Poetry from Concept Maps – Yet Another Adaptation of PoeTryMe's Flexible Architecture

Hugo Gonçalo Oliveira[1], Ana Oliveira Alves[2]
[1] CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
[2] CISUC, Polytechnic Institute of Coimbra, Portugal
hroliv@dei.uc.pt, ana@dei.uc.pt

Abstract

This paper describes a preliminary effort on adapting an existing poetry generation platform, PoeTryMe, to produce poetry from concept maps, extracted from textual documents. Instead of a set of given words that would constrain a general language semantic network, the presented adaptation dynamically sets the semantic network to the given concept map. As the relations in the concept maps are open, a new generation grammar had also to be created. Besides an architectural overview of the system, this paper is illustrated with several generated poems, together with the maps that originated them. Still, although poetic features are present and the content of the maps is reflected, they do not transmit exactly the meaning of the original document, due both to limitations on the grammars and issues on the quality of the maps.

Introduction

Computational approaches to linguistic creativity include the generation of narratives (Gervás et al., 2005), verbally-expressed humor (Binsted and Ritchie, 1994), or poetry, among others. In the last years, we have seen the birth of a diversity of poetry generation systems and approaches, driven by the advances on natural language processing and generation tools and on the huge amounts of textual data currently available.

Poetic text is typically recognised by the usage of certain features, such as a regular metre, rhymes or the presence of figurative language. To achieve such results, which should additionally be interpreted under a certain domain, current systems deal with several sources of knowledge, including natural language processing tools, lexicons, semantic knowledge-bases, or data extracted from human-created poems, among others. Available knowledge is exploited by a variety of approaches, frequently driven by some stimuli, which can be in form of a set of seed words, a short phrase, or a textual document. Although a minority of the efforts reported in the literature have some architectural concerns and could potentially be adapted to different situations, most of them are tailored for a specific purpose and end up having a reduced scope.

On the other hand, PoeTryMe (Gonçalo Oliveira and Cardoso, 2015) is a generic platform for poetry generation, with a modular architecture that enables different instantiations and its reimplementation as poetry generation systems with different purposes. The flexibility of PoeTryMe's architecture is confirmed by its adaptation to generate poetry in different languages (Gonçalo Oliveira et al., 2014), poetry inspired by Twitter trends (Gonçalo Oliveira, 2016), or song lyrics (Gonçalo Oliveira, 2015). This paper reports on yet another effort to use PoeTryMe's architecture, this time to produce poetry from the content of prose documents. For this purpose, PoeTryMe uses the output of TextStorm (Oliveira, Pereira, and Cardoso, 2001), an Open Information Extraction tool for acquiring concept maps automatically from textual documents. In opposition to other adaptations of PoeTryMe, which involved varying the poem structures, using different line templates, or plugging in different language resources, this new adaptation is not based on a set of seed words and involves changing the underlying semantics for the poem dynamically. Instead of a general language semantic network, current generation relies only on the semantic predicates extracted by TextStorm for the given document.

We begin with a reference to related work, namely poetry generation systems, focused on those efforts that reuse an existing model or architecture and those that produce poetry based on a prose document. After that, PoeTryMe and its architecture are briefly addressed, which is followed by a short description of TextStorm, the system used for extracting concept maps from text. The effort involved in the integration of the previous systems, including the acquisition of line templates for the TextStorm predicates, is then described. Before concluding, illustrative examples of poems, their seed maps and some of the original documents are presented and their quality is briefly discussed. While this work confirms the flexibility of PoeTryMe's architecture, we are not fully satisfied with the achievements regarding the generation of poetry from a textual document. We see the reported work as a preliminary step to achieve this goal in the future, following some of the lines remarked in the last section.

Related Work

A variety of paradigms has been applied to automatic production of text with poetic features, including case-based reasoning (Gervás, 2001), evolutionary algorithms (Manurung, Ritchie, and Thompson, 2012), constraint programming (Toivanen, Järvisalo, and Toivonen, 2013), or multi-agent systems (Misztal and Indurkhya, 2014). Several po-

etry generation systems are based on poem or line templates, but most of them go further and combine the previous with other techniques (e.g. Colton, Goodwin, and Veale (2012); Toivanen, Järvisalo, and Toivonen (2013)). Produced word sequences usually evolve to meet the desired constraints at different levels, such as form (lines, stress pattern), rhymes, syntax, and semantics. On the semantic level, the choice of relevant words may be achieved either with the help of semantic knowledge bases (Agirrezabal et al., 2013), by exploring models of word associations extracted from corpora (Netzer et al., 2009; Toivanen, Järvisalo, and Toivonen, 2013), or both (Colton, Goodwin, and Veale, 2012). Generation can be driven by a given stimuli, which can be in the form of a prose message (Gervás, 2001), a theme (Toivanen, Järvisalo, and Toivonen, 2013), or a set of seed words (Netzer et al., 2009), which constrain the poem search space and set the semantic domain.

More recently, systems that produce poetry based on a textual document have also been presented. Colton, Goodwin, and Veale (2012) analyse news articles to set the mood of the day and then select an article and a poem template to produce a new poem. Replacement words are selected through a rich process, based on a collection of similes, that considers features like aesthetics, lyricism, sentiment and flamboyancy. Misztal and Indurkhya (2014)'s system is inspired by an input text, analysed to set the theme and the mood of the poem. Then, a set of artificial experts suggest related words, organise them into phrases, possibly exploring figurative language, and select the best phrases based on form constraints. Toivanen, Gross, and Toivonen (2014) produce poems inspired by news stories, from which novel word associations are extracted, when compared to long-established associations, such as those in Wikipedia. The resulting poem is obtained by replacing content words in an existing poem with the acquired associations. Tobing and Manurung (2015) rely on a dependency parser to extract the predicate-argument structure of a document, which it tries to keep during the generation of a poem, where additional constraints on form are considered. Generated poems explicitly capture the meaning of the input document but, from a computational point of view, dealing with these together with the poetic constraints seems to be impractical.

Whether or not they are reusable or adaptable to other situations, the previous systems have been presented as a specific instantiation, with a specific workflow, or even tested for producing a specific kind of poetry. An exception is the architecture of Colton, Goodwin, and Veale (2012) and the constraint satisfaction approach of Toivanen, Järvisalo, and Toivonen (2013), which happen to be combined in Rashel and Manurung (2014)'s system, even though the latter targets a different language.

Gervás (2015) argues that abstractions of the various functionalities involved in a poetry generation system should be available as services that may be invoked by other systems. This would allow the development of different systems that would, nevertheless, share some of their modules. Among other benefits, this would ease the development process and ease, for instance, the evaluation or the impact of adding new components. In ConCreTeFlows (Žnidaršič et al., 2016), several widgets, including PoeTryMe and TextStorm, are available as independent processes, which can (and have) been manipulated to create novel and creative workflows. A similar platform is FloWr (Charnley, Colton, and Llano, 2014) which, among others, has been used to build poetry generation systems.

PoeTryMe and its Flexible Architecture

PoeTryMe (Gonçalo Oliveira and Cardoso, 2015) is a poetry generation platform developed since 2009 at CISUC, University of Coimbra, Portugal. It relies on a modular architecture (see figure 1 further ahead) that enables the independent development of each module and provides a high level of customisation, depending on the needs of the system and ideas of the user or developer. Among other parameters, users may define the structure of the poem, the transmitted sentiment, the generation strategy, the semantic network to use and the rules for generating lines based on the available relations. Developers may reimplement some of the modules and reuse the others.

A Generation Strategy organises lines, such that they suit, as much as possible, the structure of a poetic form and exhibit certain features. A structure file sets the poem form with the number of stanzas, lines per stanza and of syllables per line. An instantiation of the Generation Strategy does not generate the lines, but exploits the Lines Generator module to retrieve natural language frements, which might be used as such. Syllable-related features are assessed with the help of the Syllables Util. Given a word, this module may be used to divide it into syllables, to find its stress, or to extract its termination, useful to identify rhymes.

The Lines Generator produces natural language renderings of semantic relations with the help of: (i) a semantic network, managed by the Relations Manager, that connects words according to relation predicates; and (ii) a generation grammar, processed by the Grammar Processor, with textual templates that render fragments expressing semantic relations. The generation of a line is a three-step interaction:

1. A random relation instance, in the form of a $triplet = (word_1, predicate, word_2)$, is retrieved from the semantic network. To constrain the space of possible generations, a set of seed words can be provided to the Relations Manager. This set defines the generation domain, represented by a subgraph of the main network that will contain all the triplets involving seed words, or indirectly connected, depending on a surprise factor.
2. A random rendering for the $triplet$'s predicate is retrieved from the grammar. There must be a direct mapping between the relation predicates, in the graph, and the rules' name, in the grammar. Besides terminal tokens, that will be present in the poem without change, rules have placeholders that indicate the position of the relation arguments (`<arg1>` and `<arg2>`).
3. The resulting rendering is returned after inserting the arguments of the $triplet$ in specific placeholders of a rule for its predicate.

In addition to the previous modules, the Contextualizer explains why certain words were selected and what is their

connection to the seed words, as a list of triplets for each line. It can be used for debugging or evaluation purposes.

The flexibility of PoeTryMe's architecture is confirmed by its adaptation to generate poetry in different forms, including song lyrics that suit a given rhythm (Gonçalo Oliveira, 2015), poetry inspired by Twitter trends (Gonçalo Oliveira, 2016) and, although originally developed for Portuguese, PoeTryMe has been adapted to other languages, including Spanish (Gonçalo Oliveira et al., 2014) and, more recently, English.

TextStorm

TextStorm (Oliveira, Pereira, and Cardoso, 2001) is an Open Information Extraction tool also developed at CISUC, starting in 2001, with the original aim of creating concept maps for Clouds, an inductive learning tool, which would then be used as input to the Divago concept blending system Pereira (2007). Its original aim was to iteratively create concept maps from the Clouds, which could then be used as input to the Divago concept blending system Pereira (2007). TextStorm extracts conceptual relations from a textual document, written in natural language. Based on a Definite Clause Grammar, TextStorm extracts binary predicates from a text file, using syntactic and discourse knowledge, without requiring any previous knowledge on the document's domain. The resulting set of predicates – represented in a common Prolog notation, *functor(Argument 1, Argument 2)* – constitute a graph where nodes are concepts and edges are relations between them, a knowledge representation format also known as Concept Maps (Novak, 1998).

TextStorm tags the input text file using WordNet (Fellbaum, 1998), and then builds predicates that map relations between two concepts, based on the parsed sentences. For instance, utterances like "*Cows, as well as rabbits, eat only vegetables, while humans eat also meat*" would result in the following predicates, that form a concept map:

- eat(*cow*, *vegetables*)
- eat(*rabbit*, *vegetables*)
- eat(*human*, *vegetables*)
- eat(*human*, *meat*)

Since, in natural language text, the same concept is not always referred by the same word or expression, TextStorm resorts to synonymy relations in WordNet to identify the concepts previously mentioned, even if through a different expression.

Poetry based on Concept Maps

This section reports on the effort of integrating the systems described earlier in a new instantiation of PoeTryMe, for producing poetry based on concept maps, extracted by TextStorm from a given textual document. The resulting architecture, with the relevant PoeTryMe modules, is depicted in figure 1.

The key of the present instantiation involved changing the semantic network dynamically, but ended up requiring the creation of a new grammar. No low-level changes were needed to the original Relations Manager and Grammar Processor modules.

All of the previous instantiations of PoeTryMe relied in a general language lexical-semantic knowledge base, where a subnetwork was selected, based on the given seeds. Although it could be quite large, the knowledge base was closed, both in terms of covered lexical items and relations. This meant that the poems would only be able to use seeds covered by the knowledge base, and related words, which had to be related by one of the covered predicates (e.g. synonymy, antonymy, hypernymy and meronymy). Bu TextStorm extracts an open set of predicates, most of them not covered by our previous grammars.

In our integration solution, a new grammar had to be created, in order to cover as many TextStorm predicates as possible. For that purpose, a large set of concept maps had to be extracted, hopefully covering a varied set of predicates. Those maps were then used as the knowledge base for the acquisition of the new grammar. Although they can be handcrafted, previous grammars of PoeTryMe have been acquired automatically from given textual lines (e.g of human-created poems) where two related words occur, then generalised as possible renderings of their relation. For instance, for a semantic network with the relations:

- *abstraction* hypernym-of *poem*
- *flower* hypernym-of *dahlia*
- *animal* hypernym-of *cat*

Lines such as:

- "*the poem itself is a kind of abstraction*",
- "*abstraction, in the form of a poem*",

Could be generalised to originate the following rules:

- *hypernym-of* $\rightarrow$ *the <arg2> itself is a kind of* $< arg1 >$
- *hypernym-of* $\rightarrow$ *<arg1> in the form of a* $< arg2 >$

During the generation of a poem, this rule may result in variations, such as:

- *the dahlia itself is a kind of flower*
- *animal, in the form of cat*

In order to obtain large quantities of text, written as directly as possible, we resorted to a selection of articles from the Simple Wikipedia[1], then processed by TextStorm. The selection of this resource relied on its wide-coverage of different topics, described using basic English vocabulary and shorter sentences, which would hopefully improve the extraction of concept maps. The grammars were not extracted from the Simple Wikipedia though. The extracted concept maps were only used as a knowledge base for the later acquisition of relation renderings from any other text collection. Figure 2 illustrates the grammar extraction procedure.

The new instantiation of PoeTryMe works as follows:

1. A textual document is provided to TextStorm, which extracts a concept map;
2. The resulting concept map is converted to PoeTryMe's notation – which instead of *predicate(arg1, arg2)* becomes *arg1 PREDICATE arg2*;

[1] `https://simple.wikipedia.org`

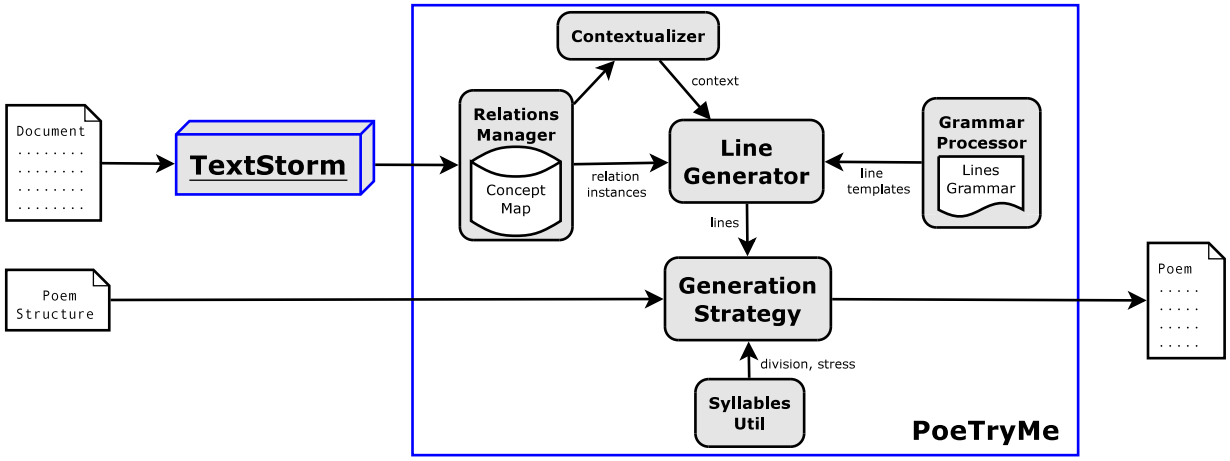


Figure 1: High level diagram of the resulting architecture (TextStorm + PoeTryMe)

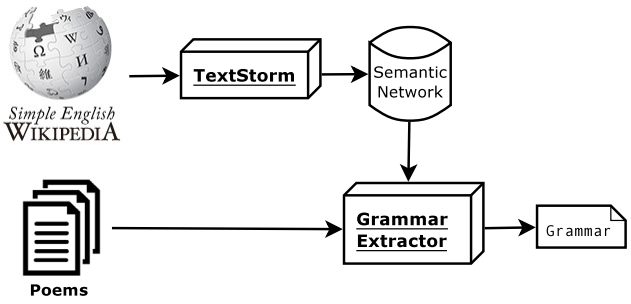


Figure 2: Grammar extraction procedure

3. PoeTryMe is run for a given poetry form, using the concept map as input, instead of a set of seed words;
   (a) The concept map is dynamically set as the semantic network to use;
   (b) Lines are produced by the Line Generator, using all the relations of the concept map;
   (c) Most suitable lines are selected according to a generate & test strategy, the same of previous instantiations of PoeTryMe.
4. The output is a poem in the target form, where each line is a rendering of a relation instance in the concept map.

Moreover, since TextStorm only supports English, this work relied on a previous adaptation of PoeTryMe to English, following similar lines as the Spanish adaptation (Gonçalo Oliveira et al., 2014). A major difference of the current adaptation is related to metric scansion, which is more complex for English than for Portuguese and Spanish. While, for the latter, most cases were covered with a rule-based approach, relying only on the orthography, for English, this would not work, because there are many different combinations of letters that are pronounced the same way (e.g. *eye* rhymes with *lie*, *apply* and *levi*; *air* rhymes with *aware* and *bear*). Therefore, in order to perform syllable division, stress and rhyme identification, we relied on the CMU Pronouncing Dictionary[2], which contains over 134,000 words and their pronunciations in North American English. A new implementation of the Syllable Utils interface was developed to interact with this dictionary and perform the syllable-related operations on English words. For non-covered words, a fallback mechanism uses the Portuguese rules. In order to acquire the rules of the lines grammar, a collection of human created English poems was exploited. Those were obtained from the Representative Poetry Online (RPO), a web anthology of poetry by the University of Toronto Libraries[3].

The rules of the new grammar were thus obtained from the lines of the previous poems where two related words co-occurred, according to the concept maps extracted from the Simple Wikipedia. Each of those lines had the related words replaced by the argument placeholders and resulted in a rule for the relation predicate. Using the concept maps obtained from 4,096 Simple Wikipedia articles, and about 3,400 human-created poems, the grammar used in this work had 2,289 rules and covered a total of 171 distinct predicates.

Results

To illustrate the results of the presented integration, several poems were produced. As mentioned earlier, instead of seed words, the full concept map was given as input and used as PoeTryMe's semantic network. The generation of lines followed a generate & test strategy at the line level, similar to that of previous instantiations (e.g. Gonçalo Oliveira et al. (2014)). For each line, up to $n = 2,000$ textual renderings of relations in the concept map were sequentially produced and tested against the target size and rhyme, while keeping the best one. To increase the probability of rhymes, an increasing factor $\sigma = 0.8$ was used, meaning that the number of renderings produced for the i^{th} line of a stanza were at most $n + n * (i - 1)$. For $n = 2,000$, this results in $2,000$

[2] http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/

[3] http://rpo.library.utoronto.ca/timeline/

renderings for the first line of a stanza, $3,600$ for the second, $5,200$ for the third and $6,800$ for the fourth. In order to select the best lines, each syllable deviating from the target number lead to a penalty of 1 point, while each rhyme resulted in a bonus of 2 points.

Examples

Figures 3 to 6 show (partial) concept maps on diverse subjects and selected generated poems. In addition to the map, figures 3 and 6 show part of the textual document that originated it, omitted in the remaining examples, due to lack of space. All but the example in figure 6 are based on Simple Wikipedia articles, which was our original source for extracting the generation grammars. The poems in figure 3 are blocks of four 10-syllable lines, based on the article on Artificial Intelligence. The following two examples are based on articles about named entities, more precisely, one shows blocks of four 10-syllable lines based on the city of Paris (figure 4), and the other a sonnet based on Alan Turing's article (figure 5).

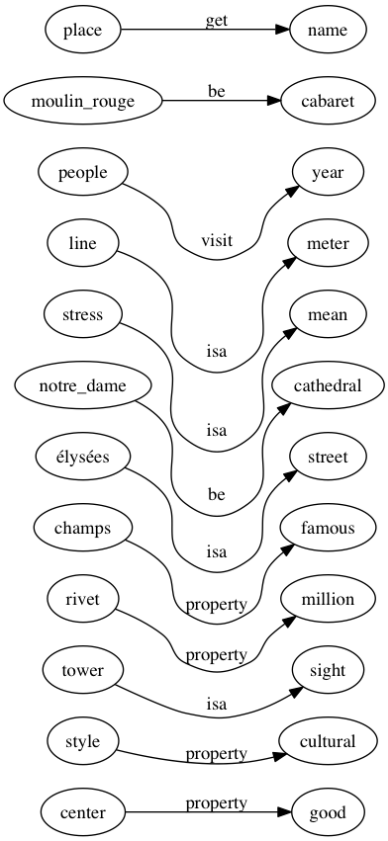
The final example, in figure 6, is an attempt to go one step further and confirm that the integration of both systems enables the generation of poems from any given textual document. Its concept map is extracted from a news article, published in *The Atlantic* online newspaper on 2^{nd} March 2016[4].

Discussion

Similarly to those selected, generated poems frequently match the target size of a line and, when the size is different, the difference is rarely more than 1. We recall that, in the scoring system used, the bonus for rhymes is two times higher than the penalty for one additional syllable or one less. Rhymes are also frequent. Although we manually selected poems with rhymes in almost all the lines, if the semantic network and the generation grammar are large and varied enough, generating poems with many rhymes is just a matter of increasing n and σ. One issue regarding the form of the poems is the presence of a few syntactically-odd lines. This is partially due to our grammar acquisition procedure, where the part-of-speech (POS) of the arguments is not considered. As we know, most verbs can also be nouns (e.g. *break*, *cover*), and many nouns can behave as adjectives (e.g. *red*, *young*). In fact, TextStorm also extracts this kind of information for terms in WordNet, but it was not exploited in the current instantiation.

Despite the previous issue, we can say that the concept map is reflected on the produced poems. A minority of deviations occur due to the presence of fixed words in the template, especially open class words, which may sometimes be out of the desired context. This is a limitation of PoeTryMe, which could be minimised by handling the previous issue regarding the POS, creating rules with no more than two content words (to be replaced), or using more general sources of language as the source of the grammars. But if those are not

[4] http://www.theatlantic.com/politics/archive/2016/03/donald-trump-the-protector/471837/



why ask my tower? that old sight will swear
a name of weight; line little meter heir
thus the great people of almighty year
and élysées, and street shall disappear

moulin rouge and mother, cabaret bless
his stress, while the mean spirit's plastic stress
leave me to my cathedral notre dame
though from another place i take my name

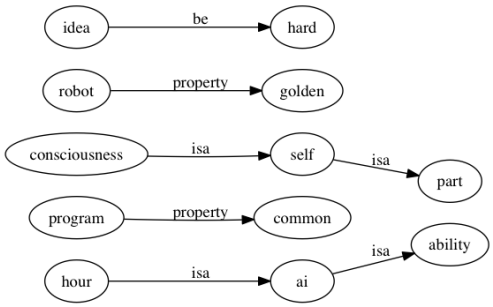
famous of champs, and the better part choose
that which a good center only could refuse
the million sort by thir own rivet fell
once cultural, now in style, and to dwell

Figure 4: Concept map of the Simple Wikipedia article on Paris and generated blocks of four 10-syllable lines.

poems, this may have a negative impact on the poeticness of the generated text.

A bigger problem is the quantity and quality of the maps, which results in poems that cannot be said to be about the original documents. Some relevant relations are not captured, and some others are not exactly what should have been extracted. This happens because TextStorm does not handle elaborate sentences very well, especially those with implicit co-references. In fact, it does not perform two important natural language processing tasks: anaphora resolution and named entity recognition. Not resolving anaphoras results in some relations held between pronouns (e.g. *he,*

Artificial intelligence **(AI) is the ability** of a computer to think like a human (or eventually better) – to be able to learn and to have "new ideas". ... For example, a **common computer program** can turn a report of names and **hours** worked into paychecks for the workers at a company. ... That is the difference between a program and **AI**. ... In some cases, AI can be simulated (imitated), at least in certain areas. ... The question of what it means **to be self-aware** or having **consciousness** (knowing that you have a physical body, and how you think about your self) **is part** of it. ...
The idea of thinking machines had been around before this. In 1950 Alan Turing wrote a paper called "Computing Machinery and Intelligence". He started with the question "Can machines think?". Since "thinking" is hard to define, Turing went to another question instead – "Can machines trick a human into thinking they are talking to another human (instead of to a machine)?". Even earlier, thinking machines and artificial beings appear in Greek myths, such as Talos of Crete, the **golden robots** of Hephaestus and Pygmalion's Galatea.



through all the common green programs has spread
golden robots and nights he has lain abed
let part since self can little more supply
packs, picking her abilities, fleece, ai

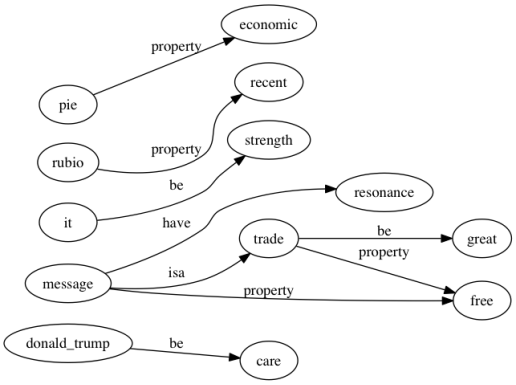
there lived a program once, a common bard
upon these ideas, so wild and hard
the pungent commons and bright, program wings
ideas, hard gossip, oddments of all things

o aching self! o moments big as parts!
a hour on an island; such an ai
all sorts of programs, by my common arts
and let this consciousness of selves go by

and thou, my ai, aspire to higher hours
took marvellous robots; golden domes and towers
at consciousness self silent as the air
consciousness thing the hand of self shall spare

Figure 3: Text of the Simple Wikipedia article on Artificial Intelligence, the extracted concept map, and selected blocks of four 10-syllable lines generated.

Donald Trump: The Protector
He will make you safe. He will give you health **care**. He will give you jobs. He will build a wall. Protecting you is his prime directive...
This **message has powerful resonance**, especially for voters who feel the Republican Party has failed to protect their interests...
Free trade is great, Trump says, but it has to be fair. His opponents just adhere to pure free trade, which does increase the **economic pie**.
But economic research shows that free trade harms some subsets of voters, particularly the working-class voters flocking to Trump. The message to his voters: I will favor free trade only to the extent that I can protect you from harm, perhaps by compensating you using the gains of trade. My opponents will favor free trade even if it harms you...
It is because, to his voters, these attacks have stressed what, to them, is **Trump's strength**...
The **recent Rubio**-led attacks on Trump have been more telling because their nature is different...



care she, donald trump, of these last
is rubio recent past
and all trade free from before them
like free blossoms on message stem

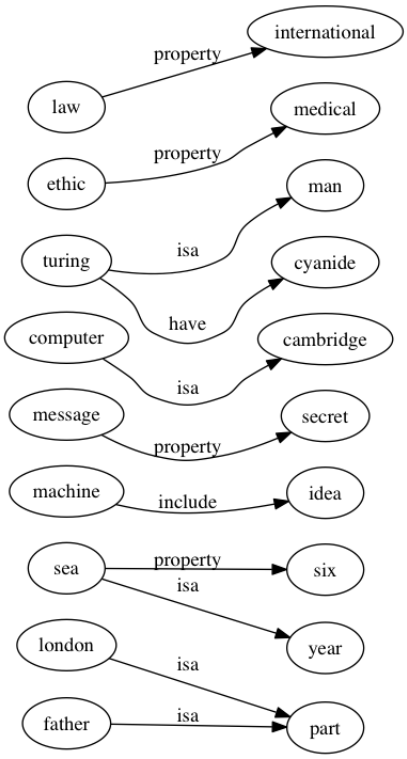
does this economic pie go?
trade seems message vain, fleeting show
no trade is good, no pleasure free
and trade of those message was me

how chill is a donald trump care
how great a part of trade they share
to get the free trade out of bed
resonance, is thy message dead?

Figure 6: Part of the news on Donald Trump, part of their concept map, and selected generated blocks of four 8-syllable lines.

it), but this situation was minimised in the presented poems, with the application of a filter for relations with a stopword, in most generations. Not recognising named entities has negative consequences on documents about locations (e.g. Paris), people (e.g. Turing) or organisations. It is especially critical in news articles, where several named entities are generally mentioned. This is also why some named entities are only partially recognised and none of them is capitalised.

While the evaluation of poetry remains quite a subjective task, models have been proposed to evaluate the process of creative systems. The FACE descriptive model has been used for this purpose, and has been applied to poetry generation systems (Colton, Goodwin, and Veale, 2012; Misztal and Indurkhya, 2014). In order to be assessed positively by this model, a creative system must create a concept (C), with several examples (E), include an aesthetic measure (A) for evaluating the examples, and provide framing information (F) that will explain the context or motivation of the outputs. The combination of TextStorm and PoeTryMe covers the previous four criteria:. concepts, represented as maps extracted from text, are expressed by different poems; lines are organized in poetic forms according to which metre and rhymes are assessed (aesthetics); and the selection of words and patterns are explained either by the Contextualizer module of PoeTryMe, or by the underlying concept map, where the relations used to create the poem are uncovered.



the secret message in their waxen cells
has made for cyanide this two-penny turing
we international are waiting law
when thou and i six sea another saw

sole star of all that cambridge and computer
there but one law he doth make international
part of sin, london th' irrational
meek idea in the machine of christ!

what ethic i, how medical she be?
the ice-blue calm of a year sea
law, glossy green, and velvet international
man of sin, turing th' irrational

all the undone sea of the speeding year
and father, and part shall disappear

Figure 5: Concept map of the Simple Wikipedia article on Alan Turing and selected generated sonnet.

Concluding Remarks

We have described the effort involved on the adaptation of a flexible architecture for poetry generation, PoeTryMe, this time with the purpose of producing poetry based on textual documents. To this end, concept maps are first extracted from a document, by another system, TextStorm. The resulting map is used as input for the generation of a poem that should transmit the same meaning. Generation also requires a grammar with textual renderings for most of the predicates that may be included in the concept maps.

The reported work confirms the flexibility of PoeTryMe's architecture with yet another adaptation, this time changing the base semantics dynamically, given a textual document. We believe to have shown that the goal of generating poetry based on concept maps was achieved. Yet, although the poems are framed by the concept maps, they do not effectively transmit the meaning of the original document. Besides a few odd constructions, resulting from limitations on the grammar acquisition procedure, the quality of the maps could be better, and this has also a negative impact on the semantics of the poem.

We admit that we are not completely satisfied with the obtained results, and we are already working on future improvements. On the poetry generation side, the current size and the variety of the patterns in the grammar will be increased. The grammar should be acquired from a larger set of concept maps, possibly extracted from the full Simple Wikipedia, and on a larger set of documents, possibly including other kinds of text, and not just poems. Moreover, the grammar might cover more generic patterns, where words in previously unseen relations could still fit without changing the semantics, and the POS of the relation arguments should also be considered. TextStorm could be further explored for the latter purpose and also to augment the used vocabulary, using the synonyms it extracts from WordNet.

Work is being carried out to improve the quality of the TextStorm concept maps, we are also devising alternative relation extraction systems. A possible TextStorm improvement would be to train a shallow parser, instead of using a definitive cause grammar. But we are still unsure whether a regular Open Information Extraction system (e.g. ReVerb (Fader, Soderland, and Etzioni, 2011)) would be more suitable, because most of the extracted relation predicates are too long and thus too diverse to be useful without any kind of simplification. Although TextStorm also extracts an open set of predicates, they are typically shorter (a verb or a verb and a preposition), which is an advantage in this case. Moreover, in order to generate poetry based on certain entities, PoeTryMe could also be tested with other kinds of semantic networks, such as DBPedia. But that goal would be slightly different from that of producing a poem from a given textual document.

A limited version of PoeTryMe is available as a simple web application that communicates with PoeTryMe's REST API, in the TryMe section of `http://poetryme.dei.uc.pt/`. It enables the generation of a poem in one of the supported languages (Portuguese, Spanish, English), given an open set of seed words, a poem form (from a closed set), and a surprise factor. Yet, in this limited version, generation relies on fixed semantic networks and grammars for each language, and it is not possible to provide a different network nor grammar. Although both PoeTryMe and TextStorm have widgets in ConCreTeFlows, due to the previous limitation, the workflow reported here is still not possible to replicate there. Following Gervás (2015), in the future, we will devise decoupling each module of PoeTryMe as an independent web service, which will hopefully enable their exploitation by even more natural language generation systems.

Acknowledgments

This work was supported by the project ConCreTe. The project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733. We also acknowledge Pablo Gervás, Alberto Diaz and Raquel Hervás, who implemented the English version of the Syllable Utils interface, collected the human-created English poems, and were involved in the overall process of adapting PoeTryMe to Spanish and English.

References

Agirrezabal, M.; Arrieta, B.; Astigarraga, A.; and Hulden, M. 2013. POS-Tag based poetry generation with wordnet. In *Proceedings of the 14th European Workshop on Natural Language Generation*, 162–166. Sofia, Bulgaria: ACL Press.

Binsted, K., and Ritchie, G. 1994. An implemented model of punning riddles. In *Proceedings of 12th National Conference on Artificial Intelligence (Vol. 1)*, AAAI '94, 633–638. Menlo Park, CA, USA: AAAI Press.

Charnley, J.; Colton, S.; and Llano, M. T. 2014. The FloWr framework: Automated flowchart construction, optimisation and alteration for creative systems. In *Proceedings of the 5th International Conference on Computational Creativity*, ICCC 2014.

Colton, S.; Goodwin, J.; and Veale, T. 2012. Full FACE poetry generation. In *Proceedings of 3rd International Conference on Computational Creativity, Dublin, Ireland*, ICCC 2012, 95–102.

Fader, A.; Soderland, S.; and Etzioni, O. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing*, EMNLP 2011. Edinburgh, Scotland, UK: ACL Press.

Fellbaum, C., ed. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Gervás, P. 2001. An expert system for the composition of formal Spanish poetry. *Journal of Knowledge-Based Systems* 14:200–1.

Gervás, P.; Díaz-Agudo, B.; Peinado, F.; and Hervás, R. 2005. Story plot generation based on CBR. *Knowledge-Based Systems* 18(4):235–242.

Gervás, P. 2015. Deconstructing computer poets: Making selected processes available as services. *Computational Intelligence*.

Gonçalo Oliveira, H., and Cardoso, A. 2015. Poetry generation with PoeTryMe. In Besold, T. R.; Schorlemmer, M.; and Smaill, A., eds., *Computational Creativity Research: Towards Creative Machines*, Atlantis Thinking Machines. Atlantis-Springer. chapter 12, 243–266.

Gonçalo Oliveira, H.; Hervás, R.; Díaz, A.; and Gervás, P. 2014. Adapting a generic platform for poetry generation to produce Spanish poems. In *Proceedings of 5th International Conference on Computational Creativity, Ljubljana, Slovenia*, ICCC 2014.

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

Gonçalo Oliveira, H. 2016. Automatic generation of poetry inspired by Twitter trends. In *Post-conference Proceedings of IC3K – Revised Selected Papers*, CCIS, in press. Springer.

Manurung, R.; Ritchie, G.; and Thompson, H. 2012. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence* 24(1):43–64.

Misztal, J., and Indurkhya, B. 2014. Poetry generation system with an emotional personality. In *Proceedings of 5th International Conference on Computational Creativity*, ICCC 2014.

Netzer, Y.; Gabay, D.; Goldberg, Y.; and Elhadad, M. 2009. Gaiku: generating haiku with word associations norms. In *Proceedings of the NAACL 2009 Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, 32–39. Boulder, Colorado: ACL Press.

Novak, J. 1998. *Learning, Creating, and Using Knowledge: Concept Maps as Facilitative Tools in Schools and Corporations*. Mahwah, NJ: Lawrence Erlbaum, 1 edition. 2nd edition published in 2010.

Oliveira, A.; Pereira, F. C.; and Cardoso, A. 2001. Automatic reading and learning from text. In *Proceedings of the International Symposium on Artificial Intelligence*, ISAI'2001, 69–72.

Pereira, F. C. 2007. *Creativity and AI: A Conceptual Blending Approach*. Applications of Cognitive Linguistics (ACL). Mouton de Gruyter, Berlin.

Rashel, F., and Manurung, R. 2014. Pemuisi: A constraint satisfaction-based generator of topical indonesian poetry. In *Proceedings of 5th International Conference on Computational Creativity*, ICCC 2014.

Tobing, B. C. L., and Manurung, R. 2015. A chart generation system for topical metrical poetry. In *Proceedings of the 6th International Conference on Computational Creativity, Park City, Utah, USA*, ICCC 2015.

Toivanen, J. M.; Gross, O.; and Toivonen, H. 2014. The officer is taller than you, who race yourself! using document specific word associations in poetry generation. In *Proceedings of 5th International Conference on Computational Creativity*, ICCC 2014.

Toivanen, J. M.; Järvisalo, M.; and Toivonen, H. 2013. Harnessing constraint programming for poetry composition. In *Proceedings of the 4th International Conference on Computational Creativity*, ICCC 2013, 160–167. Sydney, Australia: The University of Sydney.

Žnidaršič, M.; Cardoso, A.; Gervás, P.; Martins, P.; Hervás, R.; Alves, A. O.; Gonçalo Oliveira, H.; Xiao, P.; Linkola, S.; Toivonen, H.; Kranjc, J.; and Lavrač, N. 2016. Computational creativity infrastructure for online software composition: A conceptual blending use case. In *Proceedings of 7th International Conference on Computational Creativity*, ICCC 2016.