Poet's Little Helper: A methodology for computer-based poetry generation. A case study for the Basque language

Aitzol Astigarraga and **José María Martínez-Otzeta**
and **Igor Rodriguez** and **Basilio Sierra** and **Elena Lazkano**
Department of Computer Science and Artificial Intelligence
University of the Basque Country UPV/EHU
Donostia-San Sebastian 20018, Spain
aitzol.astigarraga@ehu.eus

Abstract

We present Poet's Little Helper (PLH), a tool that implements a methodology to generate poetry using minimal language-dependent information. The user only needs to provide a corpus with a set of sentences, a rhyme checker and a syllable-counter. From these building blocks, PLH produces: (1) an exploratory analysis of the suitability of the given corpus for poetry generation. (2) a novel and non-trivial poem grammatically correct under metrical and rhyming constraints. This poem also shows content that is coherent with a topic given by the user. The process of poetry generation is a cycle with three phases: lexical exploratory analysis, semantic exploratory analysis and poem generation. The goal is twofold: on the one hand PLH aims to be a useful open source poem-generator for many languages with minimal effort; on the other hand, analizes how the particularities of each corpus affect in the creation of poems. The presented PHL tool is offered in a public repository. The results of an experiment with a corpus of Basque texts is shown.

1 Introduction

Poetry is a form of literary expression intended to convey emotions in the audience, and in which the expression of feelings and ideas is given intensity by the use of style and rhythm according to predefined formal rules. Poetry generation is a challenging field in the area of Natural Language Processing. When a poem is automatically created by computational means, usually the programmer takes advantage of existing general semantic knowledge from resources like WordNet for semantic validation, or of already known formalized grammars for sentence generation. Furthermore, most of the existing poetry-generation systems are devoted to the English language (Gonçalo Oliveira, 2015). For minority languages with low presence in the natural language processing community the most usual scenario is a lack of such computational resources.

In this paper we present a methodology to help researchers in generating poetry automatically. This methodology is composed of two different phases: the first one devoted to the exploratory analysis (lexical and semantic) of the corpora provided for poetry generation, and the second one focused on the automatic generation of the final poem given the results of the previous phase. We provide a tool which implements the above mentioned steps.

The rest of the paper is organized as follows: section 2 introduces the main poetry generation systems and the strategy implemented in the PLH tool; section 3 aims to give a formal definition of the presented methodology; section 4 describes the source code of the PLH tool; section 5 shows an application of the proposed method to the Basque language; and the final section 6 presents the conclusions and possible future lines of research.

2 Poetry Generation

Computational modeling for poetry generation has become a topic in the artificial intelligence community in recent years. Before the computer science community took an interest in the area, people with a background closer to humanities made early efforts in systematic generation of poetry. We could mention works related to generating variations over a predetermined set of verses (Queneau, 1961), or to select a template to produce poems from it (Oulipo, 1981).

In recent years many different computer-based poetry generation systems have been developed. Among the most relevant ones we could include the well-known WASP (Gervás, 2000; Gervás, 2010), a Spanish verse-generation system developed following the generate-and-test strategy; Full-FACE (Colton et al., 2012), a corpus-based poetry generation system which introduces emotions in the generation process; PoeTryMe (Gonçalo Oliveira et al., 2014; Gonçalo Oliveira and Cardoso, 2015), a poetry generation platform used for Portuguese, Spanish and English; DopeLearning (Malmi et al., 2016), where the task of lyric generation is formulated as an information retrieval task. An approach using text mining methods, morphological analysis, and morphological synthesis to produce poetry in Finnish is presented in (Toivanen et al., 2012). Constraint programming for poetry composition is explored in (Toivanen et al., 2013). In (Agirrezabal et al., 2013) an approach of poetry generation based on POS-tag is described. Markov chains are also widely used as a basis of poetry generation systems. Popular and recent examples of N-gram generators are (Barbieri et al., 2012; Das and Gambäck, 2014; Gervás, 2016).

For a more thorough review of systems related to automatic generation of poetry, we point the reader to (Gervás, 2013) and (Lamb et al., 2016).

Our poetry generator is based on the following principles:

- Form: rhyme and metric compound the technical requirements of a verse. Thus, finding rhymes and counting syllables are essential abilities that the system must perform.

- Content: the output of the verse generator module must be meaningful. Methods to measure the semantic coherence of the generated text are needed.

The verse generation procedure relies in the extraction of sentences from corpora and combining them (under rhyme and metric constraints) to form the final poem. Semantic relatedness or internal coherence between poem verses is measured with an implementation of LSA (Latent Semantic Analysis) method (Deerwester et al., 1990). The main assumption of LSA is that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per verse is constructed from a corpus and singular value decomposition (SVD) is used to reduce the dimensionality of the semantic space. Verses are then compared by taking the cosine distance between their two vector representations, where a higher value is associated with a higher semantic similarity.

Thus, in a basic scenario, the user provides a topic and the kind of poem to be composed, and the system aims to give as output a novel poem with content semantically related to the proposed topic.

3 Methodology

In this section the proposed methodology is described. The process of poetry generation is a cycle with three phases: lexical exploratory analysis, semantic exploratory analysis and poem generation. Lexical and semantic richness is largely required for acceptable poem generation, and exploring the lexical and semantic dimensions of the data could help the researcher to focus on improving it along their weaknesses.

3.1 Lexical exploratory analysis

Regarding the lexical exploratory analysis, we define the following actions:

- *Count the number of potential verses.*

- *Find the number of verses which do not adjust to the rhyming convention, and show their endings.*

- *Find the number of verses which rhyme with a given word, and list them.*

- *Compute the number of rhyming equivalence classes of the set of verses.* A rhyming equivalence class is a set of verses which share the same rhyming pattern.

- *Compute the number of rhyming equivalence classes of the set of verses that have more elements than the minimum number of rhyming verses in a poem.* This is the number of valid equivalence classes, in the sense that elements from the other equivalence classes cannot form part of a poem.

- *Create a list with a verse from every equivalence class along with the number of elements in such equivalence class.*
- *Plot the number of verses in each equivalence class.*
- *Plot the logarithm of the number of verses in each equivalence class.* Useful when the distribution is very skewed.
- *Plot the histogram of the number of equivalence classes according to the equivalence class size.* Another way of exploring the distribution of equivalence classes according to their size.
- *Plot the histogram of the number of equivalence classes according to the logarithm of the equivalence class size.*

3.2 Semantic analysis

The semantic exploratory analysis subtask is comprised of several steps:

- *Build a semantic model from the set of documents Ds provided by the user.*
- *Find the verses more similar to a given theme according to the semantic models.*
- *Find the verses more similar to a given theme according to the semantic models and that also rhyme with a sentence.*

3.3 Poetry generation

For the poetry generation subtask the steps are the following:

- *Ignore equivalence classes with fewer elements that the minimum needed.* In this step the equivalence classes from which a poem cannot be created are ignored.
- *Compute the best poems given a theme according to a goodness function.* Several goodness functions are available to create poems.

3.4 Formal definition

Let us start with some terminology. $\mathbb{N}$ will denote the set of natural numbers. $\mathbb{R}$ will denote the set of real numbers. A document $d \in D$ is a sequence of words ($w \in W$) and punctuation marks ($m \in M$) and spaces which follows the syntactic conventions of the language. A verse $v \in V$ is a document d under some restrictions. The power set of a set S will be denoted with $\mathcal{P}(S)$.

The elements that the user has to define before applying this methodology are the following:

- *A set of documents (Ds) which is used to infer the semantic models used later.*
- *A set of documents (Dv) which is used to obtain the verses.*
- *A set of (M) of punctuation marks.* The elements of this set will be removed when performing semantic analysis.
- *A natural number $NV \in \mathbb{N}_{\neq 0}$ denoting the number of verses in a poem.*
- *A sequence of rhyme patterns $RP \in \{0, 1, ..., NV\}^{NV}$.* The rhyme patterns in the poem have to be encoded in the following manner: the rhyme pattern of the first verse corresponds to the number zero; for the pattern of a new verse, if such a rhyming pattern already exists, the number corresponding to that pattern is written, or the first natural number not yet chosen otherwise. For example, four verses with the same rhyming pattern would be written as 0, 0, 0, 0. Six verses with rhyming patterns by consecutive pairs will be 0, 0, 1, 1, 2, 2. The pattern 0, 1, 1, 2, 2, 1, 1, 0 would corresponding to eight verses where the last three patterns are the same as the first three patterns, but reversed.
- *A function ($extract\_verses : D \rightarrow \mathcal{P}(V)$) that extracts all the potential verses from a document.*
- *A lemmatizer function ($lemmatize : W \rightarrow W$), which returns a lemmatized word.*

- *A rhyming function ($is\_rhyme : D^2 \rightarrow \{T, F\}$) that returns True if the two documents rhyme, and False otherwise.*

The system should already have other elements defined. In the following functions, $\mathbb{S}$ will denote the set of semantic models.

- *A function ($clean\_document : D \times M \rightarrow D$) which returns the document without punctuation marks.*

- *A function ($semantics\_extractor : D \times \mathbb{N} \times W^* \times \mathbb{N} \times \mathbb{R} \rightarrow \mathbb{S}$) that returns a semantic model.* This function takes five parameters to generate a semantic model (LSA). The parameters are: a set of documents used to build the semantic model; the number of topics which should be used to create the semantic model; a list of words to be removed from the documents; a parameter indicating the minimum document number in which a word should appear; and a parameter indicating the maximum fraction of the total documents in which a word could appear.

- *A similarity function ($sim : D^2 \rightarrow R$), which returns the similarity between two documents.*

- *A rhyme detector function ($rhyme\_generator : D \times \mathcal{P}(V) \rightarrow \mathcal{P}(V)$), which returns all the rhyming verses given a document.*

- *Several poem generator functions ($poem\_generator : V \times D \times \mathbb{N} \times \mathbb{N}^{NV} \times \mathbb{S} \rightarrow \mathcal{P}(V)$), that given a set of verses, a document, a number of verses, a rhyming pattern and a semantic model, returns a poem under the constraints imposed by the number of verses and the rhyming pattern; the poem will be semantically related to a document under a semantic model.* The document above referred can be viewed as the theme of the poem. These generator functions will differ in their inner implementation of another two auxiliary functions: a poem validator function ($poem\_validator : \mathcal{P}(V) \rightarrow \{T, F\}$), which returns True if the poem conforms to the poetry rules, and False otherwise, and a poem goodness function ($poem\_goodness : \mathcal{P}(V) \rightarrow \mathbb{R}$), which returns a value for every set of verses according to its quality.

4 Open source code

A public repository has been created with the basic code needed to implement this methodology [1]. The code is a collection of Python modules and Jupyter notebooks, which, after installation, allows people with little knowledge of the Python language to perform the analysis on their own.

The structure of the code is the following:

4.1 Modules

- *Customize.* It contains the following definitions, that have to be customized according to the needs of the researcher: the filenames where D_s and D_v are stored, M and *RP* as Python tuples, the natural number *NV*, and the functions *extract_verses*, *lemmatize* and *is_rhyme*.

- *General.* General functions not directly related to natural language processing or poetry generation. For instance, list manipulation or histogram drawing functions are defined here.

- *NLP.* Functions related to natural language processing. The construction and manipulation of semantic models takes place here.

- *Poetry.* Functions related to poetry generation. Rhyming and poem construction takes place here.

4.2 Notebooks

- *Get_started.* This notebook will be called from the other two. The researcher does not need to open it, given that it only contains some code that is automatically executed to initialize the system.

- *Exploratory_analysis.* Here, the code that allows exploration of the lexical and semantic possibilities of the verses is located.

[1] https://github.com/rsait/PLH

- *poem_generation.* The notebook where the last step, the poetry generation, is performed. The researcher, after analyzing the data with the *Exploratory_analysis* notebook, is ready to execute this code and create automated poetry.

5 Case study

Verse improvisation (under the name of *Bertsolaritza*) is a traditional cultural expression in the Basque Country. With ancient roots, it has undergone a revival in the last times, being widely popular.

In this section we present the experiments made with a corpus of Basque texts to produce poems under the formal requirements of Basque poetry. This corpus is the set of all the news that appeared in the Basque newspaper Egunkaria in the years 2002-2003, which comprises 1,277,457 sentences. The results of the exploratory analysis are shown along with some automatically produced poetry. The selected poetry meter for experiments is *Zortziko Txikia*, a composition of eight lines in which odd lines have seven syllables and even ones have six. The union of each odd line with the next even line, forms a verse. Each verse has 13 syllables with a caesura after the 7th syllable (7 + 6) and must rhyme with the others[2].

5.1 Building blocks

As previously said, some building blocks are needed in order to apply the proposed methodology. In this case those blocks were defined in the following manner:

- The set of documents Ds from which the semantics models are created is the Basque corpus previously referenced.
- The set of documents Dv from which extract the potential verses is equivalent to Ds.
- The set of punctuation marks is M = (, . ? ! " ' / \).
- A $syllable\_counter$ function that counts the syllables of the input text.
- The rhyming function is_rhyme that returns all the rhyming lines given an input line. It is based on (Amuriza, 1981) and implemented using regular expressions.
- The natural number NV that denotes the number of verses in a poem. In *Zortziko Txikia* this number is 4.
- The sequence of rhyme patterns RP. In *Zortziko Txikia* this sequence is (0, 0, 0, 0). It means that all the verses have to rhyme among themselves.

5.2 Lexical exploratory analysis

The following actions have been performed automatically with the help of our lexical exploratory software:

- Count the number of potential verses: 41659.
- Find the number of verses which do not adjust to the Basque rhyming conventions: 139. Percentage of the total: 0.33%.
- Find the last words of such verses. Analyzing these words we find the sign %, making us wonder if we should expand the set M, filter these kind of characters or make another decision. We also find interjections ("eh", "hi"), foreign proper names ("olaf", "jerusalem", "bush"), Basque proper names ("unanue", "orue"), Roman numerals ("xix", "xx", "xxi"), acronyms ("eajk", "ugt", "upn") and other words not easily classifiable. After these step we could decide what to do in every case: for example, we could modify the rhyming function to add those Basque names, expand acronyms or Roman numerals in the original documents and repeat the verse extraction process, or remove all the no rhyming verses. We have chosen this last option, that is provided by our software.
- Compute the number of equivalence classes of the set of verses, according to the rhyme. In this example the number of partitions is 184.
- Compute the number of equivalence classes of the set of verses that have more elements than

[2] `http://www.bertsozale.eus/en/bertsolaritza/what-is-a-bertso`

the minimum number of rhyming verses in a poem. In Zortziko Txikia the rhyme pattern (RP) is (0, 0, 0, 0), which means any valid partition has to contain at least four elements, because a poem is composed by four rhyming verses. The number of equivalence classes of minimum size in our example is 141. If RP would be (0, 1, 0, 1, 0, 1), the minimum equivalence class size would be three, and in the case of (0, 1, 0, 1, 2, 2), the minimum size would be two.

- Create a list with a verse of every equivalence class along with the number of elements in such equivalence class. Our list is of the form [('a aita zenarekin joan zen bertara', 4441), ('a arrieta hartu zituzten mendean', 4209), ... , ('zeelandarrekin ez da ariko lomu', 1), ('ziganda badiola zalakain gaztelu', 1)].

- Plot the number of verses in each equivalence class (figure 1), the logarithm of the number of verses in each equivalence class (figure 2), the histogram of the number of equivalence classes according to the equivalence class size (figure 3) and the histogram of the number of equivalence classes according to the logarithm of the equivalence class size (figure 4).

These four figures can help the user in the interpretation of the distribution of the rhyming equivalence classes and their relative size, that appears to follow a power law (Piantadosi, 2014).

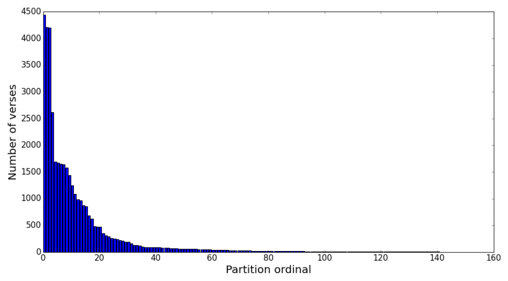

Figure 1: Number of verses in each equivalence class

5.3 Semantic exploratory analysis

The following actions have been performed automatically with the help of our semantic exploratory software:

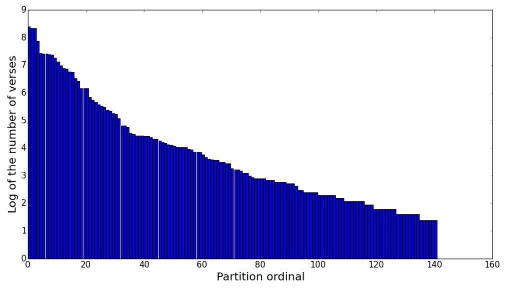

Figure 2: Logarithm of the number of verses in each equivalence class

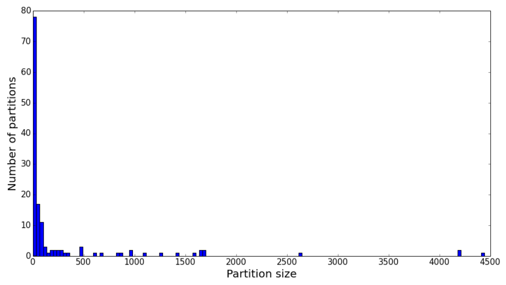

Figure 3: Histogram of the number of equivalence classes according to the equivalence class size

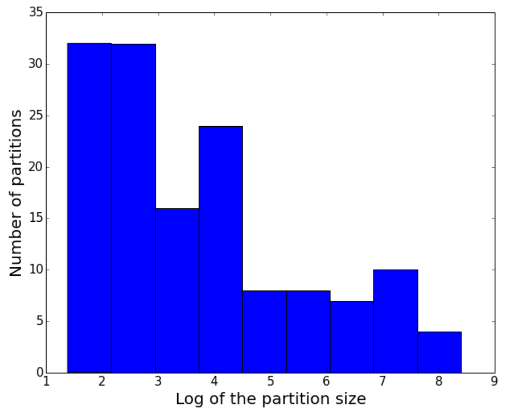

Figure 4: Histogram of the number of equivalence class according to the logarithm of the equivalence class size

- Build a semantic model from the set of documents Ds provided by the user. The number of topics has been assigned to 100, filtering all the words that do not appear in at least 5 documents, and all the words that appear in more than 20% of the documents. We have also filtered the stopwords.

- Find the verses more similar to a given theme according to the semantic models. Taking for example the theme "itsaso" (sea), we find sentences with the word ”itsaso”, but also sentences without that word, but other related words, as ”txalupa” (small boat) or ”ur” (water). We find sentences with the word ”ontzi”, that could be translated as ”ship” but also as a ”generic container”, highlighting the challenging issues which polysemy implies.

- Find the verses which are more similar to a given theme according to the semantic models and that also rhyme with a sentence. Following with the ”itsaso” example, the more similar verse is ’oliobideetan eta itsasoan’, with a similarity value of 0.99833471). The three sentences more similar to ”itsaso” that also fulfil the rhyming restrictions, along with their similarity values, are (’zenbait otzara eta ontzi hareatzan’, 0.91221404), (’paper eta ontzien birziklapenean’, 0.88736451), and (’delas eta izura oraindik ontzian’, 0.83049005).

5.4 Poetry generation

For a poem to be valid, it has to be composed of the number of verses given by *NV* and follow the rhyming pattern given by *RP*. No two verses are allowed to share the same final word. In the following we will refer to the document to which the poem has to be semantically related, as the theme t. Two different *poem_generator* strategies have been used to build a poem.

1. Choose the verse v more similar to the theme t. Then choose the $NV - 1$ verses more similar to t that follow the rhyming pattern RP.

2. Choose the ten verses v more similar to the theme t. Then, for each of the ten verses, choose the $NV - 1$ verses more similar to t that follow the rhyming pattern RP. The poem with highest score is chosen.

For each of these two functions, two examples are shown, using different themes t. In the first example, the theme ”itsaso” (sea) has been chosen, and in the other ”gurasoak” (parents) has been used. NV is equal to four, and the RP pattern is (0, 0, 0, 0), meaning that all the verses have to share the same rhyming pattern.

The verse more similar to the theme is ’oliobideetan eta itsasoan’ (in oil paths and in the sea). The poem generated choosing the three verses more similar to to theme is shown in Table 1.

Table 1: Poem created with the verse more similar to the theme ”itsaso” (”sea”) (in Basque and its English translation)

Basque	oliobideetan eta itsasoan
	atentatu susmoak itsaso beltzean
	itsaso baretura itzuli nahian
	zenbait otzara eta ontzi hareatzan
English	in oil paths and in the sea
	attack rumors in the sea’s darkness
	trying to return to calm sea
	several container and ships are going on

Table 2: Best poem in our opinion with the theme ”itsaso” (”sea”) (in Basque and its English translation)

Basque	Kantauri itsasoa haserre zeharo
	ozeano haunditan egin dugu txango
	putz egitea ere tokatu ezkero
	maitatu eta negar egin baitut Bilbo
English	Cantabrian Sea very angry
	we have made a trip to the vast stormy ocean
	if we are emerged to make blow
	I have loved and cried, Bilbao

The same experiment has been performed choosing the best ten verses and then computing the best poem among the ten ones generated. The same poem is ranked the first with this approach, but in our opinion, the poem in Table 2 (which ranked 8th of 10) is the best of all ten.

Table 3: Poem created with the verse more similar to the theme ”gurasoak” (”parents”) (in Basque and its English translation)

Basque	nire gurasoentzat Peret kristona zen
	haiekin bi urteko alaba zeukaten
	familia osoak topa zitezkeen
	noizbait haur bat badator nahiz ez jakin nor den
English	Peret was very good in my parents’ opinion
	with these people they had a two years old daughter
	for all the family to get together
	sometimes a child comes and I do not know who (s)he is

In the second example the theme ”gurasoak” (”parents”) has been chosen. With the same procedure as in the first example, the verse more similar to the theme is ’nire gurasoentzat Peret kristona zen’ (Peret[3] was very good in my parents’ opinion), and the poem generated choosing the three verses more similar to the theme is shown in Table 3. When performing the same experiment with the best ten verses, another poem, shown in Table 4 is chosen. As in the previous example, we found the poem in Table 5 (ranked 10th of 10) the best for our liking.

[3] A Spanish singer

Table 4: Best poem built from the ten verses more similar to the theme ”gurasoak” (”parents”) (in Basque and its English translation)

Basque	ikastolako haurren txanda izango da
	haurtzaindegietatik haur eskoletara
	haur danborradarako Easo Ederra
	eta gero eta haur gehiago dira
English	It will be school children turn
	from nursery to school
	children drum performance in San Sebastian
	there are more and more kids

Table 5: Best poem in our opinion with the theme ”gurasoak” (”parents”) (in Basque and its English translation)

Basque	ahizpa ere haurrei irakasten dabil
	aitona hola dabil makur eta ixil
	zuk errondak atera bai ibili trankil
	Londreseko Paddington geltokitik hurbil
English	(S)he is teaching to his/her sister and children
	the grandfather goes on bowed and silent
	you carry on calm proposing challenges
	close to London’s Paddington station

Let us remember that the goal of the methodology and the associated code is to help the user to explore the possibilities of their data. In this example we find that the generated poems are not of high quality, and that even we do not agree with the relative ordering of them given by the code. So, which conclusions could we extract from these facts? We already have tools to find all the verses related to a theme ordered by relevance, so one first step could be to check if such ordering is suitable. If that it is not the case, it is very likely the process of the semantic model construction needs some tuning. Or maybe the problem lies in the few number of verses that rhyme with the most promising candidates. This could be also explored with our tool. In our case, it looks as if the verses with a haiku-like structure are better valued by our ear. This makes us wonder if the poem goodness function could take into account this fact, and weight down the poems with all the verses very related to the theme, or those with too many repeated words between verses.

6 Conclusions and further work

In this work a methodology to guide the exploration of the possibilities of a collection of documents to perform automatic poetry generation has been described. Along with it a tool and its source code written in Python has been presented. The possibilities of the system have been shown with an example in Basque language.

As further work, we intend to add more functionalities to the lexical and semantic exploratory subsystems, as well as to the poetry generation subsystem. Another ways of building poems, as for example using genetic algorithms or Markov chains, would be of interest. The *poem_goodness* functions are fixed and predefined, but it would be possible to be customizable by the researcher. Another idea would be to get a feedback from the researcher or a knowledgeable user about the subjective goodness of a poem, in order to improve the goodness functions that the system uses. At this moment a brute force approach is applied in the lexical and semantic exploration and in the poetry generation, which could imply a heavy computational load with big corpora. We plan to tackle these issues in new versions of the software.

Acknowledgements

This paper has been supported by the Spanish Ministerio de Economía y Competitividad, contract TIN2015-64395-R (MINECO/FEDER, UE), as well as by the Basque Government, contract IT900-16.

References

Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga, and Mans Hulden. 2013. POS-Tag based poetry generation with WordNet. In *ENLG 2013 - Proceedings of the 14th European Workshop on Natural Language Generation, August 8-9, 2013, Sofia, Bulgaria*, pages 162–166.

Xabier Amuriza. 1981. *Hiztegi Errimatua (Rhyming Dictionary)*. Lanku Kultur Zerbitzuak (publisher).

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 115–120. IOS Press.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-FACE poetry generation. In *Proceedings of the Third International Conference on Computational Creativity*, pages 95–102.

Amitava Das and Björn Gambäck. 2014. Poetic machine: Computational creativity for automatic poetry generation in bengali. In *5th International Conference on Computational Creativity, ICCC*, pages 230–238.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990.

Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

Pablo Gervás. 2000. WASP: Evaluation of different strategies for the automatic generation of Spanish verse. In *Proceedings of the AISB-00 Symposium on Creative & Cultural Aspects of AI*, pages 93–100. Citeseer.

Pablo Gervás. 2010. Engineering linguistic creativity: Bird flight and jet planes. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, pages 23–30. Association for Computational Linguistics.

Pablo Gervás. 2013. Computational modelling of poetry generation. In *Artificial Intelligence and Poetry Symposium, AISB Convention 2013*, University of Exeter, United Kingdom. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

Pablo Gervás. 2016. Constrained creation of poetic forms during theme-driven exploration of a domain defined by an N-gram model. *Connection Science*, 28(2):111–130.

Hugo Gonçalo Oliveira and Amílcar Cardoso. 2015. Poetry generation with PoeTryMe. In *Computational Creativity Research: Towards Creative Machines*, pages 243–266. Springer.

Hugo Gonçalo Oliveira, Raquel Hervás, Alberto Díaz, and Pablo Gervás. 2014. Adapting a generic platform for poetry generation to produce spanish poems. In *ICCC*, pages 63–71.

Hugo Gonçalo Oliveira. 2015. Automatic generation of poetry inspired by twitter trends. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, pages 13–27. Springer.

Carolyn Lamb, Daniel G Brown, and Charles LA Clarke. 2016. A taxonomy of generative poetry techniques. In *Bridges Finland Conference Proceedings*, pages 195–202.

E. Malmi, P. Takala, H. Toivonen, T. Raiko, and A. Gionis. 2016. Dopelearning: a computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 195–204.

Association Oulipo. 1981. *Atlas de litterature potentielle*. Collection Idees. Gallimard.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21(5):1112–1130.

Raymond Queneau. 1961. *100.000.000.000.000 de poemes*. Gallimard Series. Schoenhof's Foreign Books.

Jukka Toivanen, Hannu Toivonen, Alessandro Valitutti, Oskar Gross, et al. 2012. Corpus-based generation of content and form in poetry. In *Proceedings of the Third International Conference on Computational Creativity*, pages 175–179.

Jukka Toivanen, Matti Järvisalo, Hannu Toivonen, et al. 2013. Harnessing constraint programming for poetry composition. In *Proceedings of the Fourth International Conference on Computational Creativity*, pages 160–167.