcal{F}_{metre}$, takes the value computed by the minimum edit distance algorithm, adjusts it using the context-sensitive compensation scheme, and normalizes it to the interval [0,1].

Line 1: *There was a young lady called Bright.* `relativity1`: $\{lady(l), young(l), name(l,b), bright(b)\}$
Line 2: *She could travel much faster than light.* `relativity2`: $\{travel(t,l), faster(f,t,li), light(li), much(f), can(t)\}$
Line 3: *She set out one day in a relative way.* `relativity3`: $\{leave(le,l), relative(le), oneday(le)\}$
Line 4: *She returned on the previous night.* `relativity4`: $\{return(r,l), on(r,n), night(n), previous(n)\}$

Table 1: Modified limerick consisting of four sentences

Evaluating meaningfulness

An approach to meaningfulness is essentially similar to the above approach to poeticness: try to maximize the similarity between the *target semantics*, a specification of the meaning an optimal solution should convey, and a *candidate semantics*, the meaning conveyed by a candidate solution. This requires a method for computing the similarity between two semantic expressions.

Love (2000) proposes 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).

Computing a structural similarity mapping between two expressions is an instance of the NP-hard graph isomorphism problem. However, Manurung implemented a greedy algorithm that runs in $O(N^3)$, based on Gentner's structure mapping theory (Falkenhainer, Forbus, and Gentner 1989). It takes two sets of logical literals, S_{target} and $S_{candidate}$, and attempts to 'align' the literals. In doing this, it creates various variable bindings and also two sets of *unmatched* literals that are left over (from S_{target} and $S_{candidate}$).

A function $\mathcal{F}_{sem}$, normalised to [0,1], is then applied to compute a score based on various aspects of the best match that has been achieved; this is based on Love's computational model of similarity (Love 2000).

Experiments in multiobjective optimization

For our experiments, we used Buller's 'relativity' limerick shown in Figure 1 as the target to be generated. The semantics and metre of this limerick are encoded and provided to the system as the target semantics and target metre.

Furthermore, aside from the task of generating the entire limerick, we experiment with the generation of this limerick on a line-by-line basis. Thus, we have modified the text slightly so that it consists of four complete sentences which our system can generate individually. These four sentences are shown in Table 1, along with the respective semantic targets, `relativity1` to `relativity4`.

This modified limerick preserves the metre and syllable count of the original. However, the third and fourth lines from the original limerick have now been merged into one line. The form targets for lines 1, 2, and 4 of the modified limerick are the same, as follows:

`limerickline1: [w,s,w,w,s,w,w,s,b]`

The form target for line 3 is as follows:

`limerickline2: [w,s,w,w,s,w,s,w,w,s,b]`

To summarize, we will be using `relativity1`, `relativity2`, and `relativity4` as S_{target} along with `limerickline1` as F_{target} to generate lines 1, 2, and 4, `relativity3` as S_{target} and `limerickline2` as F_{target} to generate line 3, and finally, the union of `relativity1` to `relativity4` as S_{target} along with the limerick pattern in 2 as F_{target} to generate the entire modified limerick.

Experimental setup

For each target, we ran the genetic algorithm using both the SPEA2 multiobjective optimization algorithm and the algorithm used in (Manurung 2003), i.e. where the fitness score being optimized is simply a linear combination of the semantic and metre evaluation functions, i.e. $(\mathcal{F}_{sem} + \mathcal{F}_{metre})/2$. Moreover, the GA employs *proportionate selection*, which assigns a distribution that accords parents a probability to reproduce that is proportional to its fitness (Bäck, Fogel, and Michalewicz 1997), where individuals are sampled from this distribution using *stochastic universal sampling*, which minimises chance fluctuations in sampling, using an *elitist* population of 20% of the entire population. All other parameters were also adapted from Manurung (2003), i.e. the population size was set to 40, each test was run ten times, each run lasted for 500 iterations, the mutation operators used, along with their probabilities, were creation (0.5), adjunction (0.3), and deletion (0.2). For crossover, the subtree swapping operator was used. The linguistic resources used, i.e. grammar and lexicon, are unchanged from MCGONAGALL.

Results and discussion

Table 2 shows a statistical summary of the best fitness scores obtained during the various experiments. It shows the minimum, maximum, mean, and standard deviation of the best fitness scores obtained after the last iteration across the ten runs of each test. To further confirm the results obtained on Buller's limerick, we experimented with a different set of target semantics and metre, namely Hillaire Belloc's "The Lion", a poem that is used throughout Manurung (2003). We used two variations, the "half" variation which makes use of the first two lines of Belloc's poem, and the "full" variation which uses all four lines.

Figure 3 shows, over time, the maximum and average of the best fitness scores across all the GA runs for the generation of limerick line 4. Finally, Tables 3 and 4 show the actual highest-scoring solutions from the various limerick tests. It shows the fitness score, text, and semantic mapping of S_{target} to $S_{candidate}$. Note that for all these experimental results, the fitness scores reported for the SPEA2 results are in fact $(\mathcal{F}_{sem}+\mathcal{F}_{metre})/2$. This is merely for the purpose of

Test	Min	Max	Mean	Std.Dev
Line 1				
Linear combination	0.57	0.77	0.60	0.06
SPEA2	0.69	0.88	0.76	0.08
Line 2				
Linear combination	0.52	0.62	0.56	0.03
SPEA2	0.56	0.88	0.74	0.12
Line 3				
Linear combination	0.52	0.61	0.53	0.03
SPEA2	0.6	0.87	0.79	0.10
Line 4				
Linear combination	0.57	0.69	0.65	0.05
SPEA2	0.59	0.83	0.74	0.07
Entire				
Linear combination	0.42	0.61	0.51	0.05
SPEA2	0.49	0.62	0.55	0.04
Lion "half"				
Linear combination	0.45	0.70	0.57	0.07
SPEA2	0.61	0.71	0.68	0.03
Lion "full"				
Linear combination	0.47	0.53	0.51	0.02
SPEA2	0.56	0.65	0.60	0.03

Table 2: Statistical summary of test results

presenting the results in such a way that it is meaningful to compare them against the linear combination fitness. During the evolution phase of SPEA2, it uses a Pareto-based fitness score based on population domination statistics.

From all of these experimental results it can be observed that the GA consistently performs better using the SPEA2 algorithm compared to the simple linear combination used in (Manurung 2003). This can be easily seen in Figure 3, where the average and best scores using SPEA2 are always higher than the average and best scores using the linear combination. This pattern recurs in the plots for all the other targets. Note that none of the runs achieve a perfect fitness score of 1.0. This is to be expected, given that the target text

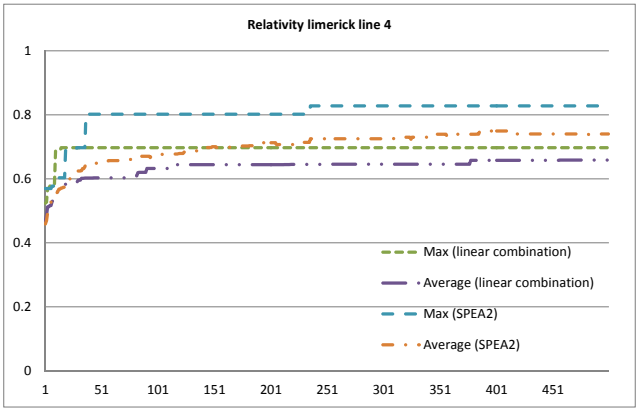

Figure 3: Fitness scores progression for limerick line 4

Line 1 (Linear combination) fitness: 0.77
Text: A **la**dy called **Bright** is on **Bright**.
Matched: $\{name(l,b), lady(l), bright(b)\}$
Umatched: $\{young(l)\}$
Line 1 (SPEA2) fitness: 0.88
Text: There **was** the young **la**dy called **Bright**.
Matched: $\{name(l,b), lady(l), young(l), bright(b)\}$
Umatched: $\{\}$
Line 2 (Linear combination) fitness: 0.62
Text: It **mel**ted her. **Light** could be **small**.
Matched: $\{can(t), light(li)\}$
Umatched: $\{travel(t,l), faster(f,t,li), much(f)\}$
Line 2 (SPEA2) fitness: 0.88
Text: *They* could **tra**vel much **fas**ter than **light**.
Matched: $\{faster(f,t,li), travel(t,l),$ $much(f), light(li), can(t)\}$
Umatched: $\{\}$
Line 3 (Linear combination) fitness: 0.61
Text: An **a**nimal **left**. An **a**nimal **left**.
Matched: $\{leave(le,l)\}$
Umatched: $\{relative(le), oneday(le)\}$
Line 3 (SPEA2) fitness: 0.87
Text: They **re**lative*ly* **left** one **day**. It sur**vived**.
Matched: $\{leave(le,l), oneday(le), relative(le)\}$
Umatched: $\{\}$
Line 4 (Linear combination) fitness: 0.69
Text: An **a**nimal **dwells** on a **night**.
Matched: $\{on(r,n), night(n)\}$
Umatched: $\{return(r,l), previous(n)\}$
Line 4 (SPEA2) fitness: 0.83
Text: They **left**. On the **night**, she re**turned**.
Matched: $\{on(r,n), return(r,l), night(n)\}$
Umatched: $\{previous(n)\}$

Table 3: Best found solution for limerick lines 1 to 4

Entire limerick (Linear combination) fitness: 0.61
Text: A **la**dy re**sides** on a **night**.
A **night** could be **re**lative. **Facts**,
that **could** be on **Bright**,
could **wan**der on **Bright**.
A **la**dy re**sides** on the **light**.
Matched: $\{on(r,n), can(t), bright(b), light(li),$ $lady(l), relative(le), night(n)\}$
Umatched: $\{young(l), name(l,b), travel(t,l),$ $faster(f,t,li), much(f), leave(le,l), oneday(le),$ $return(r,l), previous(n)\}$
Entire limerick (SPEA2) fitness: 0.62
Fitness: 0.62
Text: She **could** left much **fas**ter than **men**.
There **is** the young **la**dy, *who* with **men**,
on the **eve**ning, re**turned**,
called **pre**vious **Bright**.
She one **day** left much **fas**ter than **them**.
Matched: $\{name(l,b), on(r,n), faster(f,t,li), lady(l),$ $young(l), return(r,l), bright(b), night(n), much(f),$ $can(t), oneday(le)\}$
Umatched: $\{travel(t,l), light(li), leave(le,l),$ $relative(le), previous(n)\}$

Table 4: Best found solution for entire limerick

itself, i.e. Buller's limerick, is suboptimal from a rhythmic perspective. Note, for example, the extraneous leading syllables in lines 2, 4, and 5 of Figure 1, and the fact that open class words such as 'young' and 'called' receive no stress in line 1. All of these incur penalties when measured by $\mathcal{F}_{metre}$ against the target metre in Figure 2.

Examining the best solutions from Table 3, it can be seen that the linear combination GA satisfies the metre pattern perfectly (modulo the destressing of the open class word 'called' in line 1), at the expense of some unrealized semantics, i.e. unmatched literals in all cases. This would suggest that the linear combination approach hinders the growth of the rarer opportunities to satisfy the target semantics. On the other hand, SPEA2 is able to achieve perfect semantics in all but the last line, however does occasionally introduce extraneous syllables (lines 2 and 3). Since our system ignores the gender and number of pronouns, the fact that some lines contain *they* as opposed to *she* is not penalized.

Putting it all together, the result of generating the limerick line-by-line using the linear combination approach in Manurung (2003) is as follows:

A **la**dy called **Bright** is on **Bright**.
It **mel**ted her. **Light** could be **small**.
An **a**nimal **left**. An **a**nimal **left**.
An **a**nimal **dwells** on a **night**.

whereas using SPEA2 it is as follows:

There **was** the young **la**dy called **Bright**.
They could **tra**vel much **fas**ter than **light**.
They **re**lative*ly* **left** one **day**. It sur**vived**.
They **left**. On the **night**, she re**turned**.

Turning our attention to the generation of the entire limerick, we can see that both approaches still fail to generate a successful poem (Table 4). The same failure occurs for the generation of Belloc's "Lion" poem. This may suggest that some form of discourse modelling is imperative if the approach is expected to generate anything beyond the sentence level. The ability of McGonagall to successfully generate a text is directly determined by the ability of its evaluation functions to discriminate between "good" and "bad" solutions. Since our evaluation functions only measure coverage of propositional semantics, it has no way of discerning between coherent and incoherent texts. When attempting to generate longer multi-sentence texts, such as limericks, this approach appears rather naive. One solution is to construct an evaluation function that accounts for discourse.

Summary and future work

This paper has shown how the multiobjective optimization nature of poetry generation as a stochastic search that seeks to produce a text that simultaneously satisfies the properties of grammaticality, meaningfulness, and poeticness, needs to be handled by appropriate algorithms, such as the SPEA2 algorithm. Our results show that it consistently outperforms the previous system in its ability to generate a meaningful metrical text according to given semantic and metre specifications. It is quite successful in generating short one-line sentences.

Unfortunately, it is still unable to find a solution given the much harder task of generating an entire limerick. As discussed in the previous section, we believe this can be rectified by augmenting the evolutionary algorithm with evaluation functions that account for discourse models. One idea is to employ Rhetorical Structure Theory (Mann and Thompson 1987), which is quite often used in NLG systems, but rarely in a discriminative model. Another exciting avenue is to explore the possibilities of integrating the work of story generation systems, which explicitly aim to generate narratives, e.g. (Gervás et al. 2006).

References

Bäck, T.; Fogel, D.; and Michalewicz, Z., eds. 1997. *Handbook of Evolutionary Computation*. Oxford University Press and Institute of Physics Publishing.

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

Gervás, P.; Lönneker-Rodman, B.; Meister, J. C.; and Peinado, F. 2006. Narrative models: Narratology meets artificial intelligence. In *Proceedings of the LREC-06 workshop Toward Computational Models of Literary Analysis*.

Hobbs, J. 1985. Ontological promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, 61–69. Chicago, USA: The Association for Computational Linguistics.

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

Knowles, E., ed. 2009. *The Oxford Dictionary of Quotations*. Oxford University Press, 7 edition.

Levin, S. R. 1962. *Linguistic Structures in Poetry*. Number 23 in Janua Linguarum. 's-Gravenhage: Mouton.

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

Mann, W. C., and Thompson, S. A. 1987. Rhetorical structure theory: A framework for the analysis of texts. Technical Report RS-87-185, USC Information Science Institute, Marina Del Rey, UK.

Manurung, H. M. 2003. *An Evolutionary Algorithm Approach to Poetry Generation*. Ph.D. Dissertation, School of Informatics, University of Edinburgh.

Mitchell, M. 1996. *An Introduction to Genetic Algorithms*. Cambridge, USA: MIT Press.

Reiter, E., and Dale, R. 2000. *Building Natural Language Generation Systems*. Cambridge, UK: Cambridge University Press, first edition.

Zitzler, E.; Laumanns, M.; and Thiele, L. 2001. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich.