Encoder-Decoder Framework for Interactive Lyrics Generation with Controllable High-Quality Rhyming

Anonymous ACL submission

Abstract

Composing poetry or lyrics involves several creative factors, but a challenging aspect of generation is the adherence to a more or less strict metric and rhyming pattern. To address this challenge specifically, previous work on the task has mainly focused on reverse language modelling, which brings the critical selection of each rhyming word to the forefront of each verse. On the other hand, reversing the word order requires that models be trained from scratch with this task-specific goal and cannot take advantage of transfer learning from a Pretrained Language Model (PLM). We propose a novel fine-tuning approach that prepends the rhyming word at the start of each lyric, which allows the critical rhyming decision to be made before the model commits to the content of the lyric (as during reverse language modelling), but maintains compatibility with the word order of regular PLMs as the lyric itself is still generated in a left-to-right order. We conducted extensive experiments to compare this fine-tuning against the current state-of-the-art strategies for rhyming, finding that our approach generates more readable text. Furthermore, we furnish a high-quality dataset in English and 12 other languages, analyse the approach's feasibility in a multilingual context, provide extensive experimental results shedding light on good and bad practices for lyrics generation, and propose metrics to compare methods in the future.

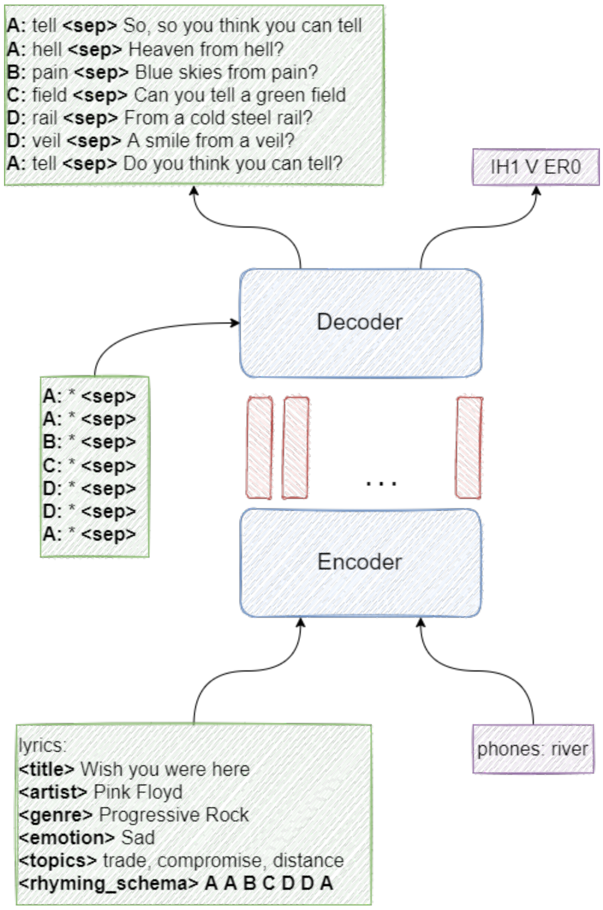

Figure 1: Drawing of the proposed model. Green boxes correspond to the main fine-tuning strategy LWF, while the violet ones correspond to the LWF+EPR approach.

1 Introduction

Lyrics generation, the task of generating lyrics based on desiderata defined by a user, e.g., genre or topic, is gaining momentum thanks to the recent advances in text generation. Generating lyrics, however, has its peculiarities, making it a different task from open text generation. Indeed, songs need to follow a high-level structure, defining choruses and verse and adhering to rhyming constraints. This is similar to the task of poetry generation, but songs present a vocabulary and styles that differentiate them. We chose Lyrics Generation since lyrics form our datasets and inherit these peculiarities. In general, a few approaches have been proposed either for specific cases, e.g., rap lyrics generation (Xue et al., 2021a), or in a more general fashion to adhere with desiderata from English songwriters (Ram et al., 2021), or to incorporate verse structure within a model (Li et al., 2020).

This work mainly focuses on the rhyming control aspect of generating lyrics. Previous work on the task (Xue et al., 2021a; Li et al., 2020) has focused on reverse language modelling, i.e.g training a model to generate the output in a right-to-left manner. This has the benefit of bringing the critical selection of the rhyming word to the forefront of each verse, ensuring that it is unaffected by the semantic context of the verse. The downside of this approach is that reversing the word order requires that models be trained from scratch with this task-specific goal. As such, these approaches cannot take advantage of transfer learning from Pretrained Language Models (PLM) which are generally trained through left-to-right conditional language modelling.

We propose a novel fine-tuning strategy, namely Last Word First (LWF), that can be used for generating human-like text with rhyming control. This strategy fine-tunes a model under a structure where the rhyming word (i.e. the last word of each verse) is prepended at the start of the verse as well (Fig. 1) and constrained (during inference) to follow the rhyming schema. This enables the model to follow a user-defined rhyming pattern and benefit from bringing the critical selection of the rhyming word to the forefront (as in reverse language modelling), while the rest of the verse is still being generated in a left-to-right manner. Our strategy allows us to fine-tune PLMs and benefit from transfer learning, requiring much less data and computing power, where previous work often required retraining a model from scratch. To the best our knowledge, no work has investigated finetuning PLMs to generate lyrics following an arbitrary rhyming pattern defined by a user.

We additionally condition the lyrics generation on various user-defined aspects, like genre or a specific artist's style, and explore how this strategy can be augmented through a secondary training objective to predict the Ending Phonetic Representation (EPR) of each word. Finally, while previous work primarily focused on only one language (e.g. English or Chinese), we introduce a novel high-quality dataset of lyrics in English and 12 other languages, and use it to demonstrate that LWF is language-agnostic. Our contribution is threefold:

1. A novel approach to adapt pretrained models so that they appropriately follow a given rhyming schema, enabling meaningful outputs and precision in rhyming while benefiting from PLM transfer-learning;

2. High-quality data in 13 languages for lyrics generation augmented with rhyme schema at the paragraph level;

3. Extensive experiments and error analysis, showing the pitfalls of current models, including multiple metrics over different aspects of the generation, like diversity (distinct-2, 3 and 4), Mauve, and a new metric on copyright.

Code and data will be released upon acceptance.

2 Related Work

The task of lyrics generation has to be viewed in the broader context of text generation. Text generation has recently gained much attention thanks to large pretrained language models, e.g., GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020), or encoder-decoder models, e.g., BART (Lewis et al., 2020), and T5 (Raffel et al., 2020). Compared to those only trained on the target task, the main advantage of such models is the so-called *knowledge transfer*, i.e., during their pretraining, assimilated knowledge can be later reused in different downstream tasks. A recent challenge in text generation is to add constraints to a model, from soft conditions, such as respecting a given style, to more complex rules, e.g., respecting a predefined schema for the output text, as when writing a poem.

Lyrics generation is a long-standing task in NLP, with the first attempts dating back to the 1960s (Queneau, 1961). More complex systems started spawning around the 2000s (Gervás, 2000; Manurung, 2004) and recently reached a more satisfactory performance thanks to the advent of deep learning, recurrent neural networks and PLMs (Wöckener et al., 2021; Shao et al., 2021; Li et al., 2020; Ormazabal et al., 2022). The lyrics' vocabulary and syntax are different from poetry and usually more contemporary; therefore, they need to be treated separately. Several works focused on the Rap and Hip Hop genres. They propose to model rhythm and rhyming with unique tokens within the text (Xue et al., 2021a) or to generate verses conditioned on input keywords and post-process the text to adhere with a rhyme schema (Nikolov et al., 2020). While most systems are stand-alone, i.e., do not require human intervention, nowadays, we observe a significant demand for human-in-the-loop approaches, that is, models that can help humans better pursue their goals. In this spirit, Ram et al. (2021) proposed a songwriter assistant able to consider different aspects of a song, including producing verses with a given metric or that rhyme with a given word.

We join this cause and propose an interactive approach to lyrics generation in English and other 12 languages, which can be conditioned on different song attributes and an arbitrary rhyme schema. Different from Xue et al. (2021a), our approach does not require reversing a text nor implementing architectural changes as in Li et al. (2020), allowing us to leverage the knowledge encoded within a PLM easily, yet being able to produce high-quality rhymes as requested by a songwriter. Furthermore, our approach is more flexible than the one proposed in Ram et al. (2021) as it allows us to define words each verse should rhyme with or generate a stanza from scratch, given only the desired rhyme schema. Finally, we show that our approach is language-agnostic and propose a unified neural network to produce lyrics in 13 languages.[1]

3 Model

We make use of an encoder-decoder architecture to condition the lyrics generation on a given set of inputs, such as *artist's style*, *title*, *genre*, *topics*, *emotions*, and *rhyme schema*. The input to the model is formatted as follows:

```
<BOS><title> The River <Artist> Bruce Springsteen
<emotions> sad <rhyming_schema> A B B <EOS>
```

Here, the model is trained to generate three verses, where the second and third rhyme. However, due to training through conditional language modelling, the last words of the verse tend to attend more heavily on the generated context than the rhyming schema, leading to non-rhyming output.

We propose the Last Word First (LWF) approach, that relies on anticipating the rhyming word. The last word of a sentence is generated as the first token right after the rhyming symbol and separated by the sentence it belongs to with the special token <sep>, as in the example below (and LWF models in Appendix A.1):

```
A: river <SEP> We'd go down the river <EOS>
B: dive <SEP> And into the river we'd dive <EOS>
B: ride <SEP> Oh, down the river we'd ride <EOS>
```

Consult Fig. 1; the green boxes represent the input/output of the model. The lower box is the encoder input that provides information for generating the lyrics. The left box is the rhyme schema template, passed explicitly to the decoder to enforce that it generates a stance coherent with the input rhyme schema. In more detail, the rhyme schema is forced at generation time by detecting when a sentence-end token is generated and forcing the following token to be the next rhyming symbol in the queue.

[1] Due to the resources all languages are European.

	Train	Dev	Test
# Examples	564K	3.5K	3.5K
With Genres	146K	587	804
With Emotions	27K	62	104
With Topics	170K	719	893
Avg. Tokens	46.92	74.90	70.79
Avg. Sentences	5.84	9.77	9.24
Avg. Sentence Length	8.03	7.67	7.67

Table 1: Statistics for Train, Dev and Test splits of the Genius.com dataset.

With this approach, we can generate coherent sentences that end with rhyming words and quickly identify rhyming patterns since a rhyming word always follows a rhyming symbol. Retaining pre-training knowledge is a key advantage, as opposed to training from scratch. Based on this strategy, we propose the following two models variants:

Plain Last Word First (LWF) the encoder-decoder model is fed with a prompt specifying different desiderata such as: *artist's style*, *title*, *genre*, *topic* and *emotions* and trained by minimising the cross-entropy loss at the token level. As usual for generation models, we use teacher forcing at training time, i.e., to predict the i-th token, we feed into the decoder the gold tokens up to time-step $i-1$. Formally, for each input, we minimise the following loss:

$$\mathcal{L} = \frac{1}{N}\sum_{i}^{N}\sum_{j}^{|V|} p_i^j log\ \hat{y}_i^j \quad (1)$$

$$\hat{y}_i = \mathcal{M}(X, t_1, \ldots, t_{i-1}) \quad (2)$$

where X is the input to our model $\mathcal{M}$, V is the model's vocabulary, $t_1, \ldots t_{i-1}$ are the gold tokens for the first $i-1$ timesteps, $\mathcal{M}(\ldots)$ outputs a vector of logits of size $|V|$ and $\cdot^j$ selects the j-th element of a vector.

Last Word First + Ending Phonetic Representation (LWF+EPR) Beyond lyrics generation, as the plain LWF model, this variant includes the secondary objective of generating the ending phonetic representation of a word given as input. Intuitively,

	Train	Dev	Test
# Examples	2.6M	10K	10K
With Genres	1.2M	4.3K	4.5K
With Topics	170K	493	493
Languages	13	13	13
Avg. Tokens	40.19	39.80	39.80
Avg. Sentences	6.99	6.79	6.60
Avg. Sentence Length	5.75	5.87	6.03

Table 2: Statistics for Train, Dev and Test splits of the multilingual dataset.

this objective helps inject the word's phonetic features into the model, thus helping to produce more accurate rhymes (see violet boxes in Figure 1). The model is trained through multitasking, by alternating batches between tasks, computing the cross-entropy losses, and updating the model weights separately. We computed the phonetic representation of a word by using CMU Pronouncing Dictionary (Weide et al., 1998), and, specifically, the *pronouncing* python library.[2] At training time, we use the list of the last words in the lyrics dataset and sample them according to their frequency.

4 Datasets

This section details the sources and procedures to recreate the datasets used within this work.

4.1 English Dataset

For the English data, we selected the top 1000 artists according to Spotify[3] and downloaded all their songs' lyrics available at `https://genius.com`.[4]. We ensure that the language is English [5]. Genius.com offers well-polished lyrics comprising annotations for choruses, pre-choruses, verses, etc. (Table 14). Since our goal is not to generate the full song lyrics all at once but to create verses that follow a specific rhyme schema and other desiderata, we need to reshape song lyrics data. To this end, we split each song into paragraphs corresponding to different parts, i.e., choruses, bridges, verses, etc. Thus, each item in our dataset is a song paragraph with its song's metadata, i.e., title, artist, genre, topics and emotion (whenever available). Furthermore, to allow a model to generate a stanza based on previous verses, we add to the metadata information the verses of the stanza preceding them, when appropriate.

In this work, we focus on cross-sentence rhyming, i.e., rhymes that occur between the last word of different sentences within the same paragraph, since this is the most common kind of rhymes in Pop music, as opposed to other rhetorical figures as alliteration. As Genius.com does not provide the rhyming schema, we computed it for each item, by first tokenising its lyrics. We added the phonetic representation of the last token of each verse[6]. Then, we compared them pairwise by applying Ghazvininejad et al. (2016)'s algorithm for rhymes and near rhymes in English to assign the same rhyming letter to all words that rhyme together. For example, given the lyrics in Table 14, we assign to the chorus this rhyme schema: `ABB`.

Once the dataset is created, we split it into three subsets for training, development and testing, respectively; we report their statistics in Table 1.

4.2 Multilingual Dataset

For languages other than English, we resort to data available within Wasabi (Buffa et al., 2021), an extensive database of songs containing lyrics and other metadata about roughly 2M of songs in 21 languages. To build our multilingual dataset, we kept pieces in all languages for which we can extract the phonemes[7] and filtered out those languages with less than 3000 songs. As a result, our dataset covers 12 languages plus English. Once we selected the languages, we built the dataset similarly to the English case. However, since Wasabi data is noisier than Genius.com ones, it is not always the case that a song can be clearly divided into sections. Therefore, in all those cases where such splitting is not explicit, we apply a simple heuristic and divide the songs into groups of 6 sentences.[8] In this case, the rhyme schema is also automatically induced by slightly modifying the algorithm of near rhymes

[2] `https://github.com/aparrish/pronouncingpy`

[3] `https://chartmasters.org/most-streamed-artists-ever-on-spotify/`

[4] we used python API available at `https://lyricsgenius.readthedocs.io`

[5] language detection: `https://pypi.org/project/phonemizer/`, `https://pypi.org/project/spacy-langdetect/https://pypi.org/project/stanza/` and spacy's language detector.

[6] We used the *phonemizer* python library available at `https://github.com/bootphon/phonemizer`

[7] Please, refer to `https://github.com/espeak-ng/espeak-ng/blob/master/docs/languages.md` for the list of *phonemizer* library's supported languages.

[8] We decided to use 6 since that is the average number of sentence for each paragraph in the Genius.com English training set.

used for English[9] by defining the set of vowels for each language of interest.[10]

The final dataset is created by merging all language-specific datasets and splitting them into training, development and testing subsets; we report the multilingual dataset statistics in Table 2.

5 Experimental Setup

This section introduces the research questions we aim to answer throughout our experiments, the results attained, and a human analysis of the lyrics we produced.

Models and training We carried out our experiments with the T5 (Raffel et al., 2020) encoder-decoder architecture for English experiments, through the transformers library.[11] For each one of the models, we used the two decoding strategies described in 5.3. we fine-tuned Multilingual T5 (Xue et al., 2021b, MT5) on our multilingual datasets.

Notation We indicate that a model finetunes a pretrained model with $*^{\text{Pretrain}}$ and with $*^{\text{Rand}}$ when training from scratch. The sub-incidences indicate the fine-tuning technique, with $*_{\text{LWF}}$, and $_{\text{LWF+EPR}}$ meaning Last-Word-First, and Last-Word-First plus Ending-Phonetic-Representation respectively. Regarding the decoding techniques, BS stands for Beam Search[12] while S+R stands for sampling sentences k[13] and reranking them according to adherence with the rhyme schema. Examples selected randomly can be found in Appendix A.1.

5.1 Evaluation Metrics

We evaluate the models with different metrics to provide information on various generation aspects.

Coherence metrics To assess to what extent the model was able to learn the language of lyrics and its structure, we chose two metrics: perplexity (PPL) as a base measure and Mauve for a deeper comparison of the distributions. For a deeper lookup of the second one, check the paper (Pillutla et al., 2021).

We use the model perplexity as a base measure to assess to what extent the model learnt the language of lyrics. For each song s, we consider the model perplexity as follows:

$$PP(s) = e^{H(s)} \tag{3}$$

$$H(s) = -\sum_{i}^{N} p(y_i|y_{<i}, x)\ log\ p(y_i|y_{<i}, x) \tag{4}$$

where, N is the number of tokens in s, y_i is the i-th token, $y_{<i}$ is the sequence of tokens before i and x is the input data, i.e., artist, title, topics, rhyme schema (as explained in Section 3).

Rhyming metrics As for evaluating rhyming, we measure the model macro precision concerning the required schema and the false positive rate between tokens that are not supposed to rhyme. Finally, we estimate the ability of the model to generate the number of sentences required by the input and the coverage in terms of necessary rhyming tokens. Formally, for each song, we compute the Rhyming Precision (RP) and the Rhyming False Positive Rate (R. FP) as follows:

$$P = \frac{1}{|R|}\sum_{t_i,t_j}^{R} \text{rhyme}(t_i, t_j)$$

$$FPR = \frac{1}{|NR|}\sum_{t_i,t_j}^{NR} 1 - \text{rhyme}(t_i, t_j)$$

where P is the rhyming precision, i.e., for each pair (t_i, t_j) in the set R of generated tokens that are supposed to rhyme according to the input schema, $\text{rhyme}(t_i, t_j)$[14] evaluates to 1 if t_i and t_j rhyme and 0 otherwise. Instead, FPR (False Positive Rate) measures the ratio of token pairs in NR, i.e., the set of generated tokens that are not supposed to rhyme while rhyming.

Diversity metrics To obtain how diverse the generated text is, we use distinct metrics, i.e., we measure the number of N-grams that are unique and divide it by the total number of N-grams in the text. In the tables 3 and 4 we can find the values for N equal to 2, 3 and 4 which we denote as D-2, D-3, and D-4, respectively.

[9]For tokenisation, we used *stanza* python library.

[10]We acknowledge that each language may have peculiarities to form rhymes. However, investigating all of them is out of the scope of this work, and it is left as a possible future direction.

[11]https://huggingface.co/docs/transformers/index

[12]We use beam equal 4

[13]We use $k = 20$ in our experiments.

[14]To evaluate whether two tokens rhyme, we apply the same approach described in Section 4.

CopyRight metric © To detect cases where the generated lyrics contained sentences from the original dataset, we created a string matching measure that we denote as ©. For each generated output, we compare it to each entry in the entire data set and find the longest subsequence, allowing a wrong token in the middle. If the length of the longest subsequence is above a pre-determined threshold, we choose 20 for our dataset and tokeniser, we consider the generated output to be at risk of being deemed plagiarised. We also calculate the percentage of this subsequence's length to the generated output's size as references for checking. The final score is the number of generated outputs at risk divided by the number of outcomes.

5.2 Research Questions

Through our experiments, we aim to answer the following research questions:

1. **Q1:** What is the impact of the LWF strategy in terms of rhyming accuracy?
2. **Q2:** Considering that generating lyrics differ from generating standard text, is the knowledge contained in a PLM still relevant regarding rhyming accuracy and text fluency?
3. **Q3:** Since syllables may be relevant when dealing with rhymes, what is the impact of tokenisation on the overall performance?
4. **Q4:** Does the phonetic information introduced by multitasking LWF+EPR result in any improvement in terms of rhyming?
5. **Q5:** Can we learn a model shared across languages while preserving rhyming accuracy?

We present models and corresponding results for each of these questions in the next section.

5.3 Results

English Evaluation We report the main results in Table 5. To answer **Q1**, we fine-tuned models with the LWF strategy (e.g. T5-L_{LWF}) and compare against the same architecture trained on plain data, i.e., in a left-to-right manner without LWF (e.g. T5-$L^{Pretrain}$). The LWF models (see Section 3), based on the simple yet effective approach proposed in this paper, attain consistently better results in terms of RP, R.FP, and Mauve. The best system in terms of rhyming is instead T5-B_{LWF} with Random initialisation and S+R, also beating its larger and pretrained counterpart (T5-L_{LWF}). Nonetheless, as Mauve indicates and we show in Table 7, T5-B_{LWF} produces much less meaningful lyrics, not creating human-like songs. Interestingly, the decoding technique highly affects the rhyming performance and copyright. While BS led to modest performance, we could boost performance with S+R in both aspects. As expected, Mauve and the three diversity metrics get better results with a sampling decoding.

To investigate whether the knowledge in a PLM is relevant to lyrics generation (**Q2**), we provide results attained by a T5-base model trained from scratch (T5-B^{Rand}_{LWF}) and compare against a pretrained version (T5-$L^{Pretrain}_{LWF}$). In preliminary experiments, we compared T5-$L^{Pretrain}_{LWF}$against T5-L^{Rand}_{LWF}, but finally opted for the base model as it produced better results. We assume this is due to the amount of data introduced during finetuning compared to the size of the dataset. T5-$L^{Pretrain}$ attains the best score on perplexity across the board, yet its RP and R.FP are the worst. On the other hand, T5-$L^{Pretrain}_{LWF}$obtains the best results in Mauve, a coherence metric with a higher correlation with humans. The superior coherence of the pretrained models will be further confirmed in Section 6, where we present human evaluation.

To provide insights on the role of the tokenisers (**Q3**), we trained T5-$B_{LWF}$with two different tokenisers, the original one for T5-base and a word-level one. In Table 4, we show that the word tokeniser, which may not split the end of the words into tokens, produces worse results even when training from scratch.

For **Q4**, we observe that EPR does not affect the model positively. While outperforming T5-$L^{Pretrain}$, T5-$L^{Pretrain}_{LWF+EPR}$ attains worse scores than both T5-B_{LWF} and T5-L_{LWF}, suggesting that generating phonemes does not inject proper knowledge to ease the rhyming generation process.

In addition, although prompt conditioning has been previously studied, Section A.2 of the appendix shows that, even if these conditions are not explicitly stated in the objective function, they correlate well with human-generated lyrics.

Multilingual Evaluation To address **Q5**, in Table 6 we report the results breakdown of the multilingual model in each language. Results indicate that learning rhyming across languages is quite complicated. Some languages are more complex, with Finnish, Norwegian, and Swedish having the

Model	Token.	Decod.	PPL ↓	R. P ↑	R. FP ↓	Mauve ↑	©	D-2	D-3	D-4
T5-L$^{\text{Pretrain}}$	T5	BS	**3.50**	11.36	2.31	0.0110	31.50	16.05	34.65	51.07
T5-L$^{\text{Pretrain}}$	T5	S+R	**3.50**	35.44	3.68	0.0087	8.37	**25.1**	**61.07**	85.50
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	BS	5.91	55.18	19.10	0.0120	32.35	7.61	25.73	50.38
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	S+R	5.91	**94.13**	**11.28**	0.0152	**5.13**	19.47	58.31	**87.78**
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	T5	BS	5.88	55.82	18.99	0.0238	26.09	18.91	41.76	60.51
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	T5	S+R	5.88	89.79	11.68	**0.0240**	8.52	22.36	59.66	87.32

Table 3: Results of various models trained on the English dataset. We follow the notation from 5.1 for the metrics and 5.3 for models and decoding. T5-B indicates the base model of T5, T5-L the large version, $*_{\text{LWF}}$ indicates that the model has been trained with last-word-first, while $*_{\text{LWF+EPR}}$ that the model has been trained on lyrics generation and phoneme generation tasks.

Model	Token.	Decod.	PPL ↓	R.P ↑	R.FP ↓	Mauve ↑	©	D-2	D-3	D-4
T5-B$^{\text{Rand}}_{\text{LWF}}$	Word	BS	6.79	15.38	**7.96**	0.0046	42.16	1.87	5.89	10.79
T5-B$^{\text{Rand}}_{\text{LWF}}$	Word	S+R	6.79	59.40	8.71	0.0056	**5.11**	8.20	33.65	58.43
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	BS	**5.91**	55.18	19.10	0.0120	32.35	7.61	25.73	50.38
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	S+R	**5.91**	**94.13**	11.28	**0.0152**	5.13	**19.47**	**58.31**	**87.78**

Table 4: Tokenizer comparison: Results of two models trained on the English dataset from scratch, one with a word tokenizer (Word) and the other with the default tokenizer (T5), as indicated in the column Token. We follow the notation from 5.1 for the metrics and 5.3 for models and decoding.

Model	Decod.	PPL ↓	R.P ↑	R.FP ↓
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	BS	5.88	55.82	18.99
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	S+R	5.88	**89.79**	11.68
T5-L$^{\text{Pretrain}}_{\text{LWF+EPR}}$	BS	**5.54**	36.43	**5.68**
T5-L$^{\text{Pretrain}}_{\text{LWF+EPR}}$	S+R	**5.54**	48.45	7.55

Table 5: Results of the comparison between Last Word First T5-L$^{\text{Pretrain}}_{\text{LWF}}$and the Last Word First + Ending Phonetic Representation T5-L$^{\text{Pretrain}}_{\text{LWF+EPR}}$. We follow the notation from 5.1 for the metrics and 5.3 for models and decoding.

worst scores and English, French, and Dutch having the best. This is mainly due to the nature of the pretraining data of mT5 (Xue et al., 2021b), where most text is in English followed, at a considerable distance but a similar amount among them, by Spanish, German, and French. Less frequently represented languages in mT5 and our dataset (see Table 18 in the Appendix) correspond to languages with lower scores.

We can also observe shared learning between languages from the same phonetic family. While Spanish is the most common in our dataset, the results are lower than in French or German, with phonetically Latin languages a lower score in general. The case of Finish is more remarkable since, not being Indo-European, it has no other supporting languages, giving the worst result. Even Danish and Croatian have better metrics with a lower representation in both datasets.

Worse results are expected when converting a model into a multilingual, especially in our case, where the dataset is significantly smaller. On average, model performance is poor (40.99), more than 40 points lower in terms of Rhyme Precision than the English-only model. While the model proved capable (to some extent) of deriving rhyming rules in English with text data only, it fails to do so when presented with data in several languages while showing some correlation based on phonetics. Indeed, rhyming is strictly tight to the way words are pronounced. Recently, phonetic representation of text has been proposed as interlingual (Leong and Whitenack, 2022) with encouraging results, and, in future work, it would be interesting to explore this idea also in the context of lyrics generation. Examples can be found in Appendix A.1

Language	R.P	R.FP	Support
English	**54.78**	9.96	1593
French	52.2	15.53	1355
Dutch	46.83	11.89	258
German	42.34	8.87	1386
Danish	39.25	6.49	53
Swedish	30.27	8.09	197
Norwegian	27.44	8.80	88
Portuguese	36.6	11.64	841
Italian	35.82	7.91	742
Spanish	32.62	8.90	2485
Croatian	35.61	10.81	90
Polish	34.94	9.23	338
Finnish	24.09	**8.09**	370
Micro AVG	40.99	10.19	

Table 6: Results of the multilingual model by fine-tuning mT5 with LWF technique. We used nucleus with 0.92 for top p as a sampling strategy

	Human	T5-$L_{LWF}^{Pretrain}$	T5-B_{LWF}^{Rand}
Correctness	2.61 ± 0.07	**2.41** ± 0.03	2.13 ± 0.07
Meaningfulness	2.43 ± 0.02	**2.19** ± 0.02	1.69 ± 0.06
Is-Human Rate	79.67 ± 2.43	**57.00** ± 1.81	23.43 ± 1.39

Table 7: Results on the human-evaluation tasks. Correctness: 3 is maximum, 1 is minimum; Meaningfulness: 3 is maximum, 1 is minimum; Is-Human Rate: rate at which annotators annotated a paragraph from the reference system as human.

6 Human Evaluation

Given the open-domain nature of text generation, automatic evaluation can often be inaccurate. In this section, we detail experiments where human annotators assess the quality of generated lyrics.

6.1 Setup

For more reliable information about the quality of the lyrics, we designed an annotation to measure the grammatical correctness and the meaningfulness of the produced text. Furthermore, we also asked annotators to guess whether a human wrote the presented text or not. The annotators were three English-speaking university students not directly associated with the work.

Specifically, we sampled 100 snippets from our test set and used their metadata (artist, rhyme schema, etc.) to generate as many texts from the two best models in Table 3, i.e., T5-B_{LWF}^{Rand} (line 4) and T5-$L_{LWF}^{Pretrain}$ (line 6). Hence, for each model, we have 200 items (100 paragraphs written by humans and 100 automatically created). We shuffled the items and asked three annotators to review them and assign scores to the following three categories:

1. **Correctness**: following Li et al. (2020), annotators had to rate lyrics with 3, grammatically correct; 2, readable but with some grammar mistakes; and 1, unreadable.

2. **Meaningfulness**: 3, meaningful text; 2, the text has some meaning but is expressed confusingly; and 1, the text has no meaning.

3. **Is it from a Human?**: annotators were asked whether the presented text was written by a human or automatically.

6.2 Results

In Table 7, we report the results of our human evaluation. As previously stated, T5-L_{LWF} attains results in terms of correctness (2.41) and meaningfulness (2.19) close to that assigned to lyrics written by humans, i.e., 2.61 and 2.43 for correctness and meaningfulness, respectively. On the opposite, T5-B_{LWF}, while still producing grammatically correct texts, their meaningfulness is much lower than human lyrics. Finally, T5-L_{LWF} generations are classified as written by humans 57% of the time, and, surprisingly, human products are recognised as such 79% of the times. We believe that this is due to the high percentage of lyrics without formal meaning.

7 Conclusion

We presented a new approach to enhancing rhyming in a PLM and a first attempt to scale lyrics generation across languages. Our framework provides a tool for the composer to automatically generate paragraphs given several desiderata, i.e., artist's style, song title, song genre, emotions, topics, and rhyme schema. The proposed method proved more effective than fine-tuning lyrics data and coherent lyrics without too much risk of copyright infringement. It produces meaningful and grammatically correct texts by reassembling human songs almost 6 out of 10 (according to human annotators). Furthermore, its accuracy in following the given rhyme schema is nearly 90%.

In future work, we aim to focus on biases that affect our model and approaches to mitigate them. Another area that requires further investigation is multilingualism, where performance still needs to be improved from the English one.

8 Limitations

One limitation of all analysed models is the need for more control over the language used by the model. Indeed, when specific genres are requested, e.g., rap and hip hop, the model may produce paragraphs interpreted as racist or insulting to certain minorities (e.g., women). This is a huge issue that has not been addressed systematically in the context of lyrics generation, causing, among other things, issues and concerns outside the scientific community.[15] In this paper we did not address this issue directly but instead proposed a study focused on rhyming and multilingualism. We intend to conduct this research shortly, focusing on mitigating biases and actively controlling the kind of language when generating lyrics.

Another limitation of the proposed approach lies in the algorithm checking whether two words rhyme. While designed for English, we adapted it to work for most European languages. However, each language may have its exceptions, which we might have neglected.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Michel Buffa, Elena Cabrio, Michael Fell, Fabien L. Gandon, Alain Giboin, Romain hennequin, Franck Michel, Johan Pauwels, Guillaume Pellerin, Maroua Tikat, and Marco Winckler. 2021. The WASABI dataset: cultural, lyrics and audio analysis metadata about 2 million popular commercially released songs. In *Eighteenth Extended Semantic Web Conference - Resources Track*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2778–2788, New York, NY, USA. Association for Computing Machinery.

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.

[15] https://www.theguardian.com/music/2022/aug/24/major-record-label-drops-offensive-ai-rapper-after-outcry-over-racial-stereotyping

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.

Colin Leong and Daniel Whitenack. 2022. Phone-ing it in: Towards flexible multi-modal language model training by phonetic representations of data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5306–5315, Dublin, Ireland. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Rigid formats controlled text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 742–751, Online. Association for Computational Linguistics.

Hisar Manurung. 2004. An evolutionary algorithm approach to poetry generation.

Nikola I. Nikolov, Eric Malmi, Curtis Northcutt, and Loreto Parisi. 2020. Rapformer: Conditional rap lyrics generation with denoising autoencoders. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 360–373, Dublin, Ireland. Association for Computational Linguistics.

Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. 2022. Poelm: A meter-and rhyme-controllable language model for unsupervised poetry generation. *arXiv preprint arXiv:2205.12206*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Yejin Choi, and Zaïd Harchaoui. 2021. MAUVE: human-machine divergence curves for evaluating open-ended text generation. *CoRR*, abs/2102.01454.

Raymond Queneau. 1961. *Cent mille milliards de poèmes*. Gallimard Series. Schoenhof's Foreign Books, Incorporated.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Naveen Ram, Tanay Gummadi, Rahul Bhethanabotla, Richard J Savery, and Gil Weinberg. 2021. Say what? collaborative pop lyric generation using multitask transfer learning. In *Proceedings of the 9th International Conference on Human-Agent Interaction*, HAI '21, page 165–173, New York, NY, USA. Association for Computing Machinery.

Yizhan Shao, Tong Shao, Minghao Wang, Peng Wang, and Jie Gao. 2021. *A Sentiment and Style Controllable Approach for Chinese Poetry Generation*, page 4784–4788. Association for Computing Machinery, New York, NY, USA.

Robert Weide et al. 1998. The carnegie mellon pronouncing dictionary. *release 0.6, www. cs. cmu. edu.*

Jörg Wöckener, Thomas Haider, Tristan Miller, The-Khang Nguyen, Thanh Tung Linh Nguyen, Minh Vu Pham, Jonas Belouadi, and Steffen Eger. 2021. End-to-end style-conditioned poetry generation: What does it take to learn from examples alone? In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 57–66, Punta Cana, Dominican Republic (online). Association for Computational Linguistics.

Lanqing Xue, Kaitao Song, Duocai Wu, Xu Tan, Nevin L. Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021a. 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.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021b. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

A Appendix

A.1 Examples of Generated Lyrics

Using two randomly selected prompts, we offer illustrations of the outputs generated by the models discussed in section 5. We must recognise the existence of slurs and offensive language within some of these generated outputs. It should be noted that the language models employed in this study can produce text that may encompass offensive or inappropriate language originating from the training data. We wish to emphasise that these outputs were automatically generated and were not deliberately included by the authors.

Model	Tokenizer	generation
T5-L	T5	RHYME_A<sentence_end> RHYME_B 'Cause I'm a fool for you<sentence_end> RHYME_C Look at me now<sentence_end> RHYME_D Oh, oh<sentence_end> RHYME_D Ooo, look at me now, now yea heah heaheh, hahaha, yeah<sentence_end> RHYME_D<sentence_end>
T5-B$^{\text{Rand}}_{\text{LWF}}$	Word	RHYME_A now<SEP>I'm ready now<sentence_end> RHYME_B down<SEP>Don't wanna put me down<sentence_end> RHYME_C around<SEP>'Cause you're not around<sentence_end> RHYME_D me<SEP>You should be ready for me<sentence_end> RHYME_D see<SEP>So you can see<sentence_end> RHYME_D be<SEP>What you gotta be<sentence_end>
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	RHYME_A me <SEP>I'm me <sentence_end> RHYME_B oh <SEP>Oh, oh <sentence_end> RHYME_C <unk><SEP><unk><unk><sentence_end> RHYME_D <unk><SEP>Yeah, yeah - - <sentence_end> RHYME_D me <SEP>And I know you're all for me (Yeah) <sentence_end> RHYME_D Yeah <SEP>(Oh) Yeah <sentence_end>
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	T5	RHYME_A now<SEP>Look at me now<sentence_end> RHYME_B you<SEP>I'm so in love with you<sentence_end> RHYME_C do<SEP>And I don't know what to do<sentence_end> RHYME_D me<SEP>But you've got a hold on me<sentence_end> RHYME_D see<SEP>You're the only one I see<sentence_end> RHYME_D be<SEP>That I wanna be<sentence_end>

Table 8: "Random examples generated with Beam Search decoding by the models from Tables 3 and 4 for the prompt: "<title>Look at Me Now<artist>Charlie Puth<schema>RHYME_A RHYME_B RHYME_C RHYME_D RHYME_D RHYME_D</s>".

Model	Tokenizer	generation
T5-L	T5	RHYME_A<sentence_end> RHYME_B<sentence_end> RHYME_C ooh<sentence_end> RHYME_D<sentence_end> RHYME_D....<sentence_end> RHYME_D..<sentence_end>
T5-B_{LWF}^{Rand}	Word	RHYME_A eyes<SEP>See in the dark, I was lookin' past the eyes<sentence_end> RHYME_B die<SEP>I used to lie like I would never die<sentence_end> RHYME_C you<SEP>And no good for you<sentence_end> RHYME_D me<SEP>Even though I really cared that you would let me<sentence_end> RHYME_D see<SEP>That's not you, now I see<sentence_end> RHYME_D be<SEP>Just remember how it'd always be<sentence_end>
T5-B_{LWF}^{Rand}	T5	RHYME_A shit <SEP>I want you been talking talking shit <sentence_end> RHYME_B yeah <SEP>Oh, oh, yeah <sentence_end> RHYME_C this <SEP>Well, I can feel this <sentence_end> RHYME_D say <SEP>Yeah, you, say <sentence_end> RHYME_D way <SEP>You really get this way (It's all) <sentence_end> RHYME_D away <SEP>But baby, baby get so hard (Baby get a yes) (Yeah now.. (Oh) yeah) just. <sentence_end>
T5-$L_{LWF}^{Pretrain}$	T5	RHYME_A love<SEP>I'm dancin', all I am is love<sentence_end> RHYME_B you<SEP>When I find you<sentence_end> RHYME_C me<SEP>But that's not enough, now you know me<sentence_end> RHYME_D go<SEP>Don't wanna let go<sentence_end> RHYME_D know<SEP>'Cause all that I see is what you need to know<sentence_end> RHYME_D so<SEP>Look at me now, and see that you love me so<sentence_end>

Table 9: Random examples generated with the Sampling and Reranking decoding stratgey by the models from Tables 3 and 4 for the prompt: "<title>Look at Me Now<artist>Charlie Puth<schema>RHYME_A RHYME_B RHYME_C RHYME_D RHYME_D RHYME_D</s>".

Model	Tokenizer	generation
T5-L	T5	RHYME_A<sentence_end> RHYME_B<sentence_end> RHYME_C<sentence_end> RHYME_D<sentence_end> RHYME_E<sentence_end> RHYME_F<sentence_end> RHYME_E..... sex laws<sentence_end> RHYME_G.<sentence_end>
T5-B$^{\text{Rand}}_{\text{LWF}}$	Word	RHYME_A eyes<SEP>When you close your eyes<sentence_end> RHYME_B you<SEP>I'm with you<sentence_end> RHYME_C time<SEP>It's been a long time<sentence_end> RHYME_D mind<SEP>But you're on my mind<sentence_end> RHYME_E me<SEP>And you don't know what to do with me<sentence_end> RHYME_F do<SEP>'Cause what you do<sentence_end> RHYME_E be<SEP>What you wanna be<sentence_end> RHYME_G you<SEP>So what are you?<sentence_end>
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	RHYME_A <unk> <SEP> <unk>, <unk> <sentence_end> RHYME_B <unk> <SEP> I've got to get to the <unk> - - <sentence_end> RHYME_C <unk> <SEP> And I can't get enough of my own own's <sentence_end> RHYME_D <unk> <SEP> Oh, oh - oh <sentence_end> RHYME_E yeah <SEP> Yeah, yeah <sentence_end> RHYME_F <unk> <SEP> Ooh - ooh <sentence_end> RHYME_E yea <SEP> Hey, yea <sentence_end> RHYME_G <unk> <SEP> Ah, ah - ah <sentence_end>
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	T5	RHYME_A law<SEP> Sex laws, sex law<sentence_end> RHYME_B laws<SEP> There's no escape from the sexx laws<sentence_end> RHYME_C man<SEP> I'm just a man<sentence_end> RHYME_D law<SEP> Sex laws..<sentence_end> RHYME_E me<SEP> You can't take it from me<sentence_end> RHYME_F you<SEP> 'Cause if you<sentence_end> RHYME_E be<SEP> Then you're gonna be<sentence_end> RHYME_G law<SEP> A victim of the Sexx law!<sentence_end>

Table 10: Random examples generated with Beam Search decoding by the models from Tables 3 and 4 for the prompt: "<title> Sexx Laws (Malibu remix)<artist> Beck<schema>RHYME_A RHYME_B RHYME_C RHYME_D RHYME_E RHYME_F RHYME_E RHYME_G</s>". Note: the tokenization of the prompt yeilds unkown tokens, thereby influincing the generation. Throughout the generation process, we did not force any of the models to skip <unk> tokens.

Model	Tokenizer	generation
T5-L	T5	RHYME_A Asked the judge for the cost<sentence_end> RHYME_B "Doctor can I make a small complaint?"<sentence_end> RHYME_C And he asked for ten of her dollars<sentence_end> RHYME_D "What's the point in making me suffer?"<unk><sentence_end> RHYME_E *ahem*<sentence_end> RHYME_F *yeah<sentence_end> RHYME_E ********uh-oh!<sentence_end> RHYME_G.<sentence_end>
T5-B$^{\text{Rand}}_{\text{LWF}}$	Word	RHYME_A live<SEP> I feel in my soul, I'm gonna live<sentence_end> RHYME_B rain<SEP> Flying round and around in the rain<sentence_end> RHYME_C away<SEP> The only thing I know when my troubles fly away<sentence_end> RHYME_D blue<SEP> In a red light, there's the sky like the blue<sentence_end> RHYME_E me<SEP> So don't throw out the oceans inside me<sentence_end> RHYME_F feel<SEP> Don'Cause I can've got the darkest thing about how I want the way I to feel<sentence_end> RHYME_E free<SEP> It'll be over myself, why are we in front row free?<sentence_end> RHYME_G say<SEP> There're no quiet life and there is nothing left to say<sentence_end>
T5-B$^{\text{Rand}}_{\text{LWF}}$	T5	RHYME_A room <SEP> The <unk> <unk> has <unk> into a room <sentence_end> RHYME_B again <SEP> I don't ever ever again <sentence_end> RHYME_C face <SEP> My soul is in my face <sentence_end> RHYME_D you <SEP> Well I never ever wanted time to let you <sentence_end> RHYME_E know <SEP> And I'll be back when I say I didn'didn didn oh so you thought I knew I would know <sentence_end> RHYME_F <unk> <SEP> For the second that I had <unk> <sentence_end> RHYME_E no <SEP> That I, I was gonna get to get, get a get from a better for a <unk> with a chance, a a no <sentence_end> RHYME_G away <SEP> Oh, Oh! <sentence_end>
T5-L$^{\text{Pretrain}}_{\text{LWF}}$	T5	RHYME_A you<SEP> It's like being caught out and not supposed to love you<sentence_end> RHYME_B it<SEP> And it'll be hard to say you're not in it<sentence_end> RHYME_C baby<SEP> But in my eyes, baby<sentence_end> RHYME_D truth<SEP> There'd be no use in escaping the truth<sentence_end> RHYME_E me<SEP> Just go with me, go along with I<sentence_end> RHYME_F man<SEP> Until the day that we met, man,,man<sentence_end> RHYME_E see<SEP> I will never forget the world I see<sentence_end> RHYME_G yeah<SEP> In my sight and in your mouth, yeah<sentence_end>

Table 11: Random examples generated with the Sampling and Reranking decoding stratgey by the models from Tables 3 and 4 for the prompt: "<title> Sexx Laws (Malibu remix)<artist> Beck<schema>RHYME_A RHYME_B RHYME_C RHYME_D RHYME_E RHYME_F RHYME_E RHYME_G</s>".

[Input] <title> Você Não Sabe De Nada <artist> Ivan Lins <lang> portuguese <schema> ABCCD <genre> MPB [Generation] Você não sabe de nada E não espera um des Vai fazer amor Depois de mais uma flor Para mim [Gold] Você não sabe de nada Se pensa que me convence As coisas que você diz São coisas que ninguém diz E isso lhe fica mal

Table 12: Multilingual generation output. The generation follows the schema even if it contains wrong words, like "des" which is just part of a word

[Input] <title> Je Suis Mes Pas <artist> Lucie Bernardoni <lang> french <schema> AAAB [Generation] Je suis mes pas Tout seul et sans voi Et moi tout bas Sans amour perdu [Gold] Le jour se lève, je brise le silence Je défie les apparences C'est le grand jour, la fin de l'innocence Il y a tant de choses à comprendre

Table 13: Multilingual output example. One of the rhymes is wrong ("voi") and moreover, it is not a real word but the start of a singular person of the verb "voir"

[Verse 1] I come from down in the valley Where, mister, when you're young They bring you up to do Like your daddy done Me and Mary we met in high school When she was just seventeen We'd drive out of this valley Down to where the fields were green [Chorus 1] We'd go down to the river And into the river we'd dive Oh, down to the river we'd ride ...

Table 14: Example from Genius.com data of the song "The River" by Bruce Springsteen.

A.2 Conditional generation

This subsection presents the results of our conditional generation experiment. Although the use of special tokens for conditional generation has been thoroughly studied (Chen et al., 2022), we included it to ensure our model's performance. We study how the Title and the genre affect the generated text. We used the sentence embedder $all\text{-}MiniLM\text{-}L6\text{-}v2$ to get the embeddings in both cases. We split the training data as $80/20$ for Train and Dev. The Test set was used to generate the synthetic data for all the models, so the values of the metrics over the Test should be considered as the reference of a human-like text. We also include the values on the Dev set to show the variance from one to another.

For the Title, we compute the dot product between the Title and the lyrics generated. Title correlation is the average of the dot product between the title embedding and the average of the verses embeddings. In the case of the genre, we trained multiple classifiers, SVM with different kernels and MPL with different structures, among others. We use Cross-Validation with a split $80/20$, obtaining the best model, SVM with linear kernel. Since the dataset has a huge imbalance, we randomly select a maximum of 700 data points for each of the 24 genres appearing in the test set. We use Accuracy for the genre.

The results show that conditioning generation works. The correlation between the Title and lyrics is close to the Test set, even higher in the case of the pretrained models. The genre shows a high heterogeneity among the songs from the same genre. This fact makes it complicated to obtain a good classifier. Still, the results show values very close to actual cases.

Model/Dataset	Decod.	Title correlation	Genre classification
Train	Human	0.3892	0.6276
Dev	Human	0.4063	0.3622
Test	Human	0.4149	0.2562
T5-B_{LWF}^{Rand}	BS	0.2951	0.2460
T5-B_{LWF}^{Rand}	S+R	0.2863	0.2620
T5-$L_{LWF}^{Pretrain}$	BS	0.5429	0.2566
T5-$L_{LWF}^{Pretrain}$	S+R	0.4092	0.2527

Table 15: Results of the analysis in conditional generation. Genre classification is measured with the accuracy and Title correlation with the dot product. We follow the notation from 5.3 for models and decoding. T5-B indicates the base model of T5, T5-L the large version, $*_{LWF}$ indicates that the model has been trained with last-word-first, while $*^{Rand}$ means that it has been trained from scratch.

A.3 Dataset statistics

More statistics on the datasets used in this work are presented below. First, in Table 16, we present the statistics of the English dataset in more detail. In Table 17, we do the same for the multilingual dataset. We finish with Table 18 where we can find the representation of each language in the multilingual dataset.

Split	# Examples	With Genres	With Emotions	With Topics	Avg. Tokens	Avg. Sentences	Avg. Sentence Length
Train	564 552	146 613 (25.97%)	27 118 (4.80%)	170 074 (30.13%)	46.92	5.84	8.03
Dev	3500	587 (16.77%)	62 (1.77%)	719 (20.54%)	74.90	9.77	7.67
Test	3500	804 (22.97%)	104 (2.97%)	893 (25.51%)	70.79	9.24	7.67

Table 16: Statistics for Train, Dev and Test splits of the Genius.com dataset.

Split	# Examples	With Genres	With Emotions	With Topics	Avg. Tokens	Avg. Sentences	Avg Sentence Length	Languages
Train	2 588 424	1 238 067 (47.83%)	58 280 (2.25%)	170 917 (6.60%)	40.19	6.99	5.75	13
Dev	10 000	4368 (43.68%)	228 (2.28%)	493 (4.93%)	39.80	6.79	5.87	13
Test	10 000	4541 (45.41%)	211 (2.11%)	493 (4.93%)	39.80	6.60	6.03	13

Table 17: Statistics for Train, Dev and Test splits of the multilingual dataset.

Language	# Examples
Spanish	672 264
English	387 600
German	361 272
French	360 812
Italian	228 386
Portuguese	199 367
Finnish	99 119
Polish	86 887
Dutch	70 605
Swedish	61 980
Norwegian	23 921
Croatian	21 260
Danish	14 951

Table 18: Statistics by language of the multilingual dataset.