PoeLM: A Meter- and Rhyme-Controllable Language Model for Unsupervised Poetry Generation

Aitor Ormazabal[1] **Mikel Artetxe**[2] **Manex Agirrezabal**[3] **Aitor Soroa**[1] **Eneko Agirre**[1]
[1]HiTZ Center, University of the Basque Country (UPV/EHU)
[2]Meta AI [3]University of Copenhagen
{aitor.ormazabal,a.soroa,e.agirre}@ehu.eus
artetxe@meta.com manex.aguirrezabal@hum.ku.dk

Abstract

Formal verse poetry imposes strict constraints on the meter and rhyme scheme of poems. Most prior work on generating this type of poetry uses existing poems for supervision, which are difficult to obtain for most languages and poetic forms. In this work, we propose an unsupervised approach to generate poems that follow any given meter and rhyme scheme, without requiring any poetic text for training. Our method works by splitting a regular, non-poetic corpus into phrases, prepending control codes that describe the length and end rhyme of each phrase, and training a transformer language model in the augmented corpus. The transformer learns to link the structure descriptor with the control codes to the number of lines, their length and their end rhyme. During inference, we build control codes for the desired meter and rhyme scheme, and condition our language model on them to generate formal verse poetry. Experiments in Spanish and Basque show that our approach is able to generate valid poems, which are often comparable in quality to those written by humans.

1 Introduction

Despite the impressive generative capabilities of large Language Models (LMs) (Brown et al., 2020; Chowdhery et al., 2022; Zhang et al., 2022) automatic poetry generation remains a challenging problem. **Formal verse poetry**, in particular, imposes strict constraints on the meter and rhyme scheme of poems (Figure 1), which cannot be directly controlled in conventional LMs.

Prior work on generating formal verse poetry has primarily focused on supervised approaches, leveraging existing poems to train LMs. This is often combined with additional techniques to impose the meter and rhyme constraints at inference time, such as using finite-state automata to discard invalid candidates (Ghazvininejad et al., 2016), or generating text right-to-left to better control the rhyming word

Pen|san|do | que el | ca|mi|no i|ba | de|re|cho, →<LEN:11><END:echo>
vi|ne a | pa|rar | en | tan|ta | des|ven|tu|ra, →<LEN:11><END:ura>
que i|ma|gi|nar | no | pue|do, aún | con | lo|cu|ra, →<LEN:11><END:ura>
al|go | de | que es|té un | ra|to | sa|tis|fe|cho. →<LEN:11><END:echo>

Figure 1: **A formal verse poem and its associated structure descriptor.** The poem is the first stanza of a Spanish sonnet, which must have 4 lines of 11 syllables and follow an ABBA rhyme scheme. We use control codes to describe such constraints, and train a language model that can generate text conditioned on them.

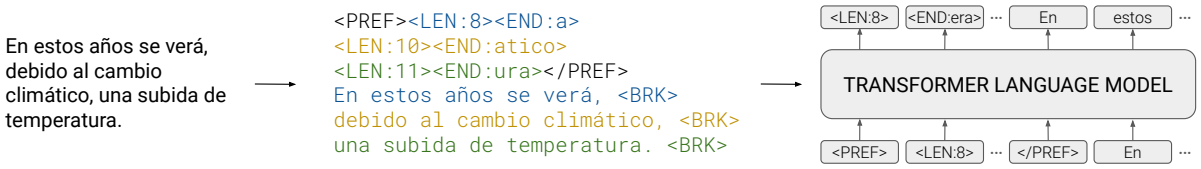
(Lau et al., 2018; Jhamtani et al., 2019; Xue et al., 2021). However, these approaches require poetic text for training, which is difficult to obtain for most languages and poetic forms.

In this paper, we propose an unsupervised approach to generate formal verse poetry. Our **Poe**tic **L**anguage **M**odel (PoeLM) can be conditioned to follow any desired meter and rhyme scheme, without requiring any poem for training. As illustrated in Figure 2, the key idea behind our method is that any text can be divided into phrases, which will each have a certain number of syllables and end in a certain sound that can make it rhyme with other phrases. While this structure will not follow a regular pattern for standard text, as it would for poetry, we can still annotate it automatically, and train a language model that can be conditioned on such structure descriptors. At inference time, we build a structure descriptor for the desired meter and rhyme scheme, and condition our language model on it to generate formal verse poetry. To improve results, we generate multiple candidates, which are automatically filtered and re-ranked.

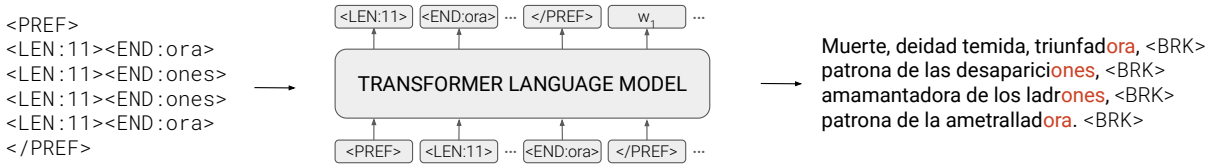
Our experiments in Spanish and Basque show that our method is able to generate high quality poems meeting the desired meter and rhyme constraints, with human evaluators ranking our system higher than other humans in more than one third of the cases. Our code is available at GitHub.[1]

[1]https://github.com/aitorormazabal/poetry_generation

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3655 - 3670
December 7-11, 2022 ©2022 Association for Computational Linguistics



(a) Training on regular, non-poetic text. Example in Spanish.



(b) Generation of formal verse poetry. Example in Spanish.

Figure 2: **Proposed method.** (a) During training, we split non-poetic text into phrases according to punctuation marks, prepend control codes describing the length and end rhyme of each phrase, and train a transformer language model on it. (b) During inference, we build a structure descriptor with control codes for the desired meter and rhyme scheme, and condition our language model on them to generate formal verse poetry.

2 Background: formal verse poetry

Poetic traditions differ across languages and cultures. In this work, we focus on **formal verse poetry** in Spanish and Basque,[2] which impose strict meter and rhyme constraints as follows:

- The **syllabic meter** specifies the number of lines in the poem, as well as the number of syllables that each line must contain.[3] Spanish syllabic meter allows for synalephas, where two syllables can be merged into one when one word ends in a vowel and the next starts with one. For simplicity, we do not consider synalephas when counting syllables, although our method could easily be extended to account for them.

- The **rhyme scheme** specifies the pattern according to which lines must rhyme. For instance, the ABAB scheme requires the 1st line to rhyme with the 3rd one, and the 2nd line to rhyme with the 4th one. Two lines are considered to rhyme if they repeat the same sound at their last syllables.[4] In addition, rhyming lines cannot end in the same word.

There are different **poetic forms** depending on the specific meter and rhyme scheme that they impose. For instance, the first stanza of a Spanish sonnet must consist of four verses with 11 syllables each, following an ABBA rhyme scheme. As illustrated in Figure 1, we use control codes to define such meter and rhyme constraints, which we refer to as **structure descriptors**.

3 Proposed method

As described in §2, we want our system to be able to generate text that adheres to a specific structure. The key idea behind our approach is that, similar to formal verse poetry, any text adheres to a certain implicit structure. In the case of non-poetic text the structure will not follow any regular pattern, but we can still extract it and build a structure descriptor for it. We can then augment the non-poetic corpus with these descriptors, and train a regular LM on it (Figure 2a). The model thus learns to respect the structure provided in the descriptor, which allows us to generate formal verse poetry at inference time, by conditioning the model on the appropriate structure descriptor (Figure 2b).

We next describe the two main components of our method: structure-aware training (§3.1) and inference with filtered re-ranking (§3.2).

[2]The selected languages where narrowed down according to the availability of publicly available high-quality syllabization and rhyme detection systems (which discarded English), as well as the fluency of the authors.

[3]Some traditions impose a stress pattern in addition to the number of syllables, which is known as *accentual-syllabic meter*. We do not consider this type of meter in our work, as it is not common in Spanish and Basque.

[4]In Spanish, two words rhyme if their sounds are identical from the last stressed vowel onwards. In Basque, two words rhyme if their sounds match from the first vowel of the second to last syllable onwards, and the following consonant groups are considered to sound the same for the purposes of rhyme: {p,t,k}, {n,m}, {s,z,x}, and {b,d,g,r}.

3.1 Structure-aware training

Let X be the space of possible text sequences, and S be the space of possible structure descriptors. We can define a function $s : X \rightarrow \mathcal{P}(S)$ that maps each sequence of text into its corresponding set of descriptors.[5] We want to build a model that can sample from $P(X|c \in s(X))$, that is, that can sample text conditioned on an structure descriptor c. In theory, one could do this through rejection sampling, by repeatedly drawing sentences from $x \sim P(X)$ until one of them satisfies $c \in s(x)$. However, this is intractable in practice, since the probability of a randomly sampled text following the desired structure is practically zero.

Instead, we train a LM that can be conditioned on any given structure (see Figure 2a). To that end, we start by **annotating the implicit structure** of a regular, non-poetic corpus. We first split the corpus in phrases, where we define a phrase as a sequence of text delimited by either a newline or a punctuation character (e.g., commas, colons or quotes). We do this so that the rhyme words at the end of these units correspond to natural stopping points. We then group the text in blocks of n phrases, where n is randomly sampled. For each block x, we choose a structure descriptor $c_x \in s(x)$ that defines the length and end rhyme of each of the phrases it contains. We then **create an augmented corpus** $(c_{x_1}, x_1, c_{x_2}, x_2, ...)$ by interleaving the previously generated structure descriptor c_{x_i} before its corresponding text block x_i (see Appendix A for more details). Finally, we **train a transformer LM** on the augmented corpus. The control codes in the structure descriptors are treated as regular tokens, and the model is trained with the standard next token prediction objective.

3.2 Generation with filtered re-ranking

At inference time, we use the LM from §3.1 to generate formal verse poetry in 3 steps:

1. Candidate generation. We specify the desired meter and rhyme scheme as a structure descriptor,[6] and use our LM to generate text conditioned on it (see Figure 2b). We repeat the process $k = 3000$ times to generate k different candidates. In our experiments, we provide the first line of the poem to generate in addition to its structure descriptor, which is useful to define the subject and make different systems easier to compare.

2. Filtering. In practice, some of the generated candidates do not meet the given constraints or are otherwise pathological. For that reason, we filter candidates according to the following conditions:

1. **#Line.** The candidate must have the number of lines specified in the structure descriptor.
2. **#Slb.** Each line must have the number of syllables specified in the structure descriptor.
3. **Rhyme.** Each line must end in the rhyme sound specified in the structure descriptor.
4. **Rep. word.** No two rhyming lines can end in the same word.
5. **BLEU.** In order to prevent the model from generating repetitive text, the maximum and average BLEU across any two lines must be be less than or equal to 35 and 20.

3. Re-ranking. We score the remaining candidates for fluency using our LM, and output the one with the highest score. Different from the first step, we do not condition on the structure descriptor when doing so, which gives a measure of the general fluency.

We test the efficacy of the second and third steps in the experiments.

4 Experimental design

We run experiments on Spanish and Basque. We next detail the training details (§4.1) and the automatic and human evaluation setup (§4.2 and §4.3).

4.1 Training details

Hyperparameters. We train transformer LMs using the same settings as Brown et al. (2020). For Basque, we train a 350M model with a learning rate of 3×10^{-4} and linear decay over 300B tokens.[7] For Spanish, we train a 760M model over 100B tokens using a constant[8] learning rate of 2.5×10^{-4}.

[5] Each sequence is mapped to a subset of S, as the same sequence could be described by multiple descriptors.

[6] A rhyme scheme specifies which lines must rhyme, but not what the rhyme sound should be. We thus generate a concrete structure descriptor from the given scheme by sampling each rhyme sound independently from the five most common rhyme sounds in the training corpus.

[7] In practice, we stop training after seeing around 85B tokens, when performance plateaus in the validation set.

[8] We initially planned to manually decay the learning rate according to validation perplexity. However, we did not observe performance plateauing (presumably due to the large corpus and our constrained compute budget), so the full training was done with a constant learning rate.

Corpora. We use EusCrawl (Artetxe et al., 2022) as our training corpus for Basque, which takes 2.5GB in plain text format, and a subset of 700GB from mC4 (Raffel et al., 2019) for Spanish. Given the small size of the Basque corpus, we combine 10 versions of the corpus using different random seeds to generate the structure descriptors.

Preprocessing. We use SentencePiece tokenization (Kudo and Richardson, 2018) with a 50k vocabulary for each language, and reserve 8.5k tokens for the control codes in the structure descriptors. For syllabification and rhyme sound extraction we use the rules provided by Agirrezabal et al. (2012),[9] which are encoded as finite-state transducers implemented in Foma (Hulden, 2009).

Models. In addition to our proposed model (PoeLM), we train a regular LM for each language as a baseline, using the exact same hyperparameters, tokenization, and corpora (without the interleaved structure descriptors).

4.2 Automatic evaluation

We use Spanish poems from the 20th century subset of the DISCO dataset (Barbado et al., 2021), and Basque poems from the BDB dataset[10] to evaluate our approach. The DISCO and BDB datasets consist of 20k and 44k tokens before our SentencePiece tokenizer is applied, respectively. We use the following automatic metrics:

Filtering rate. We take 10 poems[11] from each test set, extract the first line from them, generate poems for each as described in §3.2 following the meter and rhyme scheme of the original poem, with $k = 3000$ candidates for each, and measure the percentage of candidates that are filtered according to the criteria in §3.2. We compare the resulting filtering rate of our proposed PoeLM, which is conditioned on the relevant structure descriptor, and a regular LM, which is not conditioned on any structure but could still generate a valid poem given enough trials.

Since, unlike PoeLM, the baseline LM does not generate break tokens to separate lines, we split the generated text into lines according to the relevant number of syllables. When this cannot be done while respecting word boundaries, we consider that the candidate is rejected for breaking the *#slb* condition. As a consequence, generations from the baseline LM are never considered to be rejected due to the *#verse* condition.

Perplexity. To understand how well the model is able to leverage the information provided by a known structure, we compare the per-token perplexity of (i) PoeLM conditioned on the relevant structure descriptor, (ii) PoeLM without conditioning on any structure descriptor, (iii) the baseline LM. We do this both in the validation set of the non-poetic corpus used for training, as well as the poem datasets used for evaluation.

Consistent with training, we insert break tokens to separate lines for both PoeLM variants. However, these special tokens are excluded from the perplexity computation to make them comparable with the baseline LM.

4.3 Human evaluation

We run a qualitative evaluation in Spanish comparing poems generated by our system and humans. Given that writing poems is also challenging for humans, we consider both poems written by actual poets as well as layman volunteers. More concretely, we take the first line of 50 poems from the DISCO dataset, and compare 3 poems generated by completing them as follows:

- **Expert**: The original poem from DISCO from which the first line was extracted, authored by a renowned poet.
- **Layman**: Poems written by non-expert volunteers within a time limit of about 5 minutes.
- **PoeLM**: Poems generated by our system using the full pipeline described in §3.2.

We then give these 3 poems[12] to human evaluators in a random order, and ask them to rank from best to worst. We report results according to two metrics: the overall rank (the percentage of times that each system has been ranked in each position), and the head-to-head comparison (the percentage

[9] https://bitbucket.org/manexagirrezabal/syllabification_gold_standard

[10] https://bdb.bertsozale.eus/. We use the 2005 segment of the corpus.

[11] For Spanish, we use the first Stanza of full sonnets from DISCO, which consist of either 11 or 14 syllable lines, following a rhyme scheme of ABAB or ABBA. For Basque, we use *Zortziko Handia* poems from BDB, which consist of 8 lines, where the odd ones are 10 syllables long, the even ones are 8 syllables long, and only the even lines are required to rhyme.

[12] A 4th candidate, which we ignore when calculating the ratings, was also included for the analysis in §6.2.

	Spanish		Basque	
	PoeLM	LM	PoeLM	LM
Correct	30.9	0.0	23.4	0.0
Reject due to				
#Verse	3.7	0.0	9.6	0.0
#Slb	17.0	96.6	34.0	90.3
Rhyme	13.1	3.4	11.1	9.7
Rep. word	31.1	-	19.7	-
BLEU	4.2	-	2.3	-

Table 1: Percentage of filtered candidates, with a breakdown for the reason of rejection. See §4.2 for details.

		Spanish		Basque	
		poetic	prose	poetic	prose
Baseline LM		62.7	15.9	151.1	24.3
PoeLM	w/ struc	49.5	11.7	42.5	10.1
	no struc	129.5	18.0	634.2	81.4

Table 2: Perplexity of poetic and non-poetic (prose) corpora. See §4.2 for details.

of times that each system has been ranked before each other system).

All volunteers that wrote the poems, as well as those that ranked the candidates, are native Spanish speakers with university studies. While there was an overlap between both groups of volunteers, we made sure that volunteers were never asked to rank poems written by themselves. All volunteers are familiar with the fundamentals of formal verse poetry, but are not experts in the matter. Refer to Appendix B for more details.

5 Results

We next discuss our main results for the automatic (§5.1) and human evaluation (§5.2).

5.1 Automatic evaluation

We report **filtering rate** results in Table 1. We find that 30.9% of Spanish poems and 23.4% of Basque poems sampled from PoeLM meet the given constraints. While far from perfect, this means that sampling a few candidates is enough to obtain a valid poem with our approach. In contrast, none of the poems generated by the baseline LM is valid, showing that our proposed structure-aware training is critical to generate formal verse poetry with LMs. Regarding the reason for rejection, we find that the majority of candidates from PoeLM are discarded for repeating rhyming words, which the model was not directly trained to prevent.

S1 \ S2	Expert	Layman	PoeLM
Expert	-	54.0%	62.7%
Layman	46.0%	-	60.7%
PoeLM	37.3%	39.3%	-

Table 3: Percentage of times that system $S1$ is ranked ahead of $S2$ in the human evaluation.

	1st	2nd	3rd
Expert	44.0%	28.6%	27.3%
Layman	36.7%	33.3%	30.0%
PoeLM	19.3%	38.0%	42.7%

Table 4: Percentage of times that each system has been ranked in each position in the human evaluation.

Table 2 reports the **perplexity** results. When conditioned on structure descriptors, our model always outperforms the baseline LM, meaning that it is able to make better predictions accounting for the meter and rhyme constraints. However, when the structure descriptor is not provided, our model's perplexity is higher, presumably because the model did not see text without structure descriptors during training.

5.2 Human evaluation

We report head-to-head results in Table 3, and ranking results in Table 4. Human evaluators prefer poems generated by our system over those written by renowned poets in 37.3% of the cases. Similarly, our system does better than laymen in 39.3% of the cases. This shows that our system is able to generate high-quality poems, which humans often prefer over poems written by other humans. This can also be seen in the ranking evaluation, as our system has been ranked in first position in 19.3% of cases, and among the first two positions in 57.3% of cases.

Finally, it is surprising that layman poems are ranked above those from renowned poets nearly half of the times. We attribute this to the human evaluators themselves being laymen, leading them to prefer poems that use more plain language. This is also reflective of the subjective nature of the task, as different readers might enjoy poetry differently.

6 Analysis

We further analyze our system by quantifying at which portion of the poem its perplexity gain is highest (§6.1), experimenting with manual re-

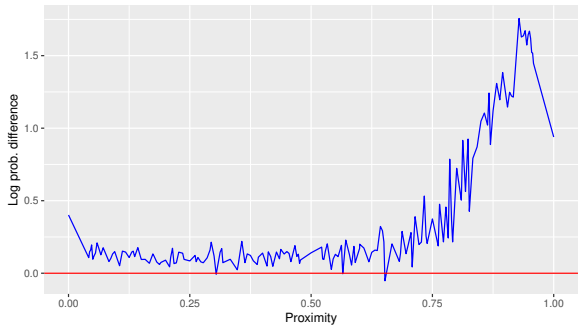


Figure 3: Interpolated advantage in log probability of our model compared to a regular LM over the Spanish mC4 validation set, as a function of normalized proximity to the next specified rhyme token. See §6.1 for details.

S1 \ S2	Exp.	Lay.	PoeLM	PoeLM +*rerank*
PoeLM	37.3	39.3	-	26.0
PoeLM +*rerank*	41.3	42.7	38.0	-

Table 5: Percentage of times that system $S1$ is ranked ahead of $S2$ in the human evaluation. Since the candidate chosen by a human annotator among the top 6 candidates will sometimes be the same as the top candidate, there can be ties, and thus the head-to-head percentages do not add up to 100.

ranking (§6.2), and looking at some sample poems (§6.3).

6.1 Perplexity gain

We quantify the predictive advantage of our system as a function of the distance to the next rhyme word. To this end, we plot the difference in token-wise log probabilities between our model and the baseline LM as a function of proximity to the next rhyme word, interpolated between 0 and 1. We only consider lines with 15 to 25 tokens.

As shown in Figure 3, our model's advantage is greatest near the rhyme word. This is not surprising, as there is less uncertainty towards the end of the line when the meter and rhyme are known. We observe a downward spike towards the end, that may initially seem counter-intuitive. We hypothesize that, since the rhyme word will often be split into multiple tokens, by the time the first tokens of the rhyme word are known the regular LM will be quite sure of what the word is, meaning that the advantage of knowing the rhyme is lower.

6.2 Manual re-ranking

A potential application of automatic poetry generation is helping (rather than replacing) humans when writing poems. As a first approximation, we ask our volunteers to manually choose a poem among the top 6 candidates generated by our system.[13] The resulting poem was considered as part of the human evaluation described in §4.3, and compared to the other 3 systems.

Table 5 reports the head-to-head performance of our model with and without manual re-ranking. As expected, the re-ranked model performs better, beating the poems generated by laymen in 42.7% of cases, as opposed to 39.3% for the base system. However, the base system beats manual re-ranking in 26% of cases, meaning that human evaluators often prefer the top candidate automatically selected by the system over the one manually selected by another human. This means that there is a considerable disagreement across annotators, which is reflective of the subjective nature of the task.

6.3 Sample poems

Tables 6 and 7 show some example poems generated by our system in Spanish and Basque. The poems were generated by providing the first line along with the structure descriptor to the system, and manually selecting a candidate among the top six. Five lines were selected at random from the evaluation in Spanish, and two for Basque. The full list with the six candidates is given in Appendix C. No *cherry-picking* was done, except to choose one poem among the six candidates per line.

We observe that the system is capable of generating coherent poems covering varied topics. For example, regarding the Spanish poems, the first, third and fourth Spanish poems cover themes of inner conflict, the second one describes a person's beauty, and the last is about an abbey called Monserrat. Note that the theme is implicit in the first line, and mirrors the typical topics of Spanish sonnets of the time. Regarding the Basque poems, the themes are friendship and swings in a park, also mirroring the themes used in contemporary spontaneous poetry contests in the dataset.

7 Related work

We next review relevant literature in poetry generation (§7.1), as well as controllable generation

[13] We take the top three candidates with and without BLEU filtering to generate this list.

Generated Poem	English translation
Siento otro Yo que contra mí se empeña, un Yo para el que no debo luchar, un Yo al que no debo acompañar, un Yo que a menudo me condena.	I feel another Self that is set against me, a Self for which I must not fight, a Self that I should not accompany, a Self that often condemns me.
Casta su faz, bajo la blanca toca, su faz es dulce,es bella sin cesar, su faz es hermosa como un jaguar, su faz es divina como la roca.	Chaste is her face, under her white cap, her face is sweet, relentlessly gorgeous, her face is beautiful like a jaguar, her face is divine like rock.
Nada más. De aquel sueño, que pasó como un ala, arranco todo lo que había en mi mente, todo lo que me atormentaba y no cala, lo que callo en el interior de mi presente.	Nothing more. Of that dream, that passed like a wing, it tore everything from my mind, all that tormented me and doesn't seep through, which I keep quiet inside my present.
Es inútil que luche por quitarme del pecho, que niegue repetidamente mis opiniones, que trague de nuevo mi entusiasmo deshecho, que rechace de nuevo todas mis negaciones.	Fighting to get it off my chest is futile, that I repeatedly deny my emotions, that I once again swallow my undone enthusiasm, that I once again reject all my negations.
Del Monserrat en la penumbra undosa, Del Monserrat silente en el Solar, Del Monserrat dolido en el remar, Del Monserrat cautivo en la prosa.	Of the Monserrat in the gloomy twilight, Of the Monserrat, silent in sunlight, Of the Monserrat, pained in paddling, Of the Monserrat, captive in prose.

Table 6: Spanish poems generated by our method, given five lines selected at random from the dataset. The five poems have been manually selected from the top six candidates generated by the system for each line, with no other form of *cherry-picking*. See Appendix C for the full list of six candidate poems.

(§7.2).

7.1 Poetry generation

Retrieval based approaches. Early work in poetry generation focused on rule-base methods, which generate text according to predefined rules that ensure the desired structure is followed (Gervás, 2000; Gonçalo Oliveira et al., 2007). A popular approach is to fill templates with text extracted from existing poems (Colton et al., 2012; Gonçalo Oliveira, 2012; Gonçalo Oliveira et al., 2017). This makes it easy to control poetic structure, since the meter and rhyme schemes of the text pieces can be annotated in advance and combined accordingly when filling the templates. However, the diversity and creativity of these approaches is limited.

Neural poetry generation. More recently, there has been work on applying neural text generation to poetry. A popular approach is to train a finite-state acceptor (FSA) that ensures all accepted sequences obey the required structure, which is then used to guide a recurrent neural network (RNN) through rejection sampling (Ghazvininejad et al., 2016, 2018; Hopkins and Kiela, 2017). However, these methods require some form of lyrical or poetic text to train the RNN or the FSA, and they must generate text right-to-left in order to respect rhyme sounds, as the model has no concept of planning. Additionally, a new FSA has to be trained for each desired poem structure. Lau et al. (2018) augment an RNN with a pentameter model and learn the meter and rhyme constraints of sonnets in a supervised way from a sonnet corpus. They then generate poem lines right-to-left, to alleviate the model's lack of planning. Van de Cruys (2020) trains an encoder-decoder RNN on prosaic text to generate each line right-to-left conditioned on the previous one, and applies constraints when decoding to ensure the generated text adheres to a rhyme scheme and consistent topic. However, their system cannot enforce a specific syllabic meter.

Multiple works focus on neural poetry generation for the Chinese language, applying techniques such as reinforcement learning (Yi et al., 2018) or planning (Wang et al., 2016). In Chinese, one character corresponds to a syllable, but meter is governed by tonal constraints. Most of the reviewed works assume that, with a sufficiently large corpus, the model should be able to learn the implicit tonal structure of poetry (Wang et al., 2016; Zhang et al., 2017; Liu et al., 2018). Yeh et al. (2019) concatenate tonal information to the character embeddings of an LSTM to create a model that is more phonologically compliant.

Notably, current neural methods capable of controlling both syllable count and rhyme scheme require some form of poetic corpus to train, and usu-

Generated Poem	English translation
Gu biok lagun handiak gara, anaia,aita,semea, eta bion ideologia, gure identitatea, konpartitzen dugu.Batzuetan, zaila da bat esatea, besteak ulertzea,benetan, zein ahula den bestea.	The both of us are great friends, brother,father,son, and our ideology, our identity, is shared. Sometimes, it is hard to say one, to understand others, truly, how weak others are.
Nahiz kulunpio pila bat egon, eguzkiak sikiera, aukera du ondo goxatzeko, eta ez beti gainera, baita asteko egun denetan, baita hemendik aurrera, ilargi erdiko orduetan, eta hori da ederra.	Even though there are many swings, at least the sun, has a chance to enjoy, not always, also during every day of the week, and, from now on, during the moon hours, and that is beautiful.

Table 7: Basque poems generated by our method, given two lines selected at random from the dataset. The poems have been manually selected from the top six candidates generated by the system for each line, with no other from of *cherry-picking*. See Appendix C for the full list of six candidate poems.

ally generate text right-to-left to alleviate a lack of planning when generating rhymes.

7.2 Controllable generation

Similar to our approach, several works attempt to control the generated output by augmenting the training data with tags. Keskar et al. (2019) augment the training corpus of a LM with codes automatically extracted from metadata. Some works in machine translation explore augmenting the training data in order to control the politeness (Sennrich et al., 2016), domain (Kobus et al., 2016), or length (Lakew et al., 2019) of generated translations. Schioppa et al. (2021) experiment with vector-valued additive tags in order to control multiple attributes of the generated text at once. However, all of these systems use tags that only broadly specify the length, domain or style of the text to generate. In contrast, our model is conditioned on a very specific meter and rhyme scheme that the text must follow.

8 Conclusions and future work

In this work, we present an unsupervised approach to generate formal verse poetry. We identify and extract the latent structure in non-poetic corpora, and feed this information along with the text to a transformer LM, allowing us to control the structure of the text at generation time. Our system is capable of generating formal verse poetry with flexible meter and rhyme schemes, without requiring any sort of poetic text to train. The required structure can be easily altered by changing the descriptor, allowing us to generate different types of poetry without needing to re-train the system. Automatic and human evaluations show that our model learns to leverage the provided structure information to better predict the text, and is capable of generating short poems that are often preferred to those created by a human.

In future work, we would like to extend our framework to be able to control other aspects of the generated text in addition to meter and rhyme.

Limitations

Given that our method requires tagging the implicit meter and rhyme of the training corpus, we are limited by the quality of available syllabization and rhyme detection systems. While rule-based systems with a low error rate are easy to create for languages such as Spanish or Basque, this is not the case for English, which is why we did not train an English version of our system. However, our approach is independent of the used syllabization and rhyme detection process, and could be readily applied on top of any system with a low error-rate.

Additionally, our Spanish syllabization system has no concept of synalephas, where two syllables can be merged into one when one word ends in a vowel and the next starts with one. This means that our system will never use this Spanish literary device when generating poems.

Acknowledgements

Aitor Ormazabal, Aitor Soroa and Eneko Agirre were partially supported by the Basque Government (IXA excellence research group IT1343-19). Aitor was supported by a doctoral grant from the Spanish MECD.

We would like to thank Ainara Estarrona, Begoña Altuna and Itziar Gonzalez-Dios for their assistance in defining literary terminology, and Edward Yao for his help translating poems.

References

Manex Agirrezabal, Inaki Alegria, Bertol Arrieta, and Mans Hulden. 2012. Finite-state technology in a verse-making tool. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 35–39.

Mikel Artetxe, Itziar Aldabe, Rodrigo Agerri, Olatz Perez de Viñaspre, and Aitor Soroa. 2022. Does corpus quality really matter for low-resource languages?

Alberto Barbado, Víctor Fresno, Ángeles Manjarrés Riesco, and Salvador Ros. 2021. DISCO PAL: Diachronic spanish sonnet corpus with psychological and affective labels. *Language Resources and Evaluation*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Simon Colton, Jacob Goodwin, and Tony Veale. 2012. Full-face poetry generation. In *International Conference on Computational Creativity 2012*, pages 95–102. University College Dublin.

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.

Marjan Ghazvininejad, Yejin Choi, and Kevin Knight. 2018. Neural poetry translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 67–71, New Orleans, Louisiana. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

Hugo Gonçalo Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Hugo Gonçalo Oliveira, Amílcar Cardoso, and Francisco Pereira. 2007. Exploring different strategies for the automatic generation of song lyrics with tra-la-lyrics. In *Proceedings of 13th Portuguese Conference on Artificial Intelligence, EPIA*, pages 57–68.

Hugo Gonçalo Oliveira, Raquel Hervás, Alberto Díaz, and Pablo Gervás. 2017. Multilingual extension and evaluation of a poetry generator. *Natural Language Engineering*, 23(6):929–967.

Jack Hopkins and Douwe Kiela. 2017. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 168–178, Vancouver, Canada. Association for Computational Linguistics.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*, pages 29–32.

Harsh Jhamtani, Sanket Vaibhav Mehta, Jaime Carbonell, and Taylor Berg-Kirkpatrick. 2019. Learning rhyming constraints using structured adversaries. In *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 6025–6031, Hong Kong, China. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Catherine Kobus, Josep Maria Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *CoRR*, abs/1612.06140.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Surafel Melaku Lakew, Mattia Di Gangi, and Marcello Federico. 2019. Controlling the output length of neural machine translation. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong. Association for Computational Linguistics.

Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. 2018. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia. Association for Computational Linguistics.

Dayiheng Liu, Quan Guo, Wubo Li, and Jiancheng Lv. 2018. A multi-modal chinese poetry generation model. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Andrea Schioppa, David Vilar, Artem Sokolov, and Katja Filippova. 2021. Controlling machine translation for multiple attributes with additive interventions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6676–6696, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

Tim Van de Cruys. 2020. Automatic poetry generation from prosaic text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2471–2480, Online. Association for Computational Linguistics.

Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. 2016. Chinese poetry generation with planning based neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1051–1060, Osaka, Japan. The COLING 2016 Organizing Committee.

Lanqing Xue, Kaitao Song, Duocai Wu, Xu Tan, Nevin L. Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021. DeepRapper: Neural rap generation with rhyme and rhythm modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 69–81, Online. Association for Computational Linguistics.

Wen-Chao Yeh, Yung-Chun Chang, Yu-Hsuan Li, and Wei-Chieh Chang. 2019. Rhyming knowledge-aware deep neural network for chinese poetry generation. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 1–6.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic poetry generation with mutual reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3143–3153, Brussels, Belgium. Association for Computational Linguistics.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative Chinese poetry generation using neural memory. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1364–1373, Vancouver, Canada. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models.

A Structure descriptors

The process of extracting the meter descriptors from a regular corpus and creating the augmented corpus consists of four steps:

1. First, we split the text into phrases according to the following set delimiters: _-?"!,:’‘()[].{}‘;»«><’. We do this so that the phrases, which will correspond to lines in

our generated poems, end at natural stopping points in speech. Additionally, we randomly merge each phrase with the next or the next two phrases, with probabilities 0.15 and 0.05, respectively, so that the model can generate verses that contain these special characters.

2. Second, we syllabize each phrase and extract the rhyme class of its final word, using our FOMA transducers.

3. Third, we split the text into blocks of n phrases, where n is sampled uniformly between 3 and 10. For each block, we construct a meter descriptor from the syllable count and rhyme class of each phrase. The descriptor begins with a <PREF> token and ends with a </PREF> token, and a pair of tokens of the form <LEN_X> <CLS_Y> for each phrase, where X is the syllable count and Y is the rhyme class. In 15% of cases, the rhyme class is replaced with a special <CLS_UNK> class, which allows us to leave the rhyme of certain verses unspecified when generating. Additionally, when there is a paragraph boundary (line break) in the text, we insert a <SEP> token in the corresponding position.

4. Fourth, we construct the corpus interleaving the meter descriptors in between the corresponding blocks of text. Additionally, we insert a <BRK> token in between phrases in the actual text. The <BRK> token lets us clearly see where the model has intended to end a phrase, allowing us to split the generated poems into verses easily.

B Human evaluation

As described in §4.3, we extract the first line from fifty real poems in the DISCO dataset, and compare the poems obtained by completing them. Specifically, for each case, we present the four candidate completions in a randomized order, and ask a human volunteer evaluator to rank them from best to worst. Additionally, the rhyme scheme that the poem should follow is provided, so that the evaluator can take it into account when judging a poem (our systems' poems and the gold standard never contain rhyme mistakes, but the poems generated by human annotators might). Each ranking is done by three different annotators.

All volunteers that participated in writing and ranking poems are familiar with the fundamentals of formal verse poetry, but are not experts in the matter. To obtain the "Layman" poems, volunteers were asked to complete a poem within a time limit of 5 minutes. However, this time limit was not enforced, and it was slightly exceeded in some cases.

C Full list of poems

Tables 8 and 9 show the full list of the six candidate poems generated by our system for each of the five lines in Spanish and two lines in Basque chosen at random from their respective datasets. No cherry-picking was done on these tables. From these poems, we did select one for each line manually, as shown in Tables 7 and 6 in the main body of the paper (see §6.3).

Siento otro Yo que contra mí se empeña,
Siento otro Yo que no quiere cambiar,
Siento otro Yo que no sabe amar,
Siento otro Yo que no se condena.

Siento otro Yo que contra mí se empeña,
siento otro Yo que no quiero amar,
siento otro Yo que no puedo hablar,
siento otro Yo que no me enseña.

Siento otro Yo que contra mí se empeña,
y otro Yo que me quiere dominar,
y otro Yo que no me quiere amar,
que me condena y que me condena.

Siento otro Yo que contra mí se empeña,
un Yo para el que no debo luchar,
un Yo al que no debo acompañar,
un Yo que a menudo me condena.

Siento otro Yo que contra mí se empeña,
o una voz que me hace suspirar,
o una risa que me hace llorar,
o un grito que me quita la pena.

Siento otro Yo que contra mí se empeña,
un Yo que no me deja amamantar,
un Yo que me ensena a no llorar,
un Yo que me enseña lo que pena.

Casta su faz, bajo la blanca toca,
vestida de dos trpitos de azahar,
Casta su faz,bajo la blanca tocar,
vestida de dos trpitos de su boca.

Casta su faz, bajo la blanca toca,
carvaja y danseá del palomar,
Casta su faz,bajo la blanca tocar,
carvaja y danseá de la roca.

Casta su faz, bajo la blanca toca,
casta su faz sin tabiques que tapar,
casta su faz cuando quiere cabalgar,
casta su faz con guarnición de boca.

Casta su faz, bajo la blanca toca,
su faz es dulce,es bella sin cesar,
su faz es hermosa como un jaguar,
su faz es divina como la roca.

Casta su faz, bajo la blanca toca,
ennoblece este cantar popular,
casta su faz,bajo la blanca tocar,
ennoblece este cantar de boca.

Casta su faz, bajo la blanca toca,
no dejaba errar;casta su mirar,
no dejaba turbar;casta su hablar,
no dejaba hacer;casta su boca.

Nada más. De aquel sueño, que pasó como un ala,
voló el agua como un ave frugalmente,
De aquel sueño,que pasó como una bala,
voló el viento como un átomo ardiente.

Nada más. De aquel sueño, que pasó como un ala,
no me devolviera la pálida piel caliente,
De aquel sueño,que pasó como una bala,
no me devolviera la herida en la frente.

Nada más. De aquel sueño, que pasó como un ala,
arranco todo lo que había en mi mente,
todo lo que me atormentaba y no cala,
lo que callo en el interior de mi presente.

Nada más. De aquel sueño, que pasó como un ala,
salió un pensamiento que viniera al frente,
un pensamiento que pasó como una bala,
salió un rayo que se encendió en mi mente.

Nada más. De aquel sueño, que pasó como un ala,
como un beso que se derrumbó de mi mente,
y que se fue al pasado como una bala,
y desapareció del mapa como un ente.

Nada más. De aquel sueño, que pasó como un ala,
un ave se paró sola en un parapente,
De aquel sueño,que pasó como una bala,
un viento se detuvo en el mar de enfrente.

Es inútil que luche por quitarme del pecho,
Es muy difícil que yo huya de mis traiciones,
Es demasiado tarde para lo que he hecho,
Es demasiado tarde para mis decisiones.

Es inútil que luche por quitarme del pecho,
que niegue repetidamente mis opiniones,
que trague de nuevo mi entusiasmo deshecho,
que rechace de nuevo todas mis negaciones.

Es inútil que luche por quitarme del pecho,
que no sea la tiranía de las pasiones,
que se ría de mí de lo que yo le he hecho,
Es inútil que se enoje por mis canciones.

Es inútil que luche por quitarme del pecho,
es inútil que llore por tus provocaciones,
es inútil que te diga qué es lo más hecho,
es inútil que afirme mis acusaciones.

Es inútil que luche por quitarme del pecho,
que luche por alcanzarme con sus oraciones,
que luche por bajarme del caballo derecho,
que me meta en mi cama con sus peticiones.

Es inútil que luche por quitarme del pecho,
que me refugie en mi casa de ilusiones,
que me grite a voces que quiero y no hecho,
que me arregle los días sin palpitaciones.

Del Monserrat en la penumbra undosa, Del Monserrat en la luz crepuscular, Del Monserrat en la vida de Aznar, Del Monserrat en la noche ansiosa.
Del Monserrat en la penumbra undosa, Del Monserrat en la niebla uncular, Del Monserrat en la luna anular, Del Monserrat en la noche brumosa.
Del Monserrat en la penumbra undosa, Del Monserrat en la noche un lugar, Del Monserrat en la luz un despertar, Del Monserrat en el sol una rosa.
Del Monserrat en la penumbra undosa, Del Monserrat con los dedos sin borrar, Del Monserrat con las ganas de cantar, Del Monserrat de la roca filosa.
Del Monserrat en la penumbra undosa, Del Monserrat silente en el Solar, Del Monserrat dolido en el remar, Del Monserrat cautivo en la prosa.
Del Monserrat en la penumbra undosa, Del Monserrat cuando hay que claudicar, Del Monserrat hasta que hay que llorar, Del Monserrat con mantilla larosa.

Table 8: Full list of six candidate poems for Spanish generated by our system for the five lines chosen at random in the Spanish dataset. No cherry-picking was done on this table. From these poems, we did select one for each line manually, as shown in Table 6 in the main body of the paper.

Gu biok lagun handiak gara,
anaia,aita,semea,
eta bion ideologia,
gure identitatea,
konpartitzen dugu.Batzuetan,
zaila da bat esatea,
besteak ulertzea,benetan,
zein ahula den bestea.

Gu biok lagun handiak gara,
bata zinemazalea,
bestea musikaria bera,
biek dugu lanbidea,
Denok egiten dugu aurrera,
dena da inportantea,
dena dago egiteko era,
izaten da idazlea.

Gu biok lagun handiak gara,
argazkilari zalea,
eta asko gustatzen zait bera,
nire bikotekidea,
argazki kamera eskuetan,
ezaguna izatea,
gogoratzen dudan argazkia,
lagunekin ikustea.

Gu biok lagun handiak gara,
Esne Beltzako kidea,
anai zaharra eta arreba,
Jon eta nire semea,
eta bi panpin gaude honetan,
Oihan Vega da bestea,
Hesian taldeko kantaria,
Sorotan Bele taldea.

Gu biok lagun handiak gara,
katuak ardi taldea,
tomateak eta haragia,
soja eta laborea,
oilaskoak fruitua lortzeko,
lekalearen jeztea,
baratzea animalietan,
pinpilinpauxa zaintzea.

Gu biok lagun handiak gara,
baina gure bikotea,
itxura guztien arabera,
ez da bideko jendea,
Bidaia ez da gauza berria,
ez da hala esatea,
Baina batzuetan,batzuetan,
izan daiteke jabea.

Nahiz kulunpio pila bat egon, galtza belauna behera, gero eskailerak,hortz parea, sokak,besoak atzera, kautxuzko motxilak bizkotxetan, eta astindu ederra, bularra,pultsuko aurpegia, izterra eta izterra.
Nahiz kulunpio pila bat egon, ez da oso gauza bera, autoa hartu eta kotxea, leku batetik bestera, beste leku batera sartzeko, jendearekin batera, egiten dugun bide horretan, kotxeak izorratzera.
Nahiz kulunpio pila bat egon, aspiranteen antzera, gorputza sentsuala da ia, betirako izaera, eta erretzen duzu airea, ez zara esterilera, kaka egiten duzu oinetan, likidoa da gainera.
Nahiz kulunpio pila bat egon, eguzkiak sikiera, aukera du ondo goxatzeko, eta ez beti gainera, baita asteko egun denetan, baita hemendik aurrera, ilargi erdiko orduetan, eta hori da ederra.
Nahiz kulunpio pila bat egon, komunera lasaitzera, komunera salto egiteko, irakurtzera,jotzera, txorimaloetaz gozatzea, konpainia egitera, burrunba burrunbatsuaz janztea, koadrilan afaltzera.
Nahiz kulunpio pila bat egon, bada zerbaiten plazera, dena zure eskura uztea, dakizuna arabera, izan zaitez supergizakia, izan nahi izatera, hau da,zuhaur,dena emateko, munduarekin batera.

Table 9: Full list of six candidate poems for Basque generated by our system for the two lines chosen at random in the Basque dataset. No cherry-picking was done on this table. From these poems, we did select one for each line manually, as shown in Table 7 in the main body of the paper.