A REST Service for Poetry Generation

Hugo Gonçalo Oliveira

CISUC, Department of Informatics Engineering, University of Coimbra, Coimbra, Portugal
hroliv@dei.uc.pt

Abstract

This paper describes a REST API developed on the top of PoeTryMe, a poetry generation platform. This API exposes several functionalities, from the production of full poems, to narrower tasks, having in mind their utility for poetry composition, including the acquisition of well-formed lines, or semantically-related words, possibly constrained by the number of syllables, rhyme, or polarity. Examples that illustrate the endpoints and what they can be used for are also revealed.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases REST, creative web services, poetry generation, computational creativity

Digital Object Identifier 10.4230/OASIcs.SLATE.2017.12

1 Introduction

The topic of poetry generation, popular among the research community of Computational Creativity [3], is a kind of knowledge-intensive natural language generation that deals with several levels of language (e.g. lexical choice, syntax, semantics) as well as with formal features (e.g. metre and rhyme), towards the production of aesthetically-pleasing text, with a creative value. PoeTryMe [6, 7, 10] is one of many poetry generation systems reported in the literature (e.g. [4, 13, 15, 14, 19]), with the particularity of being focused on semantic relations and, more relevant for this work, having a modular architecture. The latter enabled its instantiation for producing poetry in different languages [10] and forms, including song lyrics [8], following different strategies [7], and from different stimuli, usually a list of seed words, but also Twitter trends [9] or concept maps extracted from text [11].

Though not designed with this specific purpose, PoeTryMe's architecture is friendly towards the vision of a Creative Web, where creative applications are deployed as web services [16], and in line with Gervás's [5] view on the deconstruction of poetry generation systems. Therefore, we decided to embrace this vision and develop a REST API for PoeTryMe, thus enabling the interaction of third-party applications, such as poetry-composition support tools, with this system.

This paper describes the state of this API, which currently enables the autonomous generation of full poems; the acquisition of well-formed lines, or semantically-related words, possibly constrained by the number of syllables, rhyme, or polarity; and to compute a score for the metre of a piece of text, according to a given poetry form. It starts with a brief reference to related work, followed by a contextualization of PoeTryMe and its architecture. After this, some details are presented about the API's implementation, followed by an enumeration of the available endpoints, together with some illustrative examples.

2 Related Work

Veale [16] describes the vision of creativity as set of web of services, a vision that promotes interoperability between different services that can be exploited and combined by different researchers for different purposes or for the creation of new applications, also saving development time, because the services are ready to use. Following this vision, several web services for metaphor generation were developed and used by different creative systems [17].

This vision is further materialised by creativity platforms that allow users to combine different creative or creativity-support services in the development of novel creative workflows that can be tested right away. These platforms include a wide range of tools, available as services, in a web interface that enables the selection and connection of services through drag-and-drop, setting the initial parameters and enabling the inspection of the state of each service. Produced workflows may include applications from different authors and can be easily shared with other users. ConCreTeFlows [18] and FloWr [2] are notable examples of such platforms, which have, among many other tasks, been used for poetry generation. The former includes a PoeTryMe service that uses one of the endpoints described in this paper for producing poems, given a small set of parameters. This has been used for generating poems inspired by conceptual blends. As for FloWr, it has been used for producing poetry by the recombination of lines extracted from Twitter, towards predefined poetic features [1].

Specifically speaking of poetry and web services, Gervás's [5] discusses the deconstruction of poetry generation systems and argues that abstractions of the various functionalities involved in such a system should be available as services that may be later invoked by other systems. Even if inspired by different stimuli or following different generation procedures, it makes sense that poetry generators share some of their modules. The most obvious would be to share the module for metric scansion, but other modules would be useful for different systems. PoeTryMe's REST API takes advantage of PoeTryMe's's modular architecture and embraces the aforementioned vision.

3 PoeTryMe's modular architecture

PoeTryMe has a modular architecture, explained with detail elsewhere [10], with several independent modules that might be combined for poetry generation. Those include two core modules – a Generation Strategy and a Lines Generator – and some complementary ones.

The Generation Strategy implements a plan for producing poems according to user-given parameters. It can have different implementations and interact with the Syllable Utils to perform syllable-related operations, such as syllable division or rhyme identification.

The Lines Generator interacts with the Relations Manager, Morphology Handler and Grammar Processor for producing semantically-coherent fragments of text, to be used as lines of a poem. The Relations Manager and the Grammar Processor are interfaces to a semantic network and to a context-free grammar, respectively. The former may cover any kind of labelled relation, and the latter has rules for rendering relations as text, given the relation label. The Lines Generator may also resort to a Contextualizer for explaining the selection of words and lines through lists of relations and the connection of their arguments to the initial parameters.

There is also a module for expanding a set of provided words with structurally-relevant words, possibly constrained by a target polarity (positive or negative). Poetry can be generated in Portuguese, Spanish or English, depending on the underlying linguistic resources, namely the semantic network, the lexicons and the grammars.

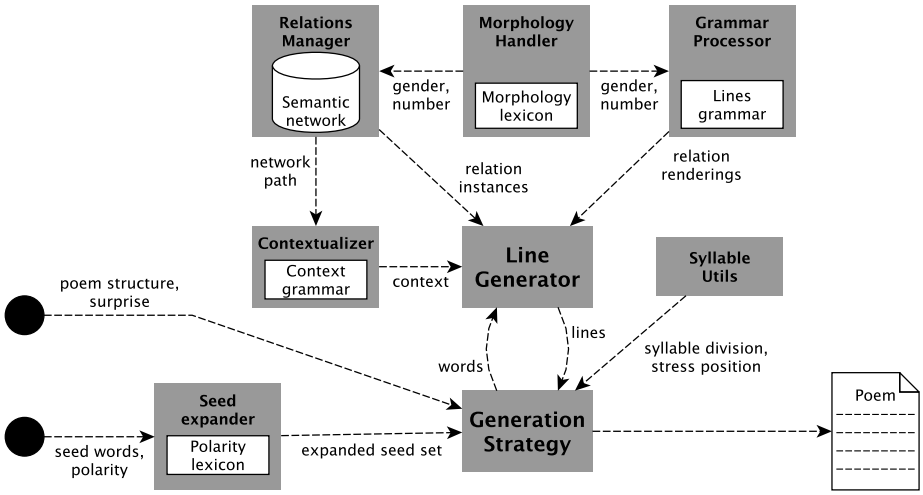

Figure 1 PoeTryMe's modular architecture.

Figure 1 depicts PoeTryMe's architecture with dashed lines representing the flow involved in the autonomous generation of a poem. Initially, the user provides a set of parameters, including the structure of the poem, the set of seed words, the target polarity and a surprise factor. The structure file indicates the number of lines, syllables per line and rhyme scheme to follow. The seeds constrain the Relations Manager to use only these words, directly-related and some indirectly-related words, depending on the surprise. A higher surprise factor will increase the number of indirectly-related words. The set of seeds can be further augmented with the Seed Expander.

In addition to the presented flow, this modular architecture enables different combinations of the available modules, which work independently of each other. In other words, different poetry generation systems can arise from a different combination of PoeTryMe's modules, or existing computational applications can be augmented by exploiting only one or two modules. This was our main motivation to expose the access to some of these modules, through a REST API. Details about this API, with a focus on the available endpoints, are described in the following section.

4 API: implementation and endpoints

PoeTryMe is implemented in Java, so it made sense to implement its REST API also in this language. To ease the process, the Jersey RESTful Web Services framework[1], an open-source, standard and portable JAX-RS API, was used. The service runs in a Tomcat server, installed in `http://poetryme.dei.uc.pt:8080`, and has several HTTP endpoints, all returning HTTP responses with JSON objects. This section describes the endpoints defined for this service, together with usage examples. Endpoints were created with PoeTryMe's architecture in mind, as well as their utility for poetry composition support tools.

Full poems can be generated from the following endpoint:

- `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry`

[1] Check `https://jersey.java.net/`.

- **Request:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry?lang=en&form=10-4&seeds=computer+language&surp=0.002`
- **Response:**


```
{"form":"quadra_10.est","language":"en","score":"-2.0",
"seeds":["computer","language"],"surprise":"0.002",
"text":"computer slots have many other names\n
  the focus accents swaying in the wind\n
  the packs of heaven content over me\n
  resurrect my vocal through potpourri\n\n"}
```


Figure 2 Using the API for the generation of a full poem.

Internally, this endpoint triggers the workflow of Figure 1, without a target polarity or expansion, because the latter could overload the server. It thus supports the following set of parameters:

- Language: `lang=[en|pt|es]`
- Form: `form=[id of the form]` (currently limited to a pre-defined list that includes, e.g. `10-2` for 10-syllable couplets, `10-4` for 10-syllable blocks-of-four, or `sonnet` for sonnets, and also some children and pop songs.)
- Seeds: `words=[comma-separated list of words]`
- Surprise: `surp=[0-1]`

Figure 2 illustrates how the previous endpoint can be used for generating a block of four lines in English, using the seeds *computer* and *language*, with a surprise of 0.002. The response is a JSON object with the properties form (internal file with that represents the target form), language (the language of the poem), score (an internal score based on the metre), seeds (an array with the seed words), surprise (the surprise factor), and text (the resulting poem).

Besides the previous, there are currently three other endpoints: one focused on single lines; another on words; and one for scoring the metre of a piece of text, according to a target poetry form. The following endpoint can be used for generating a single line, given a list of seeds and a surprise factor, selected from the best n generated lines, according to the target number of syllables:

- `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/line`
  - Language: `lang=[en|pt|es]`
  - Seeds: `seeds=[comma-separated list of words]`
  - Surprise: `surp=[0-1]`
  - Target number of syllables: `nsyl=[1-n]`
  - Generations: `bestof=[1-n]`

Figure 3 illustrates this endpoint with an example in Portuguese and another in Spanish.

Words related to a target word can be retrieved with the following endpoint. The relation can be semantic, same number of syllables, same rhyme, or a combination of the previous. Resulting words might be further constrained with a target polarity[2]:

- `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/words`
  - Language: `lang=[en|pt|es]`
  - Word: `word=[word]`

[2] Check Gonçalo Oliveira et al. [10] for a reference of the current underlying resources.

- **Request 1:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/line?lang=pt&nsyl=10&seeds=criatividade&bestof=10`
- **Response 1:**
  `{"form":"1-liner","language":"pt","score":"3.0","seeds":"criatividade", "surprise":"0.0","text":"aptidões mortas , criatividade escura"}`
- **Request 2:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/line?lang=es&nsyl=7&seeds=crear&bestof=100`
- **Response 2:**
  `{"form":"1-liner","language":"es","score":"0.0","seeds":"crear", "surprise":"0.0", "text":"mi dar no quiere crear"}`

Figure 3 Using the API for the generation of single lines.

- Relation: `rel=[id for relation type]` (supported types are: `synon` for synonymy, `anton` for antonymy, `hyper` for hypernym, `hypon` for hyponymy, `cohyp` for co-hyponymy, `other` for any other type, or `any` for any type)
- Rhyme: `syl=[word with the target rhyme]`
- Target number of syllables: `syl=[word with the target number of syllables]`
- Polarity: `pol=[-1,0,1]`

Figure 4 illustrates this endpoint with four examples: the first retrieves English words that rhyme with *state* and have a negative polarity; the second retrieves English words related to *creativity* that rhyme with *nation*; the third example retrieves Portuguese words that are hyponyms of *pessoa* and rhyme with *gerar*; the final example retrieves Portuguese words that are synonyms of *plano* and have the same number of syllables as *amigo*.

The remaining endpoint scores the metre of a piece of text according to a poetry form:

- `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/score`
  - Language: `lang=[en|pt|es]`
  - Form: `form=[id of the form]` (currently limited to a small set of forms)
  - Text: `text=[full text]` (lines split with a '`\n`')

The score is computed the same way as for the automatically-generated poems. More precisely, score is the sum of the absolute difference between the target number of syllables each line should have and the actual number. On the top of this, a bonus of -2 is given for every pair of rhyming lines, meaning that, the lower the score, the better the metre is matched. Figure 5 illustrates this endpoint when scoring text as 10-syllable couplets. In the three examples, couplets are given, yet, in the first, both lines rhyme; in the second they do not, but the number of syllables is still matched; while in the third the second line has only eight syllables.

5 Concluding remarks

A REST service was presented for autonomous poetry generation, with additional features that might be useful for poetry composition support tools. It is built on the top of PoeTryMe, a poetry generation platform with a modular architecture, and besides the autonomous generation of full poems, it enables the acquisition of well-formed lines, or semantically-related words, possibly constrained by the number of syllables, rhyme, or polarity, and the computation of a score for the metre of a piece text, according to a target poetry form.

- **Request 1:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/words/?lang=en&rhyme=state&pol=-1`
- **Response 1:**

```
{"language":"en","polarity":"-1","relation":"N/A",
"rhymesWith":"state","word":"state",
"related":["desecrate","bait","irate","negate","grate","hate",
  "berate","overweight"]}
```

- **Request 2:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/words/?lang=en&word=creativity&rel=any&rhyme=nation`
- **Response 2:**

```
{"language":"en","polarity":"N/A","relation":"any",
"rhymesWith":"nation","word":"creativity",
"related":["innovation","imagination"]}
```

- **Request 3:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/words/?lang=pt&word=pessoa&rhyme=gerar&rel=hypon`
- **Response 3:**

```
{"language":"pt","polarity":"N/A","relation":"hypon",
"rhymesWith":"gerar","word":"pessoa",
"related":["par","secular","militar","familiar","titular"]}
```

- **Request 4:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/words/?lang=pt&word=plano&rel=synon&syl=amigo`
- **Response 4:**

```
{"language":"pt","polarity":"N/A","relation":"synon",
"syllablesMatch":"amigo","word":"plano",
"related":["prospecto","traçado","intento","sistema","esquema",
  "macio","programa","direito","desenho","projecto"]}
```

Figure 4 Using the API for the retrieval of words.

This API is still in development and, in the future, we should deal with minor internal issues and will consider the inclusion of additional functionalities, such as the contextualization of given lines, or the production and scoring of poems according to any given metre, not limited to the set of covered forms. For the latter purpose, we would use the internal representation for poetry forms, which includes the number of lines of a poem, their metre, and, optionally, a rhyming scheme. Yet, this representation is not URL friendly, so a simpler but equivalent format will have to be designed.

The poetry generation module of this API is currently integrated in ConCreTeFlows [18], and may thus be included in any workflow of this platform. Furthermore, Co-PoeTryMe[3], a web application built on the top of this API is currently being developed. Co-PoeTryMe can be seen as a co-creative application, like others of such kind targeting poetry generation [12]. It enables the collaboration between the human creator and PoeTryMe in the composition of poetry, hopefully better than poems autonomously produced by PoeTryMe, or more in line with the user intents. In the future, it should also be used by the Twitter bot *@poetartificial*, which is currently based on an independent instantiation of PoeTryMe.

[3] Check `http://poetryme.dei.uc.pt/~copoetryme/`.

- **Request 1:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/score?lang=en&form=10-2&text=Be%20not%20the%20first%20by%20whom%20the%20new%20are%20tried%2C%0ANor%20yet%20the%20last%20to%20lay%20the%20old%20aside.`
- **Response 1:**

```
{"form":"10-2","language":"en","score":"-2.0",
"text":"Be not the first by whom the new are tried,\n
  Nor yet the last to lay the old aside."}
```

- **Request 2:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/score?lang=en&form=10-2&text=Be%20not%20the%20first%20by%20whom%20the%20new%20are%20tried%2C%0ANor%20yet%20the%20last%20to%20lay%20the%20old%20aside.`
- **Response 2:**

```
{"form":"10-2","language":"en","score":"0.0",
"text":"Be not the first by whom the new are tried,\n
  Nor yet the last to lay the old behind."}
```

- **Request 3:**
  `http://poetryme.dei.uc.pt:8080/PoetrymeWeb/rest/poetry/score?lang=en&form=10-2&text=Be%20not%20the%20first%20by%20whom%20the%20new%20are%20tried%2C%0AThis%20line%20has%20eight%20metrical%20syllables`
- **Response 3:**

```
{"form":"10-2","language":"en","score":"2.0",
"text":"Be not the first by whom the new are tried,\n
  This line has eight metrical syllables"}
```

Figure 5 Using the API for scoring the metre of couplets.

References

1 John Charnley, Simon Colton, and Maria Teresa Llano. The FloWr framework: Automated flowchart construction, optimisation and alteration for creative systems. In *International Conference on Computational Creativity (ICCC)*, pages 315–323, June 2014.

2 John Charnley, Simon Colton, Maria Teresa Llano, and Joseph Corneli. The FloWr online platform: Automated programming and computational creativity as a service. In *7th International Conference on Computational Creativity (ICCC)*, pages 363–370, 2016.

3 Simon Colton and Geraint A. Wiggins. Computational creativity: The final frontier? In *20th European Conference on Artificial Intelligence (ECAI)*, pages 21–26, 2012.

4 Pablo Gervás. WASP: Evaluation of different strategies for the automatic generation of spanish verse. In *AISB'00 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 93–100, 2000.

5 Pablo Gervás. Deconstructing computer poets: Making selected processes available as services. *Computational Intelligence*, 33(1):3–31, 2017.

6 Hugo Gonçalo Oliveira. PoeTryMe: a versatile platform for poetry generation. In *Proceedings of the ECAI 2012 Workshop on Computational Creativity, Concept Invention, and General Intelligence*, C3GI 2012, Montpellier, France, August 2012.

7 Hugo Gonçalo Oliveira and Amílcar Cardoso. Poetry generation with PoeTryMe. In T. R. Besold, M. Schorlemmer, and A. Smaill, editors, *Computational Creativity Research: Towards Creative Machines*, chapter 12, pages 243–266. Atlantis-Springer, 2015.

8 Hugo Gonçalo Oliveira. Tra-la-Lyrics 2.0: Automatic generation of song lyrics on a semantic domain. *Journal of Artificial General Intelligence*, 6(1):87–110, 2015. Special Issue: Computational Creativity, Concept Invention, and General Intelligence.

9 Hugo Gonçalo Oliveira. Automatic generation of poetry inspired by Twitter trends. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 631 of *CCIS*, pages 13–27. Springer, 2016.

10 Hugo Gonçalo Oliveira, Raquel Hervás, Alberto Díaz, and Pablo Gervás. *Multilanguage Extension and Evaluation of a Poetry Generator*, 2017. (in press).

11 Hugo Gonçalo Oliveira and Ana Oliveira Alves. Poetry from concept maps: Yet another adaptation of PoeTryMe's flexible architecture. In *7th International Conference on Computational Creativity*, 2016.

12 Anna Kantosalo, Jukka M. Toivanen, Ping Xiao, and Hannu Toivonen. From isolation to involvement: Adapting machine creativity software to support human-computer co-creation. In *5th International Conference on Computational Creativity*, 2014.

13 H. M. Manurung. *An evolutionary algorithm approach to poetry generation.* PhD thesis, University of Edimburgh, Edimburgh, UK, 2003.

14 Joanna Misztal and Bipin Indurkhya. Poetry generation system with an emotional personality. In *5th International Conference on Computational Creativity (ICCC)*, 2014.

15 Jukka M. Toivanen, Matti Järvisalo, and Hannu Toivonen. Harnessing constraint programming for poetry composition. In *4th International Conference on Computational Creativity (ICCC)*, pages 160–167, 2013.

16 Tony Veale. Creativity as a web service: A vision of human and computer creativity in the web era. In *Creativity and (Early) Cognitive Development: A Perspective from Artificial Creativity, Developmental AI, and Robotics*, pages 73–78. AAAI Press, 2013.

17 Tony Veale. A service-oriented architecture for metaphor processing. In *Second Workshop on Metaphor in NLP*, pages 52–60. ACL Press, 2014.

18 Martin Žnidaršič, Amílcar Cardoso, Pablo Gervás, Pedro Martins, Raquel Hervás, Ana Oliveira Alves, Hugo Gonçalo Oliveira, Ping Xiao, Simo Linkola, Hannu Toivonen, Janez Kranjc, and Nada Lavrac. Computational creativity infrastructure for online software composition: A conceptual blending use case. In *7th International Conference on Computational Creativity (ICCC)*, Paris, France, 2016.

19 Rui Yan. i, Poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2238–2244, 2016. URL: `http://www.ijcai.org/Abstract/16/319`.