PoeTryMe: a versatile platform for poetry generation

Hugo Gonçalo Oliveira[1]

Abstract. PoeTryMe is a platform for the automatic generation of poetry. It has a versatile architecture that provides a high level of customisation. The user can define features that go from the poem configuration and the sentence templates, to the initial seed words and generation strategy. A prototype was implemented based on PoeTryMe to generate Portuguese poetry, using natural language processing resources for this language, and patterns that denote semantic relations in human-created poetry. The possible results are illustrated by three generated poems.

1 INTRODUCTION

Natural language generation [23] is a well-established sub-field of artificial intelligence and computational linguistics. Its main goal is to develop computer programs capable of producing text that is understood by humans. Biographies [15] and weather forecasts [2] are examples of the genres of text that have been generated automatically. Another example is the generation of text with creative features, including story narratives [3], jokes [24] or poetry (see section 2).

We have seen several attempts to generate creative artifacts automatically, with the help of computer programs, and we now accept the computer as an artist. The creation of visual art and the composition of musical pieces are other fields where creative systems have been developed for.

In this paper, we present PoeTryMe, a platform designed for the automatic generation of poetry. Given a generation grammar and a set of relational triples, PoeTryMe generates grammatically correct and meaningful sentences. It has a versatile architecture that provides a high level of customisation and can be used as the base of poetry generation systems, which can be built on the top of it. In PoeTryMe, everything can be changed: the base semantics, represented as relational triples; the templates of the generated sentences, included in the generation grammars; the generation strategies, that select the lines to include in the poem; and, of course, the poem configuration.

We start this paper by referring some work on the automatic generation of poetry, including two categorisations for this kind of systems. Then, we present an overview on the architecture of PoeTryMe, followed by the description of a prototype, implemented for the generation of Portuguese poetry. While introducing the external resources used, we describe the process for acquiring line templates, and the implemented generation strategies. Following, we illustrate the possible results of PoeTryMe by presenting three generated poems. Before concluding with some cues for future work, we categorise the implemented strategies.

[1] CISUC, University of Coimbra, Portugal, email: hroliv@dei.uc.pt, supported by FCT scholarship grant SFRH/BD/44955/2008, co-funded by FSE

2 RELATED WORK

The automatic generation of poetry is a complex task, as it involves several levels of language (e.g. phonetics, lexical choice, syntax and semantics) and usually demands a considerable amount of input knowledge. However, what makes this task more interesting is that some of the latter levels do not have to be strictly present.

On the one hand, writing poetic text does not have to be an extremely precise task [9], as several rules, typically present in the production of natural language, need to be broken [18]. For instance, there may not be a well-defined message. On the other hand, poetry involves a high occurrence of interdependent linguistic phenomena where rhythm, meter, rhyme and other features like alliteration and figurative language play an important role.

In this section, we present two categorisations of poetry generation systems, proposed in the literature. One of them considers the applied techniques and another the generated text.

2.1 Poetry generation techniques

Regarding the followed approaches and techniques used, poetry generation systems can be roughly grouped into four categories [8]: (i) template-based, which includes systems that just fill templates of poetry forms with words that suit syntactic and/or rhythmic constraints; (ii) generate-and-test; (iii) evolutionary; and (iv) case-based reasoning.

In generate-and-test systems, random word sequences are produced according to formal requirements, that may involve meter or other constraints. Manurung's chart system [17], WASP [9] and the generate-and-test strategy of Tra-la-Lyrics [12] are systems that fall into this category.

In Manurung's chart system, sentences are logically represented by first order predicates describing the input semantics, and charts are used to generate natural language strings that match a given stress pattern. While a chart parser analyses strings and translates them to logical forms, a chart generator translates logical forms to strings. During the generation, before adding the result of a new rule to the chart, its stress pattern is checked for compatibility with the target pattern. Only results with compatible patterns are added, ensuring that the generated text satisfies the pattern. WASP is a forward reasoning rule-based system that aims to study and test the importance of the initial vocabulary, word choice, verse pattern selection and construction heuristics, regarding the acceptance of the generated verses and complete poems. Tra-la-Lyrics [13, 12] is a system that aims to generate text based on the rhythm of a song melody, given as input. Using the sequence of strong and weak beats as a rhythmic pattern, the task of generating song lyrics is very similar to the generation of poetry. In the generate-and-test strategy, grammatical sentences are produced and then scored according to their suitability to a given meter/rhythmic pattern.

Evolutionary approaches rely on evolutionary computation techniques. POEVOLVE [16] and McGonnagall [18, 19] are examples of such approaches. POEVOLVE is a prototype that generates limericks, implemented according to a model that takes the real process of human poetry writing as a reference. In McGonnagall, the poem generation process is formulated as a state space search problem using stochastic hill-climbing search, where a state in the search space is a possible text with all its underlying representations, and a move can occur at any level of representation, from semantics to phonetics. The search model is an evolutionary algorithm encompassing evaluation and evolution.

As for case-based reasoning approaches, existing poems are retrieved, considering a target message, and then adapted to fit in the required content. Systems like ASPERA [10] and COLIBRI [5] fall into this category. They are forward reasoning rule-based systems that, given a prose description of the intended message and a rough specification of the type of poem, select the appropriate meter and stanza, generate a draft poem, request modification or validation by the user, and update their database with the information of the validated verse.

2.2 Generated poetry properties

Manurung [18] affirms that poetic text must hold all the three properties of meaningfulness, grammaticality and poeticness. More precisely, it must: (i) convey a conceptual message, which is meaningful under some interpretation; (ii) obey linguistic conventions prescribed by a given grammar and lexicon; and (iii) exhibit poetic features. An alternative categorisation for poetry generation attempts considers the latter properties and divide systems into the following: (i) word salad, which just concatenate random words together, without following grammatical rules, therefore not holding any of the properties; (ii) form-aware; and (iii) actual poetry generation systems.

In form-aware systems, the choice of words follows a pre-defined textual form, by following metrical rules. They thus hold the properties of grammaticality and poeticness. The WASP system [9], POEVOLVE [16], and the generative grammar strategy of Tra-la-Lyrics [13] fall into this category.

Actual poetry generation systems must hold the three properties. ASPERA [10] and COLIBRI [5] are examples of such systems. In both of them, words must be combined according to the syntax of the language and should make sense according to a prose message provided by the user. Also, when occurring at the end of lines, words may have additional constraints imposed by the strophic form. McGonnagall [18, 19] falls into this category as well, given that a goal state is a text that satisfies the three aforementioned properties. However, after several experimentations, Manurung et al. [19] state that it is difficult to produce both semantically coherent text in a strict agreement to a predefined meter.

There are other systems whose results exhibit poetic features, obey syntactic rules and, even though not following a well-defined and precise message, try to generate meaningful text, as they select sentences or words based on given seeds or semantic similarity. Examples of those include Wong and Chun's [25] haiku generator, Gaiku [20], and Ramakrishnan's lyrics generator [22, 21]. Wong and Chun generate haikus using a Vector Space Model (VSM), established by sentences in blogs. Candidate sentences are selected according to their semantic similarity. Gaiku generates haikus based on a lexical resource that contains similar words. Haikus are generated according to a selected theme and syntactic rules. Ramakrishnan el al. learned a model of syllable patterns from real melodies. The model was used in a system that, given a melody, generates meaningful sentences that match adequate syllabic patterns and rhyme requirements. Meaningful sentences were generated with the help of n-gram models, learnt from a text corpus. In a more recent version of the system [21], meaningful sentences are generated with the help of a knowledge base.

Furthermore, the random words strategy of Tra-la-Lyrics [13] falls in what can be considered as a fourth category, as the meter of the generated text suit the given rhythm and the text contains poetic features (rhyme), but the word order does not follow grammatical rules and there are no semantic constraints. In other words, this strategy holds the property of poeticness, but none of the others.

Recently, a novel system was presented for poetry generation [4] where, besides dealing with the three aforementioned properties, poems are generated regarding the mood for a certain day (good or bad), according to newspaper articles, and an aesthetic is produced, using a set of measures (appropriateness to the mood, flamboyance, lyricism and relevance for a selected article). The lines of the poem are collected not only from the articles, but also from short phrases mined from the Internet, and variations of the latter obtained by replacing some words with others semantically similar. Moreover, comments, supporting the choices made (e.g. mood, used sentences, aesthetic measures), are generated. While the latter contextualise the poem, the produced aesthetics may be used to evaluate the obtained results more objectively.

3 PoeTryMe

PoeTryMe is a poetry generation platform that relies on a modular architecture (see Figure 1) and thus enables the independent improvement of each module. This architecture intends to be versatile enough to provide a high level of customisation, depending on the needs of the system and ideas of the user. It is possible to define the semantics to be used, the sentence templates in the generation grammar, the generation strategy and the configuration of the poem. In this section, the modules, their inputs, and interactions are presented.

3.1 Generation Strategies

A Generation Strategy implements a method that takes advantage of the Sentence Generator to obtain lines and build up a poem. The poem is generated according to a set of seed words, used to get sentences from the Sentence Generator, and a poem template. The latter contains the poem's structure, including the number of stanzas, the number of lines per stanza and the number of syllables of each line. Figure 2 shows the representation of poem structure templates, for generating a sonnet and for a haiku. In the latter, the Portuguese word *estrofe* indicate a stanza and the *verso* indicates a line.

An instantiation of the Generation Strategy does not generate sentences. It just includes one or several heuristics to find the better sentences for each line, obtained from the Sentence Generator. Heuristics might consider features like meter, rhyme, coherence between lines or other, depending on the poem's purpose. In our prototype (see section 4), we have implemented a basic strategy, a generate-and-test strategy, and an evolutionary approach.

3.2 Sentence Generator

The Sentence Generator is the core module of PoeTryMe's architecture and is used to generate meaningful sentences with the help of:

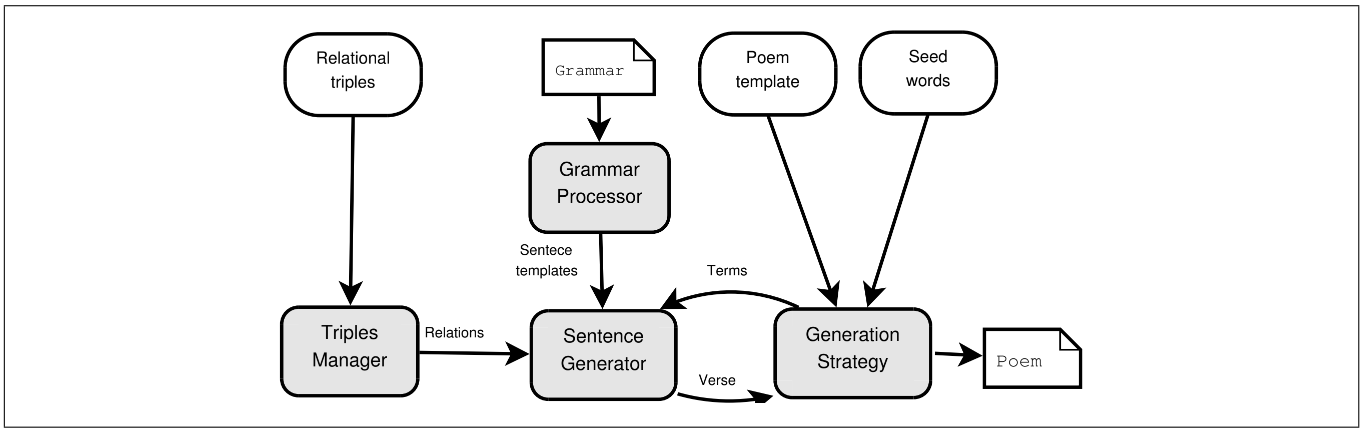

Figure 1. PoeTryMe architecture

```
#sonnet
estrofe{verso(10);verso(10);verso(10);verso(10)}
estrofe{verso(10);verso(10);verso(10);verso(10)}
estrofe{verso(10);verso(10);verso(10)}
estrofe{verso(10);verso(10);verso(10)}
```

```
#haiku
estrofe{verso(5);verso(7);verso(5)}
```

Figure 2. First, the structure of a sonnet, and then, the structure of a haiku.

- a semantic graph, managed by the Triples Manager, where the nodes are words and the edges are labelled according to a relation type. A tuple $t = (node_1, relation\_type, node_2)$ establishes a relational triple;
- generation grammars, processed by the Grammar Processor, which contain textual templates for the (chart) generation of grammatical sentences denoting a semantic relation.

The generation of a sentence starts with a set of seed words, used to select a subgraph from the main semantic graph. The former contains only relations involving the seed words, or connected indirectly to them from a path no longer than a predefined depth δ.

Generation proceeds by selecting a random triple in the subgraph and a random grammar rule matching its relation type. There must be a direct mapping between the relation names, in the graph, and the name of the head rules, in the grammar. After inserting the arguments of the triple in the rule body, the resulting sentence is returned.

Similarly to Manurung [17], the Grammar Processor uses a chart-parser in the opposite direction, in order to perform chart generation. The body of the rules should consist of natural language renderings of semantic relations. Besides the simple terminal tokens, that will be present in the poem without any change, the Grammar Processor supports special terminal tokens that indicate the position of the relation arguments (`<arg1>` and `<arg2>`), to be filled by the Sentence Generator.

4 POETRY GENERATION IN PORTUGUESE

This section is about the prototype implemented in the top of PoeTryMe, to generate Portuguese poetry. We present the natural language processing resources used in the prototype, list some renderings for semantic relations, included in the grammars after exploiting human-created poetry, describe the implemented generation strategies, and show three examples of generated poems.

4.1 Resources used

PEN[2] is an implementation of the Earley [6] chart-parsing algorithm that analyses sentences according to grammars given as input. These grammars are editable text files, where each line contains the name of a rule and its body. In order to differentiate rule tokens from terminals, rule names are upper case. An example of a simple and valid rule set is shown in Figure 3, where the Portuguese word `RAIZ`, meaning root, is the starting point of the grammar. We used PEN in the opposite direction, in order to perform chart generation.

```
RAIZ ::= RULE
RAIZ ::= RULE <&> OTHERRULE

RULE ::= terminal
OTHERRULE ::= otherterminal
OTHERRULE ::= otherterminal <&> OTHERRULE
```

Figure 3. PEN example rules.

CARTÃO [11] is a public lexical knowledge base for Portuguese, extracted automatically from three Portuguese dictionaries. It contains about 325,000 semantic triples, held between words, which can be used as a semantic graph. A semantic triple, represented as follows, indicates that one sense of the word in the first argument (`arg1`) is related to one sense of the word in the second (`arg2`) by means of a relation identified by `RELATION_NAME`:

[2] Available from http://code.google.com/p/pen/

```
arg1 RELATION_NAME arg2
e.g. animal HIPERONIMO_DE cão
(animal HYPERNYM_OF dog)
```

CARTÃO includes relations as synonymy, hypernymy, part-of, causation, purpose and property, amongst others. The name of the semantic relation also defines the part-of-speech of its arguments.

SilabasPT[3] is an API that performs syllabic division and stress identification for Portuguese words. It was developed to help generating text based on rhythm in the project Tra-la-Lyrics [13, 12], but it is an independent API that can be integrated in other applications.

LABEL-LEX[4] is a lexicon of Portuguese, with 1,5 million inflected word forms, automatically generated from about 120,000 lemmas. For each word form, it provides information such as the lemma, the part-of-speech and other morphological information.

4.2 Relations and renderings

Instead of creating our own grammars manually, we automatised this task by exploiting real Portuguese poetry. It is a well known fact that semantic relations can be expressed in running text by discriminating patterns, typically used to discover new relations (see, for instance, [14]). Therefore, in order to discover patterns for our grammar, we extracted all sentences in a collection of Portuguese poetry[5], where the arguments of, at least, one triple of CARTÃO co-occurred.

After replacing the arguments by terminal tokens, relative to the first and the second argument (`<arg1>` and `<arg2>`), we added the sentence as a rule in the grammar with the name of the relation. Table 1 shows examples of the relations used, example arguments, and automatically discovered patterns, used as renderings for the relations. About 700 patterns were discovered.

In order to deal with inflected words and to keep number and gender agreement in the generated sentences, before discovering the patterns, we added the number and the gender of the noun and adjective arguments to the relation name. For instance, the triple (*destino* synonym-of *futuro*) was changed to (*destino* ms-synonym-of-ms *futuro*), while the triple (*versos* part-of *quadras*) was changed to (*versos* mp-part-of-fp *quadras*). However, for the sake of clarity, we did not include this information in table 1. The number and gender information was obtained from LABEL-LEX.

4.3 Implemented generation strategies

Three different generation strategies were implemented in the prototype. While one is just used as a baseline for debugging, the others follow evolutionary approaches, as Manurung's [18] algorithm for poetry generation.

In both of the latter strategies, there is an evaluation function that scores each sentence according to the absolute difference between the number of syllables the poem line has in the template, with the number of syllables in the generated sentence – the lower the evaluation, the better the sentence is. SilabasPT is used to count the number of syllables of each sentence and identify its last stress. The final score of a poem, used only in the third strategy, is the sum of the scores of all lines plus a bonus for poems with lines in the same stanza with the same termination (rhyme). The other strategies do not score rhymes because they do not generate the poem as a whole, but just gather lines independently.

[3] Available from http://code.google.com/p/silabaspt/
[4] Available from http://label.ist.utl.pt/pt/labellex_pt.php
[5] We used `wget` to collect all the poems in the portal *Versos de Segunda*, available from http://users.isr.ist.utl.pt/c̃fb/VdS/

The algorithms involved in each one of the strategies are briefly described as follows:

- Basic: for each line to be filled, a random sentence is generated using the key terms;
- Generate-and-test: for each line to be filled, n random sentences are generated. The one with best score is chosen. All unused sentences are indexed and can be used if a new line needs exactly the same amount of syllables of the previously unused sentence.
- Evolutionary: an initial population of n poems is generated using the basic strategy. Then, each poem is scored according to the aforementioned evaluation function. Each new generation consists of the poems with the best evaluation, poems that are the result of crossing two random poems in the population, and newly created poems as well. When two poems are crossed, a new poem is created with lines selected from both. The best scoring poem of the last generation is returned.

4.4 Illustrative results

For illustration purposes, we present three poems obtained with the implemented prototype. In figure 4, we present a haiku, obtained with the generate-and-test strategy, using 100 generations per line, and the seed words *arte* and *paixão* (in English, art and passion), with $\delta = 1$. With more depth, the system has more word choices and thus more variations, but it is less focused on the seed words. On the other hand, using $\delta = 1$, each line will include one seed word, which is the case for the presented haiku.

The example follows the 5-7-5 syllable pattern correctly. However, the choice of words for the haiku must be done carefully, because long words prevent the generation of short lines.

```
ah paixão afecto
não tem paixão nem objecto
sem na arte modos
```

Figure 4. Example of a generated haiku.

In figure 5, we present a sonnet, this time obtained with the evolutionary approach, after 25 generations of 100 poems. In each generation, the population consisted of 40% of the best poems from the previous, 40% resulting from crossing, and 20% new. The probability of crossing, which consists of swapping two lines of two different poems, was set to 50%, and the bonus for rhymes in the end of lines to -3. Once again, we used $\delta = 1$. However, as a sonnet has fourteen lines, in order to have more variations, we used more seed words, namely: *computador*, *máquina*, *poeta*, *poesia*, *arte*, *criatividade*, *inteligência*, *artificial* (computer, machine, poet, poetry, art, creativity, intelligence, artificial).

The meter of the poem is very close to ten syllables per line. Only the third and seventh line have one additional syllable. Also, all the verses include one of the seeds and a related word. Meaning is present in each isolated verse and thus, a meaning emerges for the whole poem. However, there are no rhymes, which suggests that the bonus is not enough to generate poems with this feature.

Even so, in an attempt to force the poems to have rhymes, we generated more poems with the evolutionary approach, with similar

Type	POS	Example args.	Example rule
Synonym-of	noun,noun	*destino,futuro* *(destiny,future)*	`não sei que <arg1> ou <arg2> compete á minha angústia sem leme`
	adj,adj	*quebrada,rota* *(broken,ragged)*	`<arg1> a espada já <arg2> a armadura`
Antonym-of	adj,adj	*possível,impossível* *(possible,impossible)*	`tudo é <arg1>, só eu <arg2>`
Hypernym-of	noun,noun	*mágoa, dor* *(sorrow,heartache)*	`e a própria <arg2> melhor fora <arg1>`
Part-of	noun,noun	*versos,quadras* *(lines,blocks)*	`as minhas <arg2> têm três <arg1>`
Causation-of	noun,noun	morte,luto *(death,grief)*	`a seca, o sol, o sal, o mar, a morna, a <arg1>, a luta, o <arg2>`
	verb,noun	*dor,doer* *(pain,to_hurt)*	`é <arg2> que desatina sem <arg2>`
Purpose-of	noun,noun	*arma,munição* *(weapon,ammunition)*	`com <arg2> sem <arg1>`
	verb,noun	*taça,beber* *(cup,to_drink)*	`<arg1> para <arg2> junto á perturbada intimidade`
Has-quality	noun,adj	*certeza,certo* *(certainty,sure)*	`eu que não tenho nenhuma <arg1> sou mais <arg2> ou menos <arg2>`
Property-of	adj,noun	*letal,morte* *(letal,death)*	`a <arg2> é branda e <arg1>`

Table 1. Automatically discovered renderings, included in the grammars.

```
e não há deus nem preceito nem arte
um palácio de arte e plástica
as máquinas pasmadas de aparelhos
num mundo de poesias e versos

o seu macaco era duas máquinas
horaciano antes dos poetas
para as consolas dos computadores
num mundo de poesias e carmes

longas artes apografias cheias
tenho poesias como a harpa
poema em arte modelação

somos artificiais teatrais
máquinas engenhocas repetido
um poeta de líricas doiradas
```

Figure 5. Example of a generated sonnet.

settings as the previous, except for: (i) the bonus for rhymes, which was set to -10; (ii) δ was set to 2; (iii) regarding the higher value of δ, the provided seed words were only two, more precisely, they were the same as in the first presented poem (*arte* and *paixão*).

One of the resulting poems is a block of four lines, presented in figure 6. All the lines of this poem end with the same termination, but none of them agrees with the correct metrics. Actually, all the lines have more syllables than they should – one in the first and third lines, six in the second and four in the fourth. Regarding the semantics of the poem, it is less focused on the seeds, as expected, and none of them is actually used. Still, the poem contains words related to art, as *escultura*, *representação* and *composição* (sculpture, acting, composition).

As others have noticed for meaningfulness and poeticness [19], we confirmed that it is difficult to generate a poem that strictly obeys to the three properties of poetry generation without relaxing on, at least,

```
ah escultura representação
e os que dão ao diabo o movimento da convulsão
sua composição de preparação
é destino estar preso por orientação
```

Figure 6. Example of a generated block of four lines.

one of them. Moreover, the performed experiments showed that the generate-and-test strategy, with 100 or more generations of each line, result more consistently in poems with better evaluation. However, as it is, the latter strategy does not have bonus for rhymes, and they will only occur by chance, as in the poem of figure 4. On the other hand, the evolutionary approach is more complex and has parameters that should be deeper analysed, but can generate poems with rhymes in a trade-off for less precise meter.

5 CATEGORISATION

Regarding that we have implemented different strategies for generating poetry, the work presented here falls in more than one category. Although our approach uses sentence templates, only one is actually template-based (basic strategy), while the other two follow, respectively, a generate-and-test and an evolutionary approach.

As for their goals, since we use a semantic graph as input and we render information in it to natural language sentences, we can say that, if the graph is well constructed, and regarding that the grammars generate grammatically correct sentences, our system holds both the property of grammaticality and meaninfulness. Nevertheless, the latter property can be seen as "weaker" than the others, because the user only provides seed terms, and not a fixed and well-formed meaning. As for poeticness, our system supports different configurations of poems and two of the implemented strategies take the number of syllables per line into consideration. Furthermore, the evolutionary approach has a bonus for rhymes. Therefore, according to Manurung [18], when following the generate-and-test or the evolutionary approach, our prototype can be seen as an actual poetry generation system.

6 CONCLUSIONS AND FURTHER WORK

We have presented PoeTryMe, a platform for the automatic generation of poetry, and a prototype, implemented in the top of this platform. One of the strengths of PoeTryMe is its high level of customisation. It may be used with different lexical resources and generate different poem configurations. The generation grammars may be edited and improved at will, in order to cover new linguistic constructions. Furthermore, new generation strategies may be implemented, which can originate different and interesting types of poems, according to a predefined purpose. Therefore, PoeTryMe can be used as the starting point for one (or more) poetry generation systems, eventually after taking future directions for improvement.

For instance, more generation strategies can be developed and the evolutionary strategy can be improved after testing more complex evaluation functions. Besides the number of syllables, other aspects, such as the the stress patterns, may also be considered. A strategy for generating rhymes more consistently, without a negative impact on the meter, should as well be devised.

In the implemented prototype, the lexical knowledge base used is structured on words. On the one hand, this might be a limitation, because natural language is ambiguous and several words have more than one sense. On the other hand, poetry is often vague and does not have to convey a precise message. Nevertheless, it would be interesting to compare the results of using word-based lexical resources against sense-aware resources (e.g. WordNet [7]) Also interesting would be to use a polarity lexicon (e.g. SentiWordNet [1]), in order to generate poetry with a predefined sentimental orientation (e.g. positive or negative), as others [4] have recently accomplished.

Although our prototype was created for Portuguese, the platform's architecture could be used for generating poetry in other languages. In order to do that, we would need to use external resources for the target language, including the lexical knowledge base, the syllabic division algorithm, and the morphology lexicon.

Finally, we should add that, as it happens for other creative artifacts, it is difficult to objectively evaluate the quality of a poem. Still, in the future, our results should be the target of some kind of validation and evaluation. Ideas for validation include comparing the configuration of the generated poems with similarly structured human-created poems, while evaluation might be performed based on the opinion of human subjects, which should consider aspects like the structure, meter, novelty and semantics of generated poems.

REFERENCES

[1] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani, 'SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining', in *Proceedings of the 7th International Conference on Language Resources and Evaluation*, LREC 2010, pp. 2200–2204, Valletta, Malta, (2010). ELRA.

[2] Anja Belz, 'Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models', *Natural Language Engineering*, **14**(4), 431–455, (October 2008).

[3] Selmer Bringsjord and David A. Ferrucci, *Artificial Intelligence and Literary Creativity: Inside the Mind of BRUTUS, a Storytelling Machine*, Lawrence Erlbaum Associates, Hillsdale, NJ., 1999.

[4] Simon Colton, Jacob Goodwin, and Tony Veale, 'Full FACE poetry generation', in *Proceedings of 3rd International Conference on Computational Creativity*, ICCC 2012, pp. 95–102, Dublin, Ireland, (2012).

[5] Belén Díaz-Agudo, Pablo Gervás, and Pedro A. González-Calero, 'Poetry generation in colibri', in *Proceedings of 6th European Conference on Advances in Case-Based Reasoning (ECCBR 2002)*, pp. 73–102, London, UK, (2002). Springer.

[6] Jay Earley, 'An efficient context-free parsing algorithm', *Communications of the ACM*, **6**(8), 451–455, (1970). Reprinted in Grosz et al. (1986).

[7] *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, ed., Christiane Fellbaum, The MIT Press, May 1998.

[8] P. Gervás, 'Exploring quantitative evaluations of the creativity of automatic poets', in *Workshop on Creative Systems, Approaches to Creativity in Artificial Intelligence and Cognitive Science, 15th European Conference on Artificial Intelligence*, (2002).

[9] Pablo Gervás, 'WASP: Evaluation of different strategies for the automatic generation of spanish verse', in *Proceedings of AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pp. 93–100, Birmingham, UK, (2000).

[10] Pablo Gervás, 'An expert system for the composition of formal spanish poetry', *Journal of Knowledge-Based Systems*, **14**, 200–1, (2001).

[11] Hugo Gonçalo Oliveira, Leticia Antón Pérez, Hernani Costa, and Paulo Gomes, 'Uma rede léxico-semântica de grandes dimensões para o português, extraída a partir de dicionários electrónicos', *Linguamática*, **3**(2), 23–38, (December 2011).

[12] Hugo Gonçalo Oliveira, F. Amílcar Cardoso, and Francisco C. Pereira, 'Exploring different strategies for the automatic generation of song lyrics with tra-la-lyrics', in *Proceedings of 13th Portuguese Conference on Artificial Intelligence*, EPIA 2007, pp. 57–68, Guimarães, Portugal, (2007). APPIA.

[13] Hugo Gonçalo Oliveira, F. Amílcar Cardoso, and Francisco Câmara Pereira, 'Tra-la-lyrics: an approach to generate text based on rhythm', in *Proceedings of 4th International Joint Workshop on Computational Creativity*, pp. 47–55, London, UK, (2007). IJWCC 2007.

[14] Marti A. Hearst, 'Automatic acquisition of hyponyms from large text corpora', in *Proceedings of 14th Conference on Computational Linguistics*, COLING'92, pp. 539–545. ACL Press, (1992).

[15] Sanghee Kim, Harith Alani, Wendy Hall, Paul H. Lewis, David E. Millard, Nigel R. Shadbolt, and Mark J. Weal, 'Artequakt: Generating tailored biographies with automatically annotated fragments from the web', in *Proceedings of ECAI 2002 Workshop Semantic Authoring, Annotation and Knowledge Markup*, SAAKM 2002, pp. 1–6, (2002).

[16] R. P. Levy, 'A computational model of poetic creativity with neural network as measure of adaptive fitness', in *Proceedings of the ICCBR-01 Workshop on Creative Systems*, (2001).

[17] Hisar Manurung, 'A chart generator for rhythm patterned text', in *Proceedings of 1st International Workshop on Literature in Cognition and Computer*, (1999).

[18] Hisar Manurung, *An evolutionary algorithm approach to poetry generation*, Ph.D. dissertation, University of Edinburgh, 2004.

[19] Ruli Manurung, Graeme Ritchie, and Henry Thompson, 'Using genetic algorithms to create meaningful poetic text', *Journal of Experimental & Theoretical Artificial Intelligence*, **24**(1), 43–64, (2012).

[20] Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad, 'Gaiku: generating haiku with word associations norms', in *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pp. 32–39. ACL Press, (2009).

[21] Ananth Ramakrishnan A and Sobha Lalitha Devi, 'An alternate approach towards meaningful lyric generation in tamil', in *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, CALC '10, pp. 31–39. ACL Press, (2010).

[22] Ananth Ramakrishnan A, Sankar Kuppan, and Sobha Lalitha Devi, 'Automatic generation of Tamil lyrics for melodies', in *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pp. 40–46, Stroudsburg, PA, USA, (2009). ACL Press.

[23] Ehud Reiter and Robert Dale, *Building natural language generation systems*, Cambridge University Press, New York, NY, USA, 2000.

[24] Graeme Ritchie, Ruli Manurung, Helen Pain, Annalu Waller, Rolf Black, and Dave O'Mara, 'A practical application of computational humour', in *Proceedings of 4th International Joint Workshop on Computational Creativity*, pp. 91–98, London, UK, (2007).

[25] Martin Tsan Wong and Andy Hon Wai Chun, 'Automatic haiku generation using VSM', in *Proceeding of 7th WSEAS Int. Conf. on Applied Computer & Applied Computational Science (ACACOS '08), Hangzhou, China*, (2008).