Full-FACE Poetry Generation

Simon Colton[1], **Jacob Goodwin**[1] **and Tony Veale**[2]
[1]Computational Creativity Group, Department of Computing, Imperial College London, UK. ccg.doc.ic.ac.uk
[2] School of Computer Science and Informatics, University College Dublin, Ireland. afflatus.ucd.ie

Abstract

We describe a corpus-based poetry generation system which uses templates to construct poems according to given constraints on rhyme, meter, stress, sentiment, word frequency and word similarity. Moreover, the software constructs a mood for the day by analysing newspaper articles; uses this to determine both an article to base a poem on and a template for the poem; creates an aesthetic based on relevance to the article, lyricism, sentiment and flamboyancy; searches for an instantiation of the template which maximises the aesthetic; and provides a commentary for the whole process to add value to the creative act. We describe the processes behind this approach, present some experimental results which helped in fine tuning, and provide some illustrative poems and commentaries. We argue that this is the first poetry system which generates examples, forms concepts, invents aesthetics and frames its work, and so can be assessed favourably with respect to the FACE model for comparing creative systems.

Introduction

Mainstream poetry is a particularly human endeavour: written by people, to be read by people, and often about people. Therefore – while there are some exceptions – audiences expect the opportunity to connect on an intellectual and/or emotional level with a person, which is often the author. Even when the connection is made with characters portrayed in the poem, the expectation is that the characters have been written from a human author's perspective. In the absence of information about an author, there is a default, often romantic, impression of a poet which can be relied upon to provide sufficient context to appreciate the humanity behind a poem. Using such an explicit, default or romantic context to enhance one's understanding of a poem is very much part of the poetry reading experience, and should not be discounted.

Automated poetry generation has been a mainstay of computational creativity research, with dozens of computational systems written to produce poetry of varying sophistication over the past fifty years. In the literature review given below, it is clear that the emphasis has been almost entirely on artefact generation, i.e., producing text to be read as poetry, rather than addressing the issues of context mentioned above. Therefore, without exception, each of these systems has to be seen as an assistant (with various levels of autonomy) for the system's user and/or programmer, because that person provides the majority of the context. This is usually achieved by supplying the background material and templates; or curating the output; or writing technical papers to describe the sophistication of the system; or writing motivational text to enhance audience understanding, etc.

While such poetry assistants are very worthwhile, we aim instead to build a fully autonomous computer poet, and for its poems to be taken seriously in full disclosure of the computational setting. The first step towards this aim is to acknowledge that the poems generated will not provide the usual opportunities to connect with a human author, as mentioned above. A second step therefore involves providing a suitable substitute for the missing aspects of humanity. To partly address this, we have built a system to construct poems via a corpus-based approach within which existing snippets of human-written text are collated, modified and employed within the stanzas of poem templates. In particular, in the *Corpus-Based Poetry* section below, we describe how a database of similes mined from the internet, along with newspaper articles can be used to generate poems.

A third step, which also addresses the missing human element to some extent, involves providing a context within which a poem can be read. Software may not be able to provide an author-centric human context, but it can provide a context which adds value to a poem via an appeal to aspects of humanity, in particular emotions. In the section below entitled *Handing over High-Level Control*, we describe how the software uses a corpus of newspaper articles to (a) determine a mood for the day in which it is writing a poem, which it uses to (b) generate an aesthetic and templates within which to generate poems, then (c) selects and modifies corpus material to instantiate the templates with, ultimately producing poems that express the aesthetic as best as possible. To communicate aspects of the context, a final step has been to enable it to provide a commentary on its work, which can be referred to by readers if required.

In the *Illustrative Results* section, we present some poems along with the commentaries generated alongside them. Given our aim for the poems to be considered in full disclosure of their computational context, along with various other arguments given in (Pease and Colton 2011b), we believe it is not appropriate to use Turing-style tests in the evaluation of this poetry generation project. Instead, we turn initially to the FACE descriptive model described in (Colton, Charnley, and Pease 2011) and (Pease and Colton 2011a), which suggests mechanisms for evaluating software in terms of the types of generative acts it performs. In the *Conclusions and Future Work* section below, we argue that we can reasonably claim that our software is the first poetry generator to achieve ground artefact generation of each of the four types prescribed in the FACE model, namely: examples, concepts, aesthetics and framing information. We believe that such full-FACE generation is the bare minimum required before we can start to properly assess computer poets in the wider context of English literature, which is a longer term aim for this project. We describe how we plan to increase the autonomy and sophistication of the software to this end.

Background

Perhaps the first computational poetry generator, the *Stochastische Texte* system (Lutz 1959), sought recognisably Modernist literary affect using a very small lexicon made from sixteen subjects and sixteen predicates from Kafka's *Das Schloß*. The software randomly fitted Kafka's words into a pre-defined grammatical template. Poems by software in this genre – where a user-selected input of texts are processed according to some stochastic algorithm and assigned to a pre-defined grammatical and/or formal template – have been published, as in (Chamberlain and Etter 1984) and (Hartman 1996), and they remain popular on the internet, as discussed in (Addad 2010). Such constrained poetry generation follows on from the OULIPO movement, who inaugurated the poetics of the mathematical sublime with *Cent mille milliards de poèmes* (Queneau 1961), an aesthetic expressed today in digital poems like *Sea and Spar Between* (Montfort and Strickland 2010).

Most of the more sophisticated poetry generation software available on the internet is designed to facilitate *digital poetry*, that is, poetry which employs the new rhetorics offered by computation. For examples, see (Montfort and Strickland 2010), (Montfort 2009) and (Roque 2011). We distinguish this from a stronger definition of *computationally creative* poetry generation, where an autonomous intelligent system creates unpredictable yet meaningful poetic artefacts. Recent work has made significant progress towards this goal; in particular, the seminal evolutionary generator McGONAGALL (Manurung 2004) has made a programmatic comeback, as described in (Rahman and Manurung 2011) and (Manurung, Ritchie, and Thompson 2012). This work is based on the maxim that "deviations from the rules and norms [of syntax and semantics] must have some purpose, and not be random" and the authors specify that *falsifiable* poetry generation software must meet the triple constraints of grammaticality, meaningfulness and poeticness. McGONAGALL, the most recent incarnation of the WASP system described in (Gervás 2010), and the system described in (Greene, Bodrumlu, and Knight 2010), all produce technically proficient poems satisfying these criteria.

There are a number of systems which use corpora of human-generated text as source material for poems. In particular, (Greene, Bodrumlu, and Knight 2010) and (Gervás 2010) rely on small corpora of already-poetic texts. The Hiveku system (www.prism.gatech.edu/˜aledoux6/hiveKu/) uses real-time data from Twitter; (Wong and Chun 2008) use data from the blogosphere and search engines; and (Elhadad et al. 2009) have used Project Gutenberg and the Google n-grams corpus. These approaches all rely on user-provided keywords to start a search for source material and seed the poetry generation process. The haikus produced by the system described in (Wong and Chun 2008) using Vector Space manipulation demonstrate basic meaningfulness, grammaticality and poeticness, but are tightly constrained by a concept lexicon of just 50 keywords distilled from the most commonly used words in the haiku genre. The Electronic Text Composition (ETC) poetry engine (Carpenter 2004) is one of a few generators to use a very large corpus of everyday language in the service of meaningful poetry generation. Its knowledge base is constituted from the 85 million parsed words of the British National Corpus, which has been turned into a lexicon of 560,000 words and 49 million tables of word associations. ETC generates its own poem templates, and its corporal magnitude encourages surprising, grammatically well-formed output. A dozen of its poems were published under a pseudonym (Carpenter 2004).

The creative use of figurative language is essential to poetry, and is a notion alluded to, but declared beyond the scope of (Manurung, Ritchie, and Thompson 2012). One example of prior research in this direction is the system of (Elhadad et al. 2009), which generates haiku, based on a database of 5,000 empirically gathered word association norms. It was reported that this cognitive-associative source principle produced better poems than a WordNet based search. Other aspects of small-scale linguistic creativity relevant to poetry generation include the production and validation of neologisms (Veale 2006), and the elaborations of the Jigsaw Bard system (Veale and Hao 2011), which works with a database of simple and ironic similes to produce novel compound similes.

Concerning aspects of computational poetry at a higher level than example generation, the WASP system can be considered as performing concept formation, as it employs a cultural meta-level generation process, whereby a text is considered and evolved by a group of distinct expert subsystems "like a cooperative society of readers/critics/editors/writers" (Gervás 2010). However, the results of the iterative evaluation are not presented with the final output, and the system does not generate the aesthetics it evaluates, which are "strongly determined by the accumulated sources used to train the content generator", in a similar way to (Greene, Bodrumlu, and Knight 2010) and (Díaz-Agudo, Gervás, and González-Calero 2002).

To the best of our knowledge, there are no poetry generation systems which produce an aesthetic framework within which to assess the poems they produce. Moreover, none of the existing systems provide any level of context for their poetry. In general, the context within which the poems can be appreciated is either deliberately obfuscated to attempt to facilitate an objective evaluation, as per Turing-style tests, or is provided by the programmer via a technical paper, foreword to an anthology, or web page. There are a myriad of websites available which generate poems in unsophisticated ways and then invite the reader to interpret them. For instance, the RoboPoem website (www.cutnmix.com/robopoem), states that: "A great deal of poetry mystifies it's readers: It may sound pretty, but it leaves you wondering 'what the hell was that supposed to mean?"' then extols the virtue of randomly generating mysterious-sounding poetry. This misses the point that poets use their intellect to write poetry which might need unpicking, in order to better convey a message, mood or style. A random sequence of words is just that, regardless of how poem-shaped it may be. The RoboPoet (the smartphone version of which enables you to "generate nonsensical random poems while waiting at the bus-stop") and similar programs only serve to highlight that people have an amazing capacity to find meaning in texts generated with no communicative purpose.

Corpus-Based Poetry Generation

As we see above, using human-produced corpora is common in computational poetry. It has the advantages of (a) helping to avoid text which is utterly un-interpretable (as most human-written text is not like this), which would likely lead to a moment where readers remember that they are not reading the output of a fully intelligent agent, and (b) having an obvious connection to humanity which can increase the semantic value of the poem text, and can be used in framing information to add value to the creative act. However – especially if corpora of existing poems are used – there is the possibility of accusations of plagiarism, and/or the damning verdict of producing pastiches inherent with this approach. Hence, we have chosen initially to work with very short phrases (similes) mined from the internet, alongside the phrases of professional writers, namely journalists writing for the British Guardian newspaper. The former fits into the long-standing tradition of using the words of *the common man* in poetry, and the latter reflects the desire to increase quality while not appropriating text intended for poems.

The simile corpus comes from the Jigsaw Bard system[1] which exploits similes as *readymades* to drive a "modest form of linguistic creativity", as described in (Veale and Hao 2011). Each simile is provided with an evidence score that indicates how many times phrases expressing the simile were seen in the Google n-gram corpus[2] from which they were mined. There are 21,984 similes in total, with 16,579 having evidence 1 and the simile "As happy as a child's life" having the most evidence (1,424,184). Each simile can be described as a tuple of $\langle object, aspect, description \rangle$, for instance $\langle child, life, happy \rangle$. Our database of Guardian newspaper articles was produced by using (i) their extensive API[3] to find URLs of articles under headings such as *World* and *UK* on certain days (ii) the Jericho package[4] to extract text from the web pages pointed to by the URLs, and (iii) the Stanford CoreNLP package[5] to extract sentences from the raw text. As of writing, the database has all 12,820 articles made available online since 1st January 2012, with the *World* section containing the most articles at 1,232.

In addition to the corpora from which we directly use text, we also employ the following linguistic resources:
[1] The CMU Pronunciation Dictionary[6] of 133,000 words.
[2] The DISCO API[7] for calculating word similarities, using a database of distributionally similar words (Kolb 2008).
[3] The Kilgariff database of 208,000 word frequencies (Kilgarriff 1997), mined from the British National Corpus[8]. This database also supplies detailed part-of-speech (POS) tagging for each word, with major and minor tags given.
[4] An implementation[9] of the Porter Stemmer algorithm (Porter 1980) for extracting the linguistic stems of words.
[5] The well known WordNet[10] lexical database.
[6] An implementation[11] of the Text Rank keyphrase extraction algorithm (Mihalcea and Tarau 2004).
[7] The Afinn[12] sentiment dictionary, containing 2,477 words tagged with an integer from -5 (negative affect) to 5 (positive affect). We expanded this to a dictionary of around 10,000 words by repeatedly adding in synonyms for each word identified by WordNet.

Poetry generation is driven by a four stage process of: *retrieval, multiplication, combination* and *instantiation*. In the first stage, similes are retrieved, according to both sentiment and evidence. That is, a range of relative evidence values can be given between 1% (very little evidence) and 100% (the most evidence) along with a sentiment range of between -5 and 5 (as per [7]). Note that the sentiment value of the $\langle object, aspect, description \rangle$ triple is calculated as the average of the three words, with a value of zero being assigned to any word not found in [7]. Constraints on word frequencies, as per [3], can also be put on the retrieval, as can constraints on the pronunciation of words in the simile, as per [1]. In addition, an article from the Guardian can be retrieved from the database (with details of how the article is chosen given later), keyphrases can be extracted using [6], and these can be further filtered to only contain relatively unusual words (as per [3]), which often contain the most pertinent information in the article.

Simile Multiplication

In the second stage, variations for each simile are produced by substituting either an object, aspect or description word, or any combination thereof. The system is given a value n for the number of variations of given simile G required, plus a substitution **scheme** specifying which parts should be substituted, and a choice of three substitution **methods** to use. Denoting G_o, G_a and G_d for the object, aspect and description parts of G, the three methods are:

(d) Using DISCO [2] to retrieve the n most similar words to each word, as determined by that system.

(s) Using the corpus of similes to retrieve the n most similar words to each word. This is calculated with reference to G and the whole corpus. For instance, suppose G_d is to be substituted. Then all the matching similes, $\{M^1, \ldots, M^k\}$, for which $M^i_o = M^i_o$ or $M^i_a = M^i_a$ are retrieved from the database. The set M^i_d of words for $i = 1, \ldots, k$ are collated, and a repetition score $r(M^i_d)$ for each one is calculated as: $r(M^i_d) = |\{j \in 1, \ldots, k : M^j_d = M^i_d\}|$. Informally, for a potential substitute, this method calculates how many similes it appears in with another word from G. The n words with the highest score are used as substitutes.

(w) Using Wordnet [5] to retrieve the n most frequent synonyms of each word, with frequency assessed by [3].

Each variation, V, of G is checked and pruned if (i) the simile exists already in the database, (ii) the major POS of either V_o, V_a or V_d differs from the corresponding part of G, or (iii) the overall sentiment of V is positive when that of G is negative (or vice-versa). To determine the yield of variations each method can produce, we ran the system to

[1] afflatus.ucd.ie/jigsaw
[2] books.google.com/ngrams/datasets
[3] www.guardian.co.uk/open-platform
[4] jericho.htmlparser.net
[5] nlp.stanford.edu/software
[6] www.speech.cs.cmu.edu/cgi-bin/cmudict
[7] www.linguatools.de/disco/disco_en.html
[8] www.natcorp.ox.ac.uk
[9] www.tartarus.org/~martin/PorterStemmer
[10] wordnet.princeton.edu
[11] lit.csci.unt.edu
[12] fnielsen.posterous.com/tag/afinn

Scheme	d	s	w	Average
001	61.68	23.16	0.04	28.29
	2.02	1.68	3.22	2.31
010	59.04	25.58	4.50	29.71
	2.27	1.99	2.09	2.12
100	37.06	28.38	2.26	22.57
	2.08	1.75	1.93	1.92
011	44.50	47.78	0.26	30.85
	2.27	2.25	3.35	2.62
101	39.68	41.94	0.10	27.24
	2.25	1.89	2.83	2.32
110	37.06	40.54	5.84	27.81
	2.21	2.02	2.21	2.15
111	27.84	39.44	0.01	22.43
	2.40	2.10	2.67	2.39
Average	43.69	35.26	1.86	26.94
	2.21	1.95	2.61	2.26

Table 1: Top lines: the average yield (to 2 d.p) of variations returned by each method and substitution scheme when asked to produce 100 variations for 100 similes. Bottom lines: the average interpretation level required for similes generated by the method and scheme. Note that 101 means that the object and description were substituted but not the aspect in the $\langle o, a, d\rangle$ simile triple, etc.

generate 100 variations – before pruning – of 100 randomly chosen similes, for each method, with every possible substitution scheme. The results are given in table 1. We see that the d and s methods yield high numbers of variations, but the w method delivers very low yields, especially when asked to find substitutes for G_d. This is because the number of synonyms for a word is less than the number of similar words, and the number of synonyms for adjectives is particularly low. Unexpectedly, replacing more parts of a simile does not necessarily lead to more similes. On inspection, this is because the increase in degrees of freedom is balanced by an increase in likelihood of pruning due to (i), (ii) or (iii) above.

In addition to observing the quantity of variations produced, we also checked the variations qualitatively. We noticed subjectively that, even out of context, certain variations were very easy to interpret, others were more challenging, and for some no suitable interpretation could be derived. For each of the methods d, s and w, we extracted 1,000 variations from those produced for table 1, and the first author subjectively hand-annotated each variation with a value 1 to 4, with 1 representing obvious similes, 2 representing similes for which an interpretation was more difficult but possible, 3 representing similes which took some time to form an interpretation for, and 4 representing similes for which no interpretation was possible. Some example similes with annotations are given in table 2. On inspection of the level 4 variations, we noted that often the problem lay in the POS-tagging of an adjective as a noun. For instance, in table 2, *kind* is identified as a noun, hence similes with nouns like *form* instead of *kind* are allowed, producing syntactically ill-formed sentences. We plan to rule this out using context-aware POS tagging, available in a number of NLP packages.

The average interpretation level for each of the substitution methods and schemes is given in table 1. We turned this analysis into a method enabling the software to control (to some extent) the level of interpretation required.

Interp. Level	Method Scheme	Variation Original
1	d	as sad as the groan of a widow
	011	as lonely as the moan of a widow
2	s	as deadly as the face of a dagger
	110	as deadly as the sting of a scorpion
3	d	as shallow as the space-time of a fork
	110	as shallow as the curve of a spoon
4	w	as form as the pump of a dove
	011	as kind as the heart of a dove

Table 2: Example simile variations, given with the interpretation level required and the original versions.

Meth.	Bound.	Naïve %	Best %	Best Method
d	1/2	72.00	75.20	RandomForest
s	1/2	60.40	65.80	LogitBoost
w	1/2	68.10	72.00	Bagging
d	2/3	59.20	68.30	OneR
s	2/3	71.20	75.70	RotationForest
w	2/3	63.20	71.10	RandomCommittee
average		65.68	71.35	

Table 3: Ten-fold cross-validation results for the best classifier on the boundary problems for each method.

To do this, given a required interpretation level n for simile variations, pairings of substitution (method, scheme) which produce an average interpretation level between n and $n+1$ in table 1 are employed. So, for instance, if similes of interpretation level 1 are required, the software uses a $(s, 001), (s, 010), (s, 100), (s, 101)$ or $(w, 100)$ pairing to generate them. To increase the performance of the approach, we used the WEKA machine learning system (Hall et al. 2009) to train a predictor for the interpretation levels which could be used to prune any variation predicted to have an interpretation level different to n. To produce the data to do so, we recorded 22 attributes of each of the 3,000 annotated similes, namely: the word frequencies [3] of each part and the minimum, average and maximum of these; the pairwise similarity [2] of each pair of parts, and the min/av/max of these; the pairwise number of collocations of each pair in the corpus of similes and the min/av/max of these; the method used for finding substitutions (d, s or w); whether the object, aspect and/or description parts have been substituted from the original; and the interpretation level.

Unfortunately, using 30 different machine learning methods in WEKA (with default settings for each), the best predictive accuracy we could achieve was 47.3%, using the RotationForest learning method. We deemed this insufficient for our purposes. However, for each variation method, we were able to derive adequate predictors for two associated binary problems, in particular (i) to predict which side of the 1/2 boundary the level of interpretation an unseen simile will be on, and (ii) the same for the 2/3 boundary. The best methods, assessed under 10-fold cross validation, and their predictive accuracy for the boundary problems for the d, s and w variation methods are given in table 3. We found that in each case, a classifier which is significantly better (as tested by WEKA using a paired T-test) than the naïve classifier had been learned, and we can expect a predictive accuracy of around 71% on average. The best learning method was dif-

ferent for each boundary problem, but some methods performed well in general. While not the best for any, the RandomSubspace method was the only one which achieved a significant classifier for all the problems. The Bagging, RotationForest, and RandomForest methods all produced significant classifiers for five of the six problems.

WEKA enables the learned predictors to be used externally, so we implemented a process whereby the generative procedure above produces potential simile variants of a given level, then the result is tested against both boundary predictors appropriate to the method. If it is predicted to fall on the wrong side of either boundary, it is rejected. As a final validation of the process, we generated 300 new simile variations, with 100 of level 1, 2 and 3 each. We mixed them randomly and tagged them by hand as before. Our hand tagging corresponded with what the software expected 82% of the time, which we believe represents sufficient control.

Combination and Instantiation

The third and fourth phases of poetry generation are more straightforward. In the combination phase, similes, variations of them and keyphrases extracted from newspaper articles are combined as per user-given templates. The templates dictate what words in each of a pair of text fragments must match exactly, what the POS tags of these words and others in the fragments must be, and how they are to be combined. Templates often simply pair two phrases together according to certain constraints, to be used in the instantiation phase later. Alternatively, they can provide more interesting ways of producing a compound phrase. The process can be iterated, so that triples, quadruples, etc., can be generated.

As an example, suppose we have the keyphrase "excess baggage" from a newspaper article about travel. This can be matched with the simile "the emotional baggage of a divorce", and presented in various ways, from simple expressions such as "the emotional excess baggage of a divorce", to the more elaborate "Oh divorce! So much emotional excess baggage", as determined by the combination template. It is possible to drop certain words, for instance the keyphrase "gorgeous history" (about a 1980s pop group) and the simile "As gorgeous as the nature of a supermodel" could produce "a supermodel-gorgeous history", and variations thereof. As a final example, keyphrases such as "emotional jigsaw puzzle" (describing a surreal play in a review) can be elaborated by combination with the simile "As emotional as the journey of a pregnancy" to produce: "An emotional jigsaw puzzle, like the journey of a pregnancy".

The retrieval, multiplication and combination stages of the process perform the most important functions, which leaves the instantiation process able to simply choose from the sets of elaborated phrases at random, and populate the fields of a user-given template. Templates allow the extraction of parts of phrases to be interleaved with user-given text, and there are also some final constraints that can be applied to the choice of phrases for the template, in particular to reduce repetition by only choosing sets of phrases where the word stems (constructed by [4]) are different.

In terms of linguistic and semantic constraints, the four stage process is quite powerful, as highlighted with the example poem given in figure 1, produced using a highly constrained search for pairs of similes. We used no simile multiplication here, in order to highlight the linguistic rather than inventive abilities. The circadian aspects of the poem are part of the template, with only the similes provided by the software. We see that the poem contains only straightforward words, because during the retrieval stage, only similes with words having frequencies in the top 5% were retrieved (as determined by [3]). Moreover, the only direct repetition is there by design in the template, and no repetition even of word stems is allowed anywhere else. This was achieved during the instantiation process, which recorded the similes used, and avoided using any word where [4] suggested the same word stem with an existing word in the poem.

Circadian No. 39

Stealthy swiftness of a leopard,
Happy singing of a bird.

Shiny luster of a diamond,
Homey feeling of a bed.

In the morning, I am loyal
Like the comfort of a friend.
But the morning grows more lifeless
Than the fabric of a rag.
And the mid-day makes me nervous
Like the spirit of a bride.

In the evening, I am solid
Like the haven of a house.
But the evening grows more fragile
Than the mindset of a child.
And the twilight makes me frozen
Like the bosom of a corpse.

Active frenzy of a beehive,
Dreary blackness of a cave.

Famous fervor of a poet,
Wily movement of a cat.

In the daytime, I am slimy
Like the motion of a snake.
But the sunlight grows more comfy
Than the confines of a couch.
And the day, it makes me tasty
Like the flavor of a coke.

In the night-time, I am hollow
Like the body of a drum.
But the moonlight grows more supple
Than the coating of an eel.
And the darkness makes me subtle
Like the color of a gem.

Stealthy swiftness of a leopard,
Happy singing of a bird.

Figure 1: An example instantiation of a user-given template.

The poem also has strictly controlled meter and stress. For instance, each two-line stanza firstly uses a simile with $\langle sw, sw, sw \rangle$ pronunciation (where s and w are syllables, with s being the stressed one), and then uses a simile with $\langle sw, sw, s \rangle$ pronunciation. This is achieved during the retrieval stage, which uses the pronunciation dictionary [1] to select only similes of the right form, and the combination process, which puts together appropriate pairs of lines. There is similar regularity in the six-line stanzas. Possibly less obvious is the subtle rhyming at play, with the final phonemes of selected pairs of lines being the same (such as beehive and cave, snake and coke, drum and gem). Moreover, inadvertent rhyming – which can be jarring – is ruled out elsewhere, for instance, snake and couch were constrained to have no rhyming, as were house and child, drum and eel, etc. The rhyming constraints come into play during the combination phase, when sets of lines are collated for final use in the stanzas. Finally, we notice that the stanzas alternate in sentiment during the course of the poem, for instance the line "Happy singing of a bird" in the first two-line stanza, contrasts starkly with the line "Dreary blackness of a cave" in the second. This is also achieved during the combination phase, which can be constrained to only put together similes of certain sentiments, as approximated by [7].

Handing over High-Level Control

We see automated poetry generation as the simultaneous production of an artefact and a context within which that artefact can be appreciated. Normally, the context is provided by the programmer/user/curator, but, as described below, to give more autonomy to the software, we enabled it to provide its own context, situated in the events of the day in which it is writing poems. In order to deliver the context alongside each poem, we also implemented a rudimentary ability to provide a commentary on the poem, and how it was produced, as described in the second subsection below.

Context Generation

In overview, the software determines a mood for the day, then uses this to choose both a Guardian article from which to extract keyphrases which will be combined with simile variations and form lines of the poem, and an aesthetic within which to assess the generated poems. These are then used to produce a set of templates for the four-stage poem generation process described above. Finally, the software instantiates the templates to produce a set of poems, and chooses to output the one which maximises the aesthetic.

As in the automated collage generation of (Krzeczkowska et al. 2010), the software appeals to daily newspaper articles for raw material. We extend that approach by also using the articles to derive a mood, from which an aesthetic is generated. In particular, each of the 12,820 articles in the corpus has been assigned a sentiment value between -5 and 5, as the average of the sentiment of the words in the article, assessed by [7]. Thus, when a poetry generation session begins, the software is able to check the sentiment of the set N of newspaper articles posted during the previous 24 hours, and if it is less than the average, the software determines the mood as *bad*, or *good* otherwise. If the mood is good, then an article, A, from the happiest five articles from N is chosen, with melancholy articles similarly chosen during a bad mood. The keyphrases, $key(A)$, are then extracted from the article, and we denote as $words(A)$ the set of words appearing in $key(A)$. Note that very common words such as "a", "the", "of", etc., are removed from $words(A)$.

As an example, on 17/01/2012, the mood was assessed as bad, and a downbeat article about the Costa Concordia disaster was retrieved. In contrast, on 24/01/2012, the mood was assessed as good, and an article describing the buoyant nature of tourism in Cuba was retrieved, from which keyphrases such as "beach resorts", "jam-packed bar", "seaside boulevard" and "recent sunny day" were extracted using [6]. Note that [6] also returns a relevancy score for each keyphrase, e.g., "recent sunny day" was given a score of 0.48 for relevance, while "jam-packed bar" only scored 0.31.

The mood is sufficient to derive an aesthetic within which to create poems, but this will be projected partly through members of $words(A)$ appearing in the poem, and mood is only one aspect of the nature of a poem. Letting $words(P)$ denote the words in poem P, for more variety, the software can choose from the following four measures:

- **Appropriateness:** the distance between the average sentiment of the words in $words(P)$ from 5 if it is a good mood day, or from -5 if it is a bad mood day.
- **Flamboyance:** the average of $f(w)$ over $words(P)$, where $f(w) = 0$ if $w \in words(A)$ and $f(w) = 1/frequency(w)$ if $w \notin words(A)$, where frequency is calculated by [4].
- **Lyricism:** the proportion of linguistic constraints adhered to by P, with the constraints determined by the set of templates generated for the poem, as described below.
- **Relevancy** to the Guardian article: the average of $rel(w)$ over $words(P)$, where $rel(w) = 0$ if $w \notin words(A)$ and $rel(w)$ is the relevancy [6] of w, if $w \in words(A)$.

The choice of which set of measures, M, to use in the aesthetic for a poem is determined somewhat by A and $key(A)$. In particular, if A is assessed as being in the most emotive 10% of articles ever seen (either happy or sad), then M is chosen as either {Appropriateness} or {Appropriateness, Relevance} in order to give due consideration to the gravity or brevity of the article. If not, and the size of $key(A)$ is less than 20% of the average over the corpus, then it might be difficult to gain relevancy to A in the poem, hence M is chosen as $\{Relevance\}$. In all other cases, M is chosen randomly to consist of either 1 or 2 of the four measures – we found that mixing more than 2 diluted their effect, leading to poems with little discernible style.

The software also generates templates to dictate the structure of the poem. The number of stanzas, z, is taken to be between 2 and 10, with the number dictated by the size of $key(A)$, i.e., larger poems are produced when $key(A)$ is relatively large, compared to the rest of the corpus. The structure of the poem can be *equal*, i.e., of the form $A_1A_2 \dots A_z$ with each stanza A_i being of the same length (chosen randomly between 2 and 6 lines). The structure can also be chosen to be *alternating* of the form $A_1B_2A_3 \dots A_z$ or $A_1B_2A_3 \dots B_z$; or *bookended* of the form $A_1B_2 \dots B_{z-1}A_z$. The choice of structure is currently made randomly, and there is no relationship between pairs of stanzas, except that the templates constrain against the usage of a new phrase (combined from a keyphrase and simile as described above) in the template if one of the words has the same stem as a word in an already-used phrase. As part of the template generation, the software chooses the number of times (between 0 and 5) this constraint is allowed to be broken per phrase, as a level of repetition can add flavour to a poem. Note that the counts per phrase are reset to zero if the software runs out of phrases to add to the template.

If M contains the $Lyricism$ measure, then the templates are also constrained to express some linguistic qualities, which are added at the stanza level. In particular, the line structure of all stanzas of type A is chosen to be either equal, alternating or bookended in the same fashion as the stanza structure, with stanzas of type B also given a structure. This structure allows linguistic constraints to be added. For instance, if a stanza has alternating structure $abab$, the software chooses a single linguistic constraint from: *syllable-count*, *end-rhyme*, *start-rhyme*, and constrains all lines of type a accordingly. It does the same (with a possibly different linguistic constraint) for lines of type b. Note that *syllable-count* means that the two lines should have the same

number of syllables as each other (within a threshold of two syllables), *end-rhyme* means that the two lines should at least end in the same phoneme, with *start-rhyme* similar.

The random nature of the choices to fill in the final poem template ensures variety. In each session, the software generates 1,000 poems, and their scores for each of the measures in M are calculated. The average rank over the measures is taken as an overall rank for each poem, and the highest ranked is presented as the poem for the day. If the templates over-constrain the problem and no poems are produced, then a single constraint is chosen to be dropped, and the session re-started iteratively until poems are produced.

Commentary Generation

In addition to the four stage process of retrieval, multiplication, combination and instantiation, the software chooses a Guardian article, performs sentiment analysis, aesthetic invention, template construction and searches for appropriate poems. While some of these methods are at present rather rudimentary and perhaps a little arbitrary, it is our hypothesis that a well-formed commentary about how the software has produced a poem will provide a context for the poem and add value to the appreciation process, as argued above.

In order for the software to generate the commentary, we re-use the four stage process, but with the retrieval stage sampling not from corpora of human produced text, but rather from a set of recorded statements about how each of the processes worked, and what they produced. In particular, the software records details such as (a) the mood of the day (b) the Guardian article it retrieved and how emotive it was (c) the keyphrases extracted, which sentences they came from, and which were used in the final poem (d) the combinations of keyphrase and similes it produced (e) the nature of the poem structure dictated by the template, (f) the aesthetic weightings used, and (g) what successes and failures it had in instantiating the templates. We have produced by hand a number of (sets of) commentary templates that can present the statements in a supportive way. Currently, the software randomly chooses which set of templates to use to generate the commentary. The software chooses the title for each poem as the keyphrase occurring the most often in the poem, choosing randomly if there is a tie for the most used.

Illustrative Examples

We artificially situated the software in the days from 1/01/2012 to 10/02/2012, and asked it to produce a single poem for each day, along with a commentary. We added the constraint that the poem should be exactly four stanzas in length for presentation purposes in this paper. We curated three for presentation here, in figure 2 below. The commentaries are meant to provide enough context for proper appreciation of each poem, so we will not add detail here to the commentaries of the individual poems. Viewing the entire set of generated poem/commentaries subjectively, we were disappointed by the number of compound sentences available for the templates. Even with large sets of keyphrases extracted from an article, and extensive simile multiplication employed, we found that there were few opportunities for a simile to be used for embellishment, which meant that the software had limited choices for the final poem template, which led to an over-reliance on repeating lines, or using similar lines. More importantly, the differences in the aesthetic evaluations over the 1,000 poems generated for a day were not great, hence the aesthetic generation was driving the production of poems less than we would have liked.

Conclusions and Future Work

We agree with (Pease and Colton 2011b) that Turing-style tests encourage naïvety and pastiche in creative systems. However, eschewing their use leaves a hole regarding proper evaluation of our poetry generation system. Instead, we can turn to the FACE descriptive model put forward in (Colton, Charnley, and Pease 2011) and (Pease and Colton 2011a), which advocates describing a creative system in terms of the *creative acts* it performs, which are in turn tuples of generative acts. The generative acts produce outputs of four types: examples of concepts, concepts themselves, aesthetic measures which can evaluate concept/example pairs, and framing information. Looking at the literature review above, the WASP and Electronic Text Composition systems can be considered as generating concepts, as can any system which generates and employs a statistical model of written or verbal language (such as in Markovian approaches). It does not appear that any system invents aesthetic measures or produces framing information such as a commentary which can be used as a context for the poem. Hence, according to the FACE model, our approach can be considered favourably, as it has processes producing examples (instantiation), concepts (template generation), aesthetics (choosing measures) and framing information (producing commentaries), within the creative act of poem generation. This represents an advance in the state of the art of automatic poetry generation.

It is clear that many aspects of the process presented here are fairly rudimentary, often with random choice substituting a reasoned approach. Our main contribution has been to implement a rounded system which can function on the majority of levels required to be taken seriously as a poet, albeit in a simplistic manner. We plan further enhancements to all of the processes described above, including (i) implementing improved ways to generate phrases for templates, as the yield is currently too low to enable the software to use its more advanced linguistic constraining features (ii) working with other corpora (iii) enabling the software to automatically add higher level structures to poems via the kinds of narratives seen in the circadian poem given above, and (iv) turn the commentary generation processes into full-story telling, which may include the introduction of fictions. After the enhancements, we will work with a poet and explore gaining critical feedback via the publication of anthologies.

While the *imitation-game* aspect of Turing-style tests are not conducive for creativity, we do applaud the usage of *dialogue* they prescribe. Indeed, in the future, we imagine all creative systems being enhanced with a story generator able to produce both static framing information, and to reply with a story to any question asked of it in a dialogue situation. We believe that only with such abilities will software systems be taken seriously as creative entities in the cultural world.

It was generally a bad news day. I read an article in the Guardian entitled: "Police investigate alleged race hate crime in Rochdale". Apparently, "Stringer-Prince, 17, has undergone surgery following the attack on Saturday in which his skull, eye sockets and cheekbone were fractured" and "This was a completely unprovoked and relentless attack that has left both victims shocked by their ordeal". I decided to focus on mood and lyricism, with an emphasis on syllables and matching line lengths, with very occasional rhyming. I like how words like attack and snake sound together. I wrote this poem.

Relentless attack

a glacier-relentless attack
the wild unprovoked attack of a snake

the wild relentless attack of a snake
a relentless attack, like a glacier
the high-level function of eye sockets

a relentless attack, like a machine
the low-level role of eye sockets
a relentless attack, like the tick of a machine

the high-level role of eye sockets
a relentless attack, like a bloodhound

It was generally a good news day. I read a story in the Guardian culture section entitled: "South Africa's ANC celebrates centenary with moment in the sun". It talked of south africans, interfaith prayers and monochrome photos. Apparently, "The heroic struggle against a racist regime was remembered: those thousands who sacrificed their lives in a quest for human rights and democracy that took more than eight decades" and "At midnight he watched with amusement as Zuma lit the centenary flame, at the second attempt, with some help from a man in blue overalls marked 'Explosives'". I wanted to write something highly relevant to the original article. I wrote this poem.

Blue overalls

the repetitive attention of some traditional african chants
a heroic struggle, like the personality of a soldier

an unbearable symbolic timing, like a scream
blue overalls, each like a blueberry
some presidential many selfless leaders

oh! such influential presidents
such great presidents
blueberry-blue overalls

lark-blue overalls
a knight-heroic struggle

It was generally a bad news day. I read a story in the Guardian entitled: "Thai police hunt second bomb plot suspect in Bangkok". It talked of suspected bomb plotters, lebanese men and travel alerts. Apparently, "Sketches released late on Friday night by Thai police showed the suspect as a white Middle-Eastern man with short hair and stubble, around 1.8m (5ft 9in) tall". It's a serious story, but I have concentrated on flourishes today. I wrote this poem.

Foreign embassies

the wiry militant arm of a doorman
a white middle-eastern man, like a snowball
spaceship-foreign embassies
foreign embassies, each like a stranger

an impersonal suvarnabhumi international airport
a white middle-eastern man, like the surface of a porcelain
the sturdy design of a bangkok post

foreign embassies, each like a spaceship
an impersonal suvarnabhumi international airport
stranger-foreign embassies
the stout engineering of a bangkok post

a white middle-eastern man, like the skin of an earthenware
foreign embassies, each like a stranger
spaceship-foreign embassies

Figure 2: Illustrative poems and commentaries. For the Guardian articles on which these poems are based, please see: www.guardian.co.uk, followed by: /uk/2012/feb/09/police-race-hate-crime-rochdale /world/2012/jan/08/south-africa-anc-centenary /world/2012/jan/15/thai-second-bomb-suspect-bangkok

Acknowledgements

This work has been funded by EPSRC grant EP/J004049. Many thanks to the reviewers for their insightful comments.

References

Addad, E. 2010. Interactive poetry generation systems: an illustrated overview. netpoetic.com/2010/10/interactive-poetry-generation-systems-an-illustrated-overview/.

Carpenter, J. 2004. Electronic text composition project. http://slought.org/content/11199.

Chamberlain, W., and Etter, T. 1984. *The Policeman's Beard is Half-Constructed: Computer Prose and Poetry*. Warner Books.

Colton, S.; Charnley, J.; and Pease, A. 2011. Computational creativity theory: the FACE and IDEA models. In *Proceedings of the 2nd International Conference on Computational Creativity*.

Díaz-Agudo, B.; Gervás, P.; and González-Calero, P. 2002. Poetry generation in COLIBRI. *Advances in Case-Based Reasoning* 2416.

Elhadad, M.; Gabay, D.; Goldberg, Y.; and Netzer, Y. 2009. Gaiku: Generating haiku with word associations norms. In *Proc. of the Workshop on Computational Approaches to Linguistic Creativity*.

Gervás, P. 2010. Engineering linguistic creativity: Bird flight and jet planes. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*.

Greene, E.; Bodrumlu, T.; and Knight, K. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. 2009. The WEKA data mining software: An update. *SIGKDD Explorations* 11(1).

Hartman, C. 1996. *Virtual Muse: Experiments in Computer Poetry*. Wesleyan University Press.

Kilgarriff, A. 1997. Putting frequencies in the dictionary. *International Journal of Lexicography* 10 (2).

Kolb, P. 2008. DISCO: A multilingual database of distributionally similar words. In *Proceedings of KONVENS*.

Krzeczkowska, A.; El-Hage, J.; Colton, S.; and Clark, S. 2010. Automated collage generation – with intent. In *Proceedings of the 1st International Conference on Computational Creativity*.

Lutz, T. 1959. Stochastische texte. *Augenblick* 4(1).

Manurung, R.; Ritchie, G.; and Thompson, H. 2012. Using genetic algorithms to create meaningful poetic text. *JETAI* 24(1):43–64.

Manurung, H. 2004. An evolutionary algorithm approach to poetry generation. *PhD. Thesis*, University of Edinburgh.

Mihalcea, R., and Tarau, P. 2004. Textrank: Bringing order into texts. In *Proc. of the Conference on Empirical Methods in NLP*.

Montfort, N., and Strickland, S. 2010. Sea and spar between. http://blogs.saic.edu/dearnavigator/winter2010/nick-montfort-stephanie-strickland-sea-and-spar-between/.

Montfort, N. 2009. The ppg256 series of minimal poetry generators. In *Proceedings of Digital Arts and Culture 2009*.

Pease, A., and Colton, S. 2011a. Computational creativity theory: Inspirations behind the FACE and IDEA models. In *Proceedings of the 2nd International Conference on Computational Creativity*.

Pease, A., and Colton, S. 2011b. On impact and evaluation in computational creativity: A discussion of the Turing test and an alternative proposal. In *Proc. AISB symp. on AI and Philosophy*.

Porter, M. 1980. An algorithm for suffix stripping. *Program* 14 (3).

Queneau, R. 1961. *Cent mille milliards de poèmes*. Gallimard.

Rahman, F., and Manurung, R. 2011. Multiobjective optimization for meaningful metrical poetry. In *Proceedings of the 2nd International Conference on Computational Creativity*.

Roque, A. 2011. Language technology enables a poetics of interactive generation. *The Journal of Electronic Publishing* 14.

Veale, T., and Hao, Y. 2011. Exploiting readymades in linguistic creativity: A system demonstration of the jigsaw bard. In *Proc. of the 49th Annual Meeting of the ACL*.

Veale, T. 2006. Tracking the lexical Zeitgeist with Wordnet and Wikipedia. In *Proceedings of the 17th European Conference on Artificial Intelligence*.

Wong, M., and Chun, A. 2008. Automatic haiku generation using VSM. In *Proceedings of ACACOS*.