ByGPT5: End-to-End Style-conditioned Poetry Generation with Token-free Language Models

Jonas Belouadi
jonas.belouadi@uni-bielefeld.de

Steffen Eger
steffen.eger@uni-bielefeld.de

Natural Language Learning Group (NLLG)
Faculty of Technology, Bielefeld University
nl2g.github.io

Abstract

State-of-the-art poetry generation systems are often complex. They either consist of task-specific model pipelines, incorporate prior knowledge in the form of manually created constraints, or both. In contrast, end-to-end models would not suffer from the overhead of having to model prior knowledge and could learn the nuances of poetry from data alone, reducing the degree of human supervision required. In this work, we investigate end-to-end poetry generation conditioned on styles such as rhyme, meter, and alliteration. We identify and address lack of training data and mismatching tokenization algorithms as possible limitations of past attempts. In particular, we successfully pre-train ByGPT5, a new token-free decoder-only language model, and fine-tune it on a large custom corpus of English and German quatrains annotated with our styles. We show that ByGPT5 outperforms other models such as mT5, ByT5, GPT-2 and ChatGPT, while also being more parameter efficient and performing favorably compared to humans. In addition, we analyze its runtime performance and demonstrate that it is not prone to memorization. We make our code, models, and datasets publicly available.[1]

1 Introduction

End-to-end fine-tuning of pre-trained language models like GPT-2 (Radford et al., 2019) or T5 (Raffel et al., 2020a) on downstream tasks has been an immensely popular training paradigm for text-generation in the last few years (Li et al., 2021). End-to-end models learn to complete a task by directly learning all steps, without intermediary algorithms such as hand-crafted rules or post-processing. This approach has proven to be highly effective on a wide range of problems such as dialog generation (Sun et al., 2022; Yang et al., 2021), summarization (Zhu et al., 2021; Zhong et al., 2021; Huang et al., 2021), and machine translation (Farinha et al., 2022; Tran et al., 2020). Nevertheless, all these applications have in common that they only concern themselves with the generation of prosaic texts. Generating *formal verse poetry* on the other hand, with strict constraints on *aesthetic style* such as rhyme scheme, meter and alliteration, remains a difficult problem. Attempts to employ end-to-end solutions in this context have so far been unsuccessful (Wöckener et al., 2021), with some authors even concluding that language models cannot pick up such constraints from data alone (Popescu-Belis et al., 2022). As a consequence, state-of-the-art poetry generation systems rely on human guidance by (i) injecting prior knowledge[2] in the form of hard-coded constraints to filter model outputs or modify probability distributions or (ii) breaking the whole process down into sophisticated task-specific model pipelines.

Tian and Peng (2022), for example, propose a sonnet generation framework with four distinct pipeline steps: content planning, rhyme pairs generation, polishing for aesthetics, and finally sketch-to-sonnet generation. Further, they incorporate prior knowledge such as pronunciation dictionaries, knowledge bases, and lexically constrained decoding. Similarly, Hopkins and Kiela (2017) use Weighted Finite State Transducers to monitor whether their poetry generation system meets metric constraints and roll

The sweet wild strain, the sudden start, A
Which shakes the perfumed altar's flame, B
To make its shrine a sacred name, B
And sing its praise in every heart. A

— ByGPT5

Figure 1: Generated quatrain with ABBA rhyme scheme, high amount of alliterations (green), and iambic meter, i.e., unstressed syllable (⏑) follows stressed syllable (—).

[1] https://github.com/potamides/uniformers

[2] We define incorporating prior knowledge as "Any form of influence on model decisions not learned by the model itself."

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7364–7381

back its state in case of a violation.

Such forms of human supervision lead to ramifications that an end-to-end solution would not face. Pipelines are susceptible to errors in early modules that propagate and are amplified in subsequent modules; an effect known as cascading of errors (Castro Ferreira et al., 2019). Similarly, incorporating prior knowledge depends on the cleverness and intent of the *modeler* and generally becomes more difficult when heterogeneous constraints are involved or the number of constraints increases (Garbacea and Mei, 2022). Furthermore, standard text-generation architectures do not lend themselves well for manually applying constraints. Due to the autoregressive generation of tokens from left to right, constraints at arbitrary positions cannot be implemented easily or only with additional trade-offs (Garbacea and Mei, 2022). For example, end rhymes, which come at the end of a verse, cannot be constrained in isolation due dependencies on previously generated tokens. A commonly applied work-around for this problem is to generate each verse in reverse (Lau et al., 2018; Jhamtani et al., 2019; Van de Cruys, 2020; Xue et al., 2021a).

In this work, we thus aim to reduce the amount of human supervision in poetry generation and explore viable end-to-end solutions. We hypothesize that failing to do so far has the following root causes: (i) lack of available training data. Poetry corpora labeled with aesthetic styles are few and far between and we speculate that they do not suffice to train a generalized model. (ii) Unfavorable tokenization algorithms. Aesthetic styles of poetry such as rhyme, meter, and alliteration are often expressed at the character-level while most available off-the-shelf pre-trained models operate at the subword-level (Kudo and Richardson, 2018). Xue et al. (2022) showed that character-level models (also known as *token-free* models) excel at other character-level tasks so we assume that they would perform similarly well at poetry generation. Our key contributions are as follows:

(i) We pre-train ByGPT5, to our knowledge the first decoder-only transformer for character-level language modeling.

(ii) We create QuaTrain, a large machine-labeled poetry corpus of quatrains in German and English.

(iii) By fine-tuning ByGPT5 on QuaTrain, we show that it learns character-level styles better than subword-based systems, such as GPT-2 and mT5, as well as other token-free models like ByT5, while being more parameter efficient and also faring well compared to humans.

(iv) We further demonstrate that ByGPT5 exhibits few memorization problems, understands poetry better than GPT-2 and ChatGPT, and also performs well on tasks that do not operate at the character-level.

2 Background

In formal verse poetry, poems have to follow strict patterns and rules of language which we term *styles*. Our goal is to train an end-to-end poetry generation system which learns to adhere to specified styles by itself. We refer to this as *style-conditioned* poetry generation. In our work, we focus on generating quatrains and conditioning on the following defining styles of formal verse poetry (cf. Figure 1):

Rhyme A rhyme is the repetition of the same or similar sounds in the final accented syllables of words, which must be preceded by differing consonants (Harmon et al., 2000). If all conditions are met, we speak of perfect rhymes, and if some of them are violated, for example, because the final sounds are different or the words are identical, we speak of imperfect rhymes. In a quatrain with ABAB rhyme scheme, the first and third line endings rhyme, as do the second and fourth lines.

Meter Meter refers to the rhythmic pattern within a verse. In modern poetry, this rhythm is usually accented-syllabic, that is, the succession of stressed (—) and unstressed syllables (⏑) occurs at regular intervals (Harmon et al., 2000). The rhythmic unit is also known as a foot and the meter of a verse can thus be described as a sequence of feet. In English poetry, common feet are iambic (⏑ —), trochaic (— ⏑), anapestic (⏑ ⏑ —), and dactylic (— ⏑ ⏑). For conditioning on meter, we consider all metric feet appearing in our datasets (cf. Appendix A).

Alliteration Harmon et al. (2000) define alliteration as the repetition of the same consonant sounds or any vowel sounds at the beginning of words or syllables that are close together in a verse. In formal verse, alliteration is secondary to rhyme and meter, follows less strict constraints, and is therefore not as easily classified. In this work, we thus consider the *level* of alliteration instead, which we classify as either *low*, *medium*, or *high* (cf. §5).

Name	Size	Params	Enc/Dec	Token-free
ByGPT5	small	73.5m	Decoder	✓
	base	139.2m	Decoder	✓
	medium	289.1m	Decoder	✓
GPT-2	base	124.4m	Decoder	✗
	medium	354.8m	Decoder	✗
ByT5	small	300m	Enc-Dec	✓
mT5	small	300m	Enc-Dec	✗

Table 1: Pre-trained models we fine-tune. ByGPT5 is a new model developed by us. The German GPT-2 model we use does not exist in medium size Minixhofer (2020), which is why we only use a base model there.

3 Models

We induce a range of end-to-end poetry generation systems for English and German by fine-tuning pre-trained transformer models (Vaswani et al., 2017). For conditioning on style, we consider two architectural variants—encoder-decoder transformers (Xue et al., 2021b, 2022) and decoder-only transformers (Radford et al., 2019; Brown et al., 2020). As explained in §1, we focus on token-free models, but also consider subword-level models for comparison. We do not experiment with models with more than 400 million parameters since they exceed the capacity of our available GPU resources.

Encoder-Decoder For encoder-decoder models, we initialize the encoder with a joint triple of rhyme scheme, meter, and, alliteration level and generate a quatrain with the decoder. We represent each style by a special token which we add to the model vocabulary. We use ByT5 (Xue et al., 2022), a token-free pre-trained encoder-decoder model, as a baseline. For comparison with subword-level approaches, we fine-tune mT5 (Xue et al., 2021b).

Decoder-only As the input for encoder-decoder models is a relatively short sequence of styles, this could lead to an underutilization of the encoder. We thus hypothesize that a decoder-only model, with styles supplied as a prompt string, would be better suited for our task. On the subword-level, multiple models, such as GPT-2 (Radford et al., 2019), are readily available. However, to our best knowledge, no such model exists at the character-level yet, which is why we train our own. Since our new model shares some similarities with the GPT family of models, but has its origin in ByT5 (see §4), we refer to it as *ByGPT5*. An overview of all models we use can be seen in Table 1.

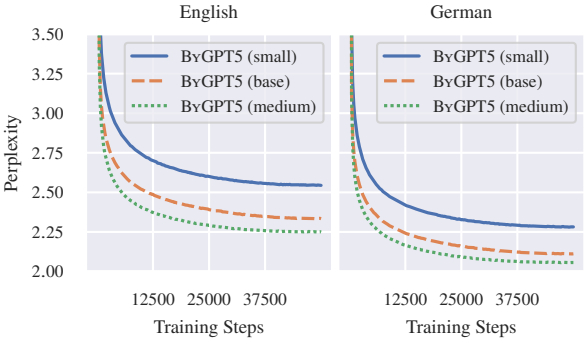

Figure 2: Perplexity on the training data when pre-training ByGPT5 for English and German.

4 ByGPT5

For pre-training our own token-free decoder-only model ByGPT5, we start by modifying the architecture of ByT5 and discard its encoder component. We then initialize the weights with the decoder of ByT5[3] to warm-start the training process (Rothe et al., 2020; Tang et al., 2022). We repeat this for the three smallest variants of ByT5. Because ByT5 has an asymmetrical architecture, the resulting models retain only 25% of its parameters. We refer to their model sizes as small, base, and medium.

As training data, we use OpenWebText2 (Gao et al., 2021) for English and cc100 (Conneau et al., 2020) for German. For hyper-parameters, we follow Radford et al. (2019) and Brown et al. (2020) and use Adam with a weight decay of 0.1, a batch size of 512, varying learning rates depending on model size, and train on a causal language modeling objective for 50k steps following Lester et al. (2021). We provide loss curves in Figure 2. As might be expected, the perplexity of larger models is generally lower than that of smaller counterparts.

5 Datasets

We collect a range of labeled and unlabeled datasets of English and German poetry (cf. Table 2). As shown, we were able to procure labeled corpora for rhyme and meter, but they are far too small to train a poetry generation system. Instead, we use the bigger unlabeled corpora, as training data, by labeling them automatically (Belouadi and Eger, 2023). To make full use of the data, we use not only real quatrains but also *pseudo-quatrains* (any consecutive sequence

[3]By referring to this process as weight initialization (rather than continued pre-training), we adopt the terminology of Rothe et al. (2020). This is intuitively sensible as we reuse only a subset of weights in a very different architecture. Consequently, our model experiences considerably less exposure to training data compared to its competitors during pre-training.

Dataset	Language	Verses	R	M	A
EPG64	English	1k	✓	✓	✗
Prosodic	English	2k	✗	✓	✗
FORB	English	1k	✓	✓	✗
Chicago	English	95k	✓	✗	✗
Anti-K	German	4k	✓	✓	✗
GRC	German	41k	✓	✗	✗
EPG	English	2.8m	✗	✗	✗
DLK	German	2.8m	✗	✗	✗
QuaTrain	English	2.7m*	✓ †	✓ †	✓ †
	German	5.9m*	✓ †	✓ †	✓ †

* Verses may occur in multiple pseudo-quatrains.
† Labels are obtained from classifiers.

Table 2: Available poetry datasets for rhyme (R), meter (M), and alliteration (A). Unlabeled corpora (middle) are orders of magnitude larger than labeled corpora (top), and we label them automatically (bottom). Further information can be found in Appendix A.

of four lines), amounting to over 660k quatrains for English and 1.4m for German. We refer to this new dataset as QuaTrain, with further statistics in Appendix A. In the following, we explain the labeling process for each style.

For automatically labeling rhyme and meter, we leverage the available labeled data and train a range of classifiers. We evaluate them on held-out gold data and subsequently use the best performing classifier for each style (cf. Appendix A). Meter classification is a multiclass classification problem with a single verse as input, while rhyme classification is a binary classification problem with two verses separated by a special token as input. We classify the meter of a quatrain by choosing the dominant meter among the verses[4], and the rhyme scheme by determining which verses rhyme and which do not.

As no readily available poetry datasets include labels for alliteration, we approach the problem in a different way. The quantification of the level of alliteration in a document is a long known research problem (Skinner, 1939; Leavitt, 1976; Blain, 1987; Benner, 2014). Let v_i be the atomic units of sound in verse v, Blain (1987) quantify alliteration as

$$\text{allit}(v) = \frac{\sum_{i=1}^{|v|} \sum_{j=i+1}^{|v|} \frac{\text{f}(v_i, v_j)}{j-i}}{\sum_{i=1}^{|v|} \sum_{j=i+1}^{|v|} \frac{1}{j-i}}, \quad (1)$$

where $\text{f}(\cdot)$ is a similarity function of two sounds; the default simply testing for equality. Intuitively, $\text{allit}(\cdot)$ counts alliterative sounds in a verse, applies a distance penalty, and normalizes the score to $[0, 1]$. To get a score for quatrains, we average the alliteration level of all verses. We consider initial phonemes of words, as well as all further stressed phonemes as atomic sound units v_i, and to determine phonemes and stress, we use a grapheme-to-phoneme conversion model (Zhu et al., 2022). Further, we conduct an internal study to determine several intensity thresholds based on a sample of quatrains. We classify the alliteration level of a quatrain as *low* if the score is below 0.05, *medium* if the score is below 0.1, and *high* if it is above that.

[4] Since in formal verse poetry, a meter is maintained throughout a poem, this procedure is meaningful.

6 Experiments

For fine-tuning, we use the same hyperparameters as in §4 for all models, but reduce the batch size to 128 (for efficiency reasons). We induce separate models for each language in QuaTrain and train for 10 epochs.[5] We conduct both automatic (§6.1) and human evaluation (§6.2). Examples of generated quatrains can be found in Appendix D.

6.1 Automatic Evaluation

For automatic evaluation, we select four common rhyme schemes (AABB, ABAB, ABBA, and ABCB), the most popular meters per language (iambus, trochee, anapest, and dactyl for English; iambus, trochee, and alexandrine for German), and all levels of alliteration to create 75 poems per model for each possible combination. To find out if styles are properly reflected in generated quatrains we reuse the classifiers from §5, i.e., we use them to classify the generated poems and see if the styles match. We define the following metrics:

Rhyme Score computes the recall of verses that should rhyme in a quatrain, as well as the recall of verses that should not and takes their arithmetic average.

Alliteration Score is 1 if a quatrain has the correct alliteration level, else 0.

Meter Score is the fraction of verses with correctly classified meters.

Coherence uses BERT for next sentence prediction (Devlin et al., 2019) to assess discourse relations of verses (Duari and Bhatnagar, 2021; Shi and Demberg, 2019). The score is the fraction of consecutive verse pairs that are correctly classified to come after one another.

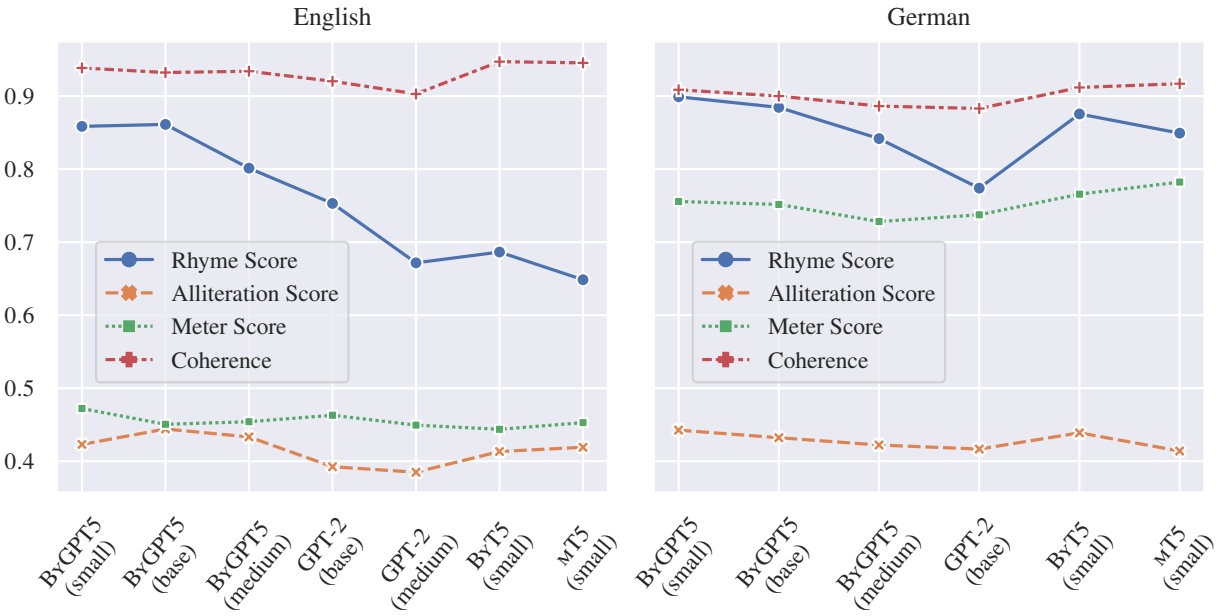

Figure 3: Automatic evaluation results for all models on English and German.

We provide the scores for each model averaged over all generated quatrains (2700 for German and 3600 for English) in Figure 3. Although all models manage to learn to follow aesthetic styles to some degree, there are noticeable score differences.

On rhyme, all ByGPT5 variants collectively outperform all GPT-2 models on both languages by 5%-20%. Similarly, ByT5 consistently outperforms mT5 by ~5%. This supports our theory that token-free models are better suited for character-level styles. Further, ByGPT5 (small) performs 2%-15% better than ByT5 (small) which means we can discard the encoder while still improving performance. Surprisingly though, base ByGPT5 and GPT-2 achieve higher scores than their medium variants. While this may initially suggest that larger decoders prioritize (meaningful) content, whereas smaller decoders focus on style, the high coherence seen across all models weakens this hypothesis. Instead, we speculate that this may be an overfitting problem. In particular, smaller models, up to base size, may be better suited for generating shorter texts such as quatrains. Another surprising finding is that ByT5 (small) performs worse than GPT-2 (base) on English. We investigate this further in §6.2.

In terms of meter, all models perform very similar to one another. Whereas ByGPT5 (small) performs best on English by a small margin, it is outperformed by mT5 (small) on German. This result is not surprising. Since meter is a syllable-level style, subword-level language models also manage to pick it up reasonably well (Lau et al., 2018). Interestingly though, on English the scores are much lower overall than on German. A reason for this may be that the occurrence of different meters is much more evenly distributed in German QuaTrain (cf. Table 6 in the appendix). While in German only about 60% of all meters are iambs, in English it is over 80%, making it difficult for models to learn other meters. We identify further reasons in §6.2.

Alliteration is the style that all models are the worst at. Our formulation of alliteration levels may make it difficult for models to pick up its semantics. Still, ByGPT5 (base) performs the best on English, and ByGPT5 (small) and ByT5 (small) perform the best on German, suggesting that token-free models have an advantage on this style.

In general, small and base ByGPT5 perform the best on all three styles in English and on two of them in German. It appears that they have an advantage in terms of tokenization algorithms (outperforming subword-level GPT-2 and mT5), architecture (outperforming encoder-decoder ByT5), and size (the medium variant performs worse).

6.2 Human Evaluation

To further validate the effectiveness of our models, we conduct a human evaluation campaign using *best-worst scaling* (BWS) as a means of annotation (Louviere et al., 2015). BWS is a variant of

[5]All models converge within the designated number of epochs, and throughout each epoch, the ordering of the systems remains consistent with our final results from automatic evaluation.

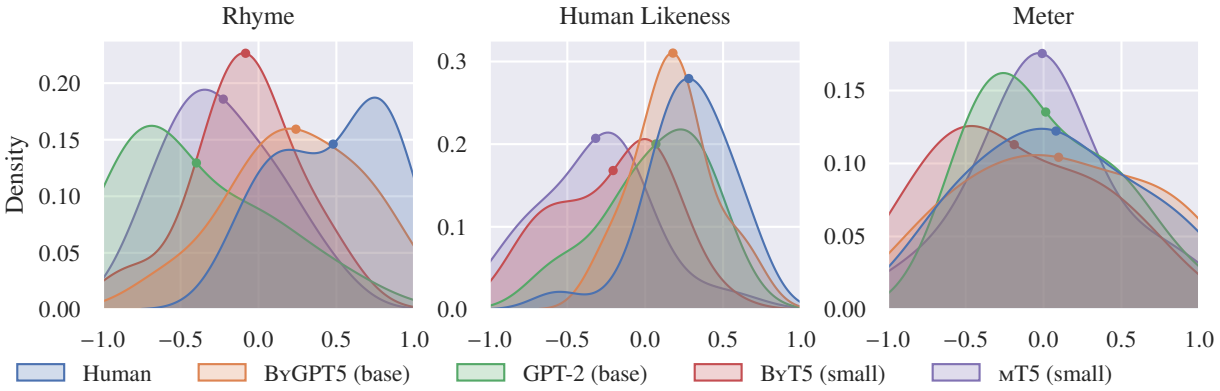

Figure 4: Distributions of BWS scores for rhyme, human likeness, and meter annotations through kernel density estimation. Scores range from -1 (very bad) to 1 (very good). The "•" markers denote expected values.

comparative annotation that produces high-quality results while keeping the number of required annotations low (Kiritchenko and Mohammad, 2016, 2017). Annotators are presented with tuples of n items (usually $n = 4$) and asked to identify the best and worst item based on a specified property. By subtracting the fraction of times an item is chosen as the best from the fraction of times it is chosen as the worst, real-valued scores ranging from -1 (bad) to 1 (good) can be obtained (Orme, 2009).

In our annotations, we consider three properties: rhyme, meter, and human likeness, i.e., the likelihood of a poem being written by a human. We exclude alliteration to reduce the workload on our annotators. Similarly, we exclusively evaluate on English (cf. Appendix B for a small-scale evaluation in German) and only consider the top-performing model within each model class based on the results of automatic evaluation. The models in question thus are ByGPT5 (base), GPT-2 (base), ByT5 (small), and mT5 (small). Furthermore, we only choose from three rhyme schemes (AABB, ABAB and ABBA), two meters (iambus and trochee), and one level of alliteration (medium), and create four poems per system for each possible combination. In addition, we also randomly sample human quatrains from our datasets that match the constraints and create 120 4-tuples from the combined set of quatrains.

Four annotators then annotate rhyme and human likeness, whereas meter is evaluated by a single expert annotator only (cf. Appendix B). Since we have multiple annotators working on rhyme and human likeness we use the *split-half reliability* (SHR) measure (Kiritchenko and Mohammad, 2017) to assess their consistency. SHR is calculated by splitting the annotations into two sets, computing scores for each set, and then computing their Spearman rank correlation coefficient.

Figure 4 displays a kernel density estimate for each property, with distributions shifted to the right indicating better performance. On rhymes, we obtain an SHR of $\rho = 0.77$ which demonstrates a high agreement between annotators. Human rhymes are ranked the highest overall, whereas ByGPT5 comes in as a close second, followed by ByT5. mT5 and GPT-2 perform the worst. This is a bit different from our findings during automatic evaluation where GPT-2 (base) was ranked higher than ByT5 (small) on English. An analysis of GPT-2 generated quatrains revealed a predominance of imperfect rhymes as a likely cause. As our rhyme classifier is trained on binary labels it is unable to detect this, but human annotators perceive this kind of rhyme as worse.

With $\rho = 0.54$, the SRH of human likeness is noticeably lower than for rhyme. On the one hand, this suggests that this task could be more subjective; on the other hand, the generated quatrains must be sufficiently human-like for subjectivity to be a factor. Indeed, although humans rank higher than ByGPT5 which in turn ranks higher than GPT-2, they all perform noticeably more similar than for rhyme. Nonetheless, we can observe that ByT5, and especially mT5 rank a bit lower. Both models were pre-trained on corrupted spans and have thus never seen truly natural text during pre-training (Zhu et al., 2022; Raffel et al., 2020b; Lester et al., 2021) which we believe could be a possible cause.

The distributions for meter have large variances for all models, and also humans. This is surprising, as it implies that our annotator does not think that humans are superior, even though the automatic evaluation of English models was not particularly

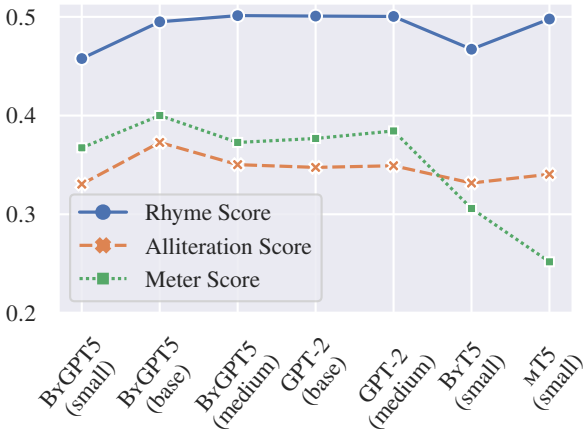

Figure 5: Automatic evaluation of low-resource models.

	Memorization		Emotion
Model	English	German	German
ByGPT5 (small)	0.0%	0.0%	0.676
ByGPT5 (base)	0.0%	0.04%	0.680
ByGPT5 (medium)	0.0%	0.81%	0.659
GPT-2 (base)	0.39%	**1.81%**	0.676
GPT-2 (medium)	**3.64%**	—	—
ByT5	0.0%	0.0%	0.691
mT5	0.0%	0.0%	**0.696**

Table 3: Extractive memorization rates (English & German) and recall on emotion generation (German).

strong on meter. We hypothesize that even among real English poets, there is a significant amount of poetry that does not strictly adhere to metric constraints, so language models only learn to follow them freely as well. Nonetheless, we can still see that, among models, ByGPT5 is rated highest, followed by GPT-2, mT5, and lastly ByT5, reflecting our findings of automatic evaluation. Interestingly, ByGPT5 also ranks higher than humans.

Overall, our human evaluation suggests that ByGPT5 performs best across all properties evaluated, which is consistent with our automatic evaluation. Moreover, ByGPT5 has shown the ability to perform at a level comparable to humans, and even surpass human performance in the meter property.

7 Analysis

We continue with a deeper analysis and look into low-resource training (§7.1), quantify memorization (§7.2), evaluate the performance of token-free models on non-character-based, high-level tasks (§7.3), introspect the models' understanding of style when predicting tokens (§7.4), and compare ByGPT5 with ChatGPT (§7.5).

7.1 Low-resource Training

We hypothesized that a large training corpus is an important factor in successfully training an end-to-end poetry generation system. We examine this hypothesis by selecting a 5% subset of English Quatrain (33k quatrains) and re-training our models using the same hyperparameters as in §6. Figure 5 shows how well these new low-resource models adhere to style constraints, similar to the automatic evaluation of full training in Figure 3.

Compared to full training, all low-resource models are noticeably worse at adhering to style. Specifically, the performance drops by 15%-40% for rhyme, 5%-20% for meter, and 5%-10% for alliteration. In addition, the overall performance difference between all models is much smaller than in §6.1. While these findings support our hypothesis that training on large datasets is essential, they also reveal that ByGPT5 demonstrates the largest improvements as the dataset size increases (cf. Figure 3). We therefore theorize that larger datasets lead to substantial performance gains in poetry generation *only* when coupled with architectures that excel at character-level tasks.

Nevertheless, even in low-resource scenarios, ByGPT5 (base) outperforms the other models in all categories except rhyme, where a few other systems perform similarly. This suggests that the conclusions drawn in §6.1 hold to some extent even when the available training data is limited.

7.2 Extractive Memorization

A common problem of language models, known as extractive memorization (EM), is generating verbatim copies from the training data during inference (Carlini et al., 2022; Raunak and Menezes, 2022; Meehan et al., 2020). According to Carlini et al. (2022) EM occurs when a language model's continuation of a string is part of the data it was trained on. Since the inputs to our language models are strings of style, this formulation lends itself well to our case: to detect memorization we simply have to check if generated poems appear in Quatrain. In Table 3, we compute the EM rates of the quatrains generated in §6.1. To account for negligible variations, we do not compare raw strings, but calculate the Ratcliff-Obershelp similarity (Ratcliff and Metzener, 1988), and assume that two strings are equivalent if their similarity exceeds 0.7.

As can be seen, GPT-2 suffers from memorization the most. On English, over 3% of all outputs of GPT-2 (medium) are copied. While ByGPT5 also copies quatrains to an extent, it is much less affected in comparison. On English, ByGPT5 (medium) does not copy anything and on German only 0.81% of all outputs. As a general trend, we can see that bigger models tend to copy more data than smaller ones—a finding shared by others (Carlini et al., 2022). Interestingly, the encoder-decoder models ByT5 and mT5 do not seem to be affected by this problem at all, most likely because styles are not used as a prompt, but are fed into the encoder separately.

7.3 Higher-level Styles

We also explore how token-free models perform on higher-level styles which are not character- or subword-level phenomena. In particular, we focus on emotion using Po-Emo (Haider et al., 2020), a dataset of eight aesthetic emotions in poetry (cf. Appendix A). By conditioning our models on these emotions, we can assess their ability to understand and depict emotion in poetry.

As in §5, we leverage automatic labeling. To that end, we train a classifier on German Po-Emo as in Haider et al. (2020) and reproduce the results. We then classify emotions in German QuaTrain and retrain our systems by conditioning them on all emotions in a quatrain. To evaluate how well the models can discriminate emotions, we condition them on every possible tuple of two distinct emotions and generate 100 poems each (2800 in total), and report the recall of correctly classified emotions. The results in Table 3 show that encoder-decoder models score highest, with mT5 performing best. During training, conditioning inputs can be long and variable in size, a scenario for which encoder-decoders may be better suited. Still, decoder-only models are not far behind. Especially ByGPT5 fares well against GPT-2, suggesting that token-free models are also competitive on higher-level tasks.

7.4 Token-level Attributions

To introspect the decision-making processes of our models, we visualize their token-level attributions when generating a quatrain. Token-level attributions explain to which degree each token in the input is involved in determining the next output token of a model, allowing us to reason about what a model has learned. To this end, Ferrando et al. (2022) decompose the attention blocks of transformers into a sum of vectors and define a new measure for visualizing token-to-token interactions based on the distance of each vector to the output (Kobayashi et al., 2021). We apply this measure on generative language models and visualize token-level attributions for ByGPT5 and GPT-2 when generating the last syllable in a quatrain. Since we have observed reoccurring trends, we use a single visualization in Figure 6 as a leading example. We provide additional examples in German in Appendix C.

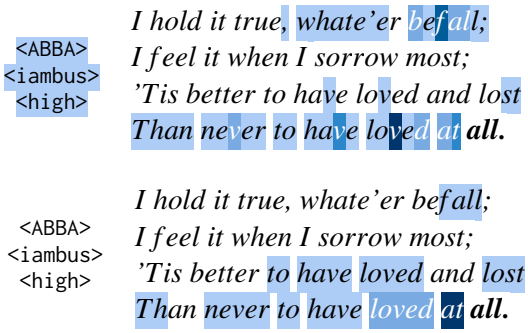

Figure 6: A famous stanza by Tennyson (1850) with visualized attention from ByGPT5 (top) and GPT-2 (bottom) when generating the last syllable.

We can see that ByGPT5 puts a big emphasis on the current verse, as well as the styles it was conditioned on. Further, possibly in response to the ABBA rhyme scheme, it also heavily stresses the ending of the first verse. Since the model also places a moderate amount of attention on the last consonants in verse three, it also seems to be aware of which sounds it should *not* generate in order maintain the rhyme scheme. Interestingly, it heavily emphasizes the letter *v* in the last two verses. We assume that this corresponds to what ByGPT5 understands by alliteration, in which case it would not have understood well at which position in a word the same sounds must occur.

Unlike ByGPT5, GPT-2 does not put any visible emphasis on input style tokens, which suggests that it does not understand how to handle them very well. Nevertheless it stresses similar aspects to ByGPT5, although, due to the subword vocabulary, at a different level of granularity.

7.5 Comparison with ChatGPT

ChatGPT (OpenAI, 2022) is a conversational large language model which specializes in dialogue. It has attracted attention for its detailed and expressive answers, raising the question of how well it performs in generating poetry. In a small-scale study, we thus ask ChatGPT to generate quatrains with

various rhyme schemes (AABB, ABAB, ABBA, and ABCB) using its web interface,[6] and similarly generate poems using ByGPT5. We then construct random pairs of quatrains of each model and want to find out which poem adheres better to rhyme constraints. Since we know the quatrains of ChatGPT beforehand, we use our rhyme scorer of §6.1 for unbiased scoring. Only in 15% of cases does our scorer prefer poems of ChatGPT over ByGPT5. Manual investigation showed that ChatGPT tends to generate rhymes at arbitrary positions, rather than adhering to specified rhyme schemes, even when giving examples in the prompt. Our verdict is that ChatGPT is a viable approach for poetry generation but not *style-conditioned* poetry generation.

8 Related Work

As indicated in §1, competing poetry generation systems usually consist of model pipelines and/or inject prior knowledge. Zhang and Lapata (2014), for example, propose a system for modeling quatrains consisting of three components: one model encodes previous verses, a second one reduces them to a single context vector, and a third one generates verses, one at a time. During decoding, phrases which violate style constraints are discarded. Similarly, Deep-speare (Lau et al., 2018) consists of a language model that generates a set of sample verses in reverse order, a model that reinitiates sampling as long as rhyme constraints are not met, and a final model that ranks the samples according to how well they adhere to iambic pentameter. Van de Cruys (2020) also induce prior knowledge into a generic language model, but they do it by modifying output probability distributions directly. Jhamtani et al. (2019) put their focus on actually *learning* rhyme and train a sonnet and limerick generator through adversarial training. The model is hierarchical, i.e., it first generates a sequence of line endings which are subsequently completed in reverse. While the model manages to learn the meaning of rhyme to an extent, the authors still filter outputs using pronunciation dictionaries. More in line with our research, Hopkins and Kiela (2017) train a model on the phonetic representation of poetry using the International Phonetic Alphabet (IPA) as a character-level vocabulary. During inference, a second model translates sounds back to human-readable text. Although promising, the model did not generalize well, and an additional model enforces rhythmic constraints in their final approach. Ormazabal et al. (2022) target *unsupervised* poetry generation by training a system on prosaic text only and conditioning it on structural information such as line endings and syllable counts. During inference, the system can generate poetry when conditioned on rhyming line endings and metric syllable counts. Nonetheless, since this information must be crafted manually, it is still supervised in a slightly different sense. In contrast, our model is supervised during training as it requires poetry to learn from, but unsupervised during inference as it is able to independently incorporate poetic elements.

[6]After experimenting with prompt engineering and in-context learning, we settled on a straightforward template: "Generate a quatrain with <pattern> rhyme scheme."

9 Conclusion

In this work, we implement end-to-end style-conditioned poetry generation systems for quatrains in English and German. Unlike other work, our systems are able to generate poetry without the need for human supervision, except for the use of poetic training data. In particular, we present ByGPT5, a novel token-free decoder-only language model, and show that fine-tuning it on a custom poetry corpus outperforms other models, such as GPT-2, mT5, and ByT5, on average, while also performing favorably against human poets in our constrained setting. Our key findings are that (i) tokenization algorithms matter, i.e., token-free language models generally perform better at generating character-level styles than subword-level transformers, and (ii) large datasets are crucial for successful training. We further show that bigger models do not necessarily perform better and that decoder-only architectures work best, i.e., we can discard the encoder of ByT5 (75% of parameters) while still improving downstream performance. We also demonstrate that token-free transformers perform competitively on tasks not tied to character-level styles, and are less susceptible to memorization of the training dataset. In addition, we conduct a visual analysis of token-level attributions during quatrain generation that is consistent with human perception of styles.

In future work, we want to to extend our system to other poetic forms such as sonnets, limericks, or villanelles.

10 Limitations

A well-known shortcoming of transformers is the computational complexity in self-attention lay-

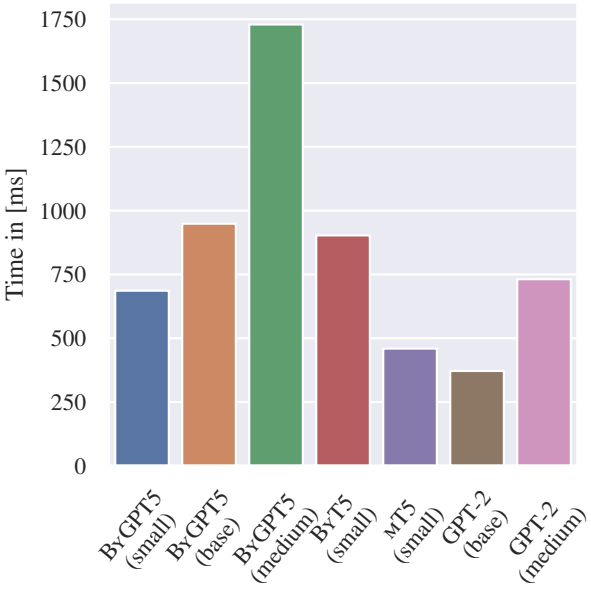

Figure 7: Inference times for generating a single quatrain (with 177 characters) on an A6000 GPU.

ers (Vaswani et al., 2017). Since the number of required calculations grows quadratically with the length of the input, transformers become prohibitively slow on very long sequences. An unfortunate side effect of processing inputs at the character-level is that internal sequences become much longer, so token-free transformers run into these efficiency problems much earlier than subword-based models. Figure 7 illustrates this problem by contrasting the runtime of all poetry generation systems when generating a single quatrain. Even ByGPT5 (small), the smallest model in terms of number of parameters (cf. Table 1) and the fastest token-free transformer, is only marginally faster than GPT-2 (medium), which is almost five times larger. Tay et al. (2022) propose a solution to this problem for transformer encoder blocks by applying a neural pooling operation over input embeddings before feeding them into the model, which could be extended to decoder blocks in future work. Alternatively, Libovický et al. (2022) propose a two-stage decoding architecture in which the transformer decoder operates on character blocks that an additional LSTM model (Hochreiter and Schmidhuber, 1997) decodes into individual characters.

Another shortcoming is that our poetry generation systems can only generate a single poetic form, i.e., quatrains. In general, poetry is a very diverse form of language and stanzas can be of arbitrary length, so this is a serious limitation. In future work, we thus plan to extend our implementation of style-conditioning to variable length poems. In particular, one could encode a rhyme scheme not as a single special token, but as an arbitrary series of letters indicating which verses rhyme with each other. Alternatively, our current systems could be used to generate longer stanzas through a sliding window approach, i.e., generating one verse at a time with the last three verse as context.

Further, our human evaluation has limitations due to its relatively small scope. We only have a limited number of annotators and only consider a subset of all style combinations. Nevertheless, we have achieved moderately high to high agreement on all tasks, and we have an additional human evaluation of German poetry in Appendix B, which points to the same conclusion.

Lastly, QuaTrain is limited in that it consists of pseudo-quatrains, which are not real quatrains and often have missing contexts. Nonetheless, as can be seen in Appendix D, models trained on QuaTrain are still able to generate meaningful poetry. In future work, we plan to improve the quality of our dataset by obtaining real quatrains from additional sources such as the Eighteenth-Century Poetry Archive (Huber, 2022).

Acknowledgments

We thank all reviewers for their valuable feedback, hard work, and time. We also thank all annotators, without whom this work would not have been possible. The last author was supported by DFG grant EG 375/5–1. We further gratefully thank the BMBF for its support via the grant Metrics4NLG.

References

Jonas Belouadi and Steffen Eger. 2023. UScore: An effective approach to fully unsupervised evaluation metrics for machine translation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 358–374, Dubrovnik, Croatia. Association for Computational Linguistics.

Drayton C. Benner. 2014. 'The Sounds of the Psalter: Computational Analysis of Soundplay'. *Literary and Linguistic Computing*, 29(3):361–378.

Derrel R. Blain. 1987. A mathematical model for alliteration. *Style*, 21(4):607–625.

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.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2022. Quantifying memorization across neural language models.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. 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 552–562, Hong Kong, China. Association for Computational Linguistics.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Javier de la Rosa, Álvaro Pérez, Mirella de Sisto, Laura Hernández, Aitor Díaz, Salvador Ros, and Elena González-Blanco. 2021. Transformers analyzing poetry: multilingual metrical pattern prediction with transfomer-based language models. *Neural Computing and Applications*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Swagata Duari and Vasudha Bhatnagar. 2021. Ffcd: A fast-and-frugal coherence detection method. *IEEE Access*, PP:1–1.

Ana C Farinha, M. Amin Farajian, Marianna Buchicchio, Patrick Fernandes, JosÃ© G. C. de Souza, Helena Moniz, and AndrÃ© F. T. Martins. 2022. Findings of the wmt 2022 shared task on chat translation. In *Proceedings of the Seventh Conference on Machine Translation*, pages 724–743, Abu Dhabi. Association for Computational Linguistics.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling.

Cristina Garbacea and Qiaozhu Mei. 2022. Why is constrained neural language generation particularly challenging?

Thomas Haider. 2021. Metrical tagging in the wild: Building and annotating poetry corpora with rhythmic features. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3715–3725, Online. Association for Computational Linguistics.

Thomas Haider, Steffen Eger, Evgeny Kim, Roman Klinger, and Winfried Menninghaus. 2020. PO-EMO: Conceptualization, annotation, and modeling of aesthetic emotions in German and English poetry. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1652–1663, Marseille, France. European Language Resources Association.

Thomas Haider and Jonas Kuhn. 2018. Supervised rhyme detection with Siamese recurrent networks. In *Proceedings of the Second Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 81–86, Santa Fe, New Mexico. Association for Computational Linguistics.

W. Harmon, C.H. Holman, and W.F. Thrall. 2000. *A Handbook to Literature*. Handbook to Literature Series. Prentice Hall.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

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.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Alexander Huber. 2022. Eighteenth-century poetry archive. [Online; accessed 15-December-2022].

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.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 465–470, Vancouver, Canada. Association for Computational Linguistics.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing reliable fine-grained sentiment associations by crowdsourcing and best–worst scaling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 811–817, San Diego, California. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4547–4568, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

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.

Jay A. Leavitt. 1976. On the measurement of alliteration in poetry. *Computers and the Humanities*, 10(6):333–342.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language model for text generation: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4492–4499. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Jindřich Libovický, Helmut Schmid, and Alexander Fraser. 2022. Why don't people use character-level machine translation? In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2470–2485, Dublin, Ireland. Association for Computational Linguistics.

Jordan J. Louviere, Terry N. Flynn, and A. A. J. Marley. 2015. *Best-Worst Scaling: Theory, Methods and Applications*. Cambridge University Press.

Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. 2020. A three sample hypothesis test for evaluating generative models. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3546–3556. PMLR.

Benjamin Minixhofer. 2020. GerPT2: German large and small versions of GPT2.

Eduard Mörike. 1832. *Maler Nolten. Novelle in zwei Teilen. 2. Teil*. G. J. Göschen'sche Verlagshandlung, Stuttgart.

OpenAI. 2022. ChatGPT: Optimizing language models for dialogue. [Online; accessed 17-December-2022].

Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. 2022. PoeLM: A meter- and rhyme-controllable language model for unsupervised poetry generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3655–3670, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Bryan K. Orme. 2009. Maxdiff analysis : Simple counting , individual-level logit , and hb. Sawtooth Software, Inc.

Andrei Popescu-Belis, Àlex Atrio, Valentin Minder, Aris Xanthos, Gabriel Luthier, Simon Mattei, and Antonio Rodriguez. 2022. Constrained language models for interactive poem generation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3519–3529, Marseille, France. European Language Resources Association.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

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

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

John W. Ratcliff and David E. Metzener. 1988. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, page 46.

Vikas Raunak and Arul Menezes. 2022. Finding memo: Extractive memorization in constrained sequence generation tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5153–5162, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Wei Shi and Vera Demberg. 2019. Next sentence prediction helps implicit discourse relation classification within and across domains. 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 5790–5796, Hong Kong, China. Association for Computational Linguistics.

Burrhus F. Skinner. 1939. The alliteration in shakespeare's sonnets: a study in literary behavior. *Psychological Record*.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2022. BORT: Back and denoising reconstruction for end-to-end task-oriented dialog. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2156–2170, Seattle, United States. Association for Computational Linguistics.

Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Mvp: Multi-task supervised pre-training for natural language generation.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Prakash Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Alfred Tennyson. 1850. *In Memoriam A.H.H.* Edward Moxon and Co., London.

Yufei Tian and Nanyun Peng. 2022. Zero-shot sonnet generation with discourse-level planning and aesthetics features. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3587–3597, Seattle, United States. Association for Computational Linguistics.

Chau Tran, Yuqing Tang, Xian Li, and Jiatao Gu. 2020. Cross-lingual retrieval for iterative self-supervised training. In *Advances in Neural Information Processing Systems*, volume 33, pages 2207–2219. Curran Associates, Inc.

Herbert F. Tucker. 2011. Poetic data and the news from poems: A "for better for verse" memoir. *Victorian Poetry*, 49(2):267–281.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

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.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14230–14238.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680, Doha, Qatar. Association for Computational Linguistics.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921, Online. Association for Computational Linguistics.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. 2021. MediaSum: A large-scale media interview dataset for dialogue summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5927–5934, Online. Association for Computational Linguistics.

Jian Zhu, Cong Zhang, and David Jurgens. 2022. ByT5 model for massively multilingual grapheme-to-phoneme conversion. In *Proc. Interspeech 2022*, pages 446–450.

Meter	Symbol
iambus	⏑ –
trochee	– ⏑
amphibrach	⏑ – ⏑
anapest	⏑ ⏑ –
dactyl	– ⏑ ⏑
alexandrine	⏑ – ⏑ – ⏑ – ‖ ⏑ – ⏑ – ⏑ –

Table 4: Meters in our dataset we consider for our experiments. An alexandrine consists of iambic feet with a caesura after the sixth syllable.

Model	Rhyme	Meter
CANINE-C	**98.05**	**58.49**
XLM-R	97.22	54.65
MBERT	97.17	49.01

Table 5: F1-Score on classifying rhyme and meter.

A Additional poetry corpus statistics

The poetry corpora we collect are English Project Gutenberg (EPG) and Deutsches Lyrik Korpus (DLK) (Haider, 2021) for unlabeled poetry, and PROSODIC[7], Chicago Rhyme Corpus (CHICAGO)[8], For-better-for-verse (FORB) (Tucker, 2011), German Rhyme Corpus (GRC) (Haider and Kuhn, 2018), as well as EPG64 and ANTI-K (Haider, 2021) for labeled poetry. We map meters which appear less than 25 times in our labeled corpora to the special label *other*. The final list of meters we consider can be found in Table 4.

The performance of the meter and rhyme classifiers we train can be seen in Table 5. For each style, we perform a 90/5/5 train-valid-test split and fine-tune a range of encoder-only transformers with classification heads jointly on both languages, as this improves performance (de la Rosa et al., 2021; Haider and Kuhn, 2018). We test subword-level MBERT and XLM-R, as well as character-level CANINE-C (Clark et al., 2022). Since character-level CANINE-C outperforms both MBERT and XLM-R on both tasks, we use it as our final classifier.

During automatic labeling, when the rhyme scheme cannot be clearly determined (e.g., according to the classifier the first verse rhymes with the second, the second with the third but the first and the third do not rhyme) or no dominant meter exists, we discard the quatrain. Frequencies of automatic labels inside QUATRAIN can be seen in Table 6. By limiting QUATRAIN to quatrains, we not only reduce the burden on poetry generation systems by considering only a single poetic form, but also ease the labeling process. As the length of poems increases, the number of verse pairs that have to be classified for rhyme grows super-exponentially, which quickly becomes intractable.

The eight emotions in PO-EMO we train our classifier on are *beauty / joy*, *sadness*, *uneasiness*, *vitality*, *awe / sublime*, *suspense*, *humor*, and *annoyance*. Since an additional emotion, *nostalgia*, almost never occurs, we follow Haider et al. (2020) and omit it from our experiments.

B Annotator Demographics

Our human annotators are fluent in English at a C1 or higher level according to the Common European Framework of Reference for Languages (CEFR). The annotators for rhyme and human likeness are one male faculty member, two male PhD students, one female undergraduate student, and three female volunteers from other departments, amounting to seven distinct annotators who are all proficient in English but may have limited knowledge of poetry. To get four sets of annotations per style, we have one PhD student annotate both styles. For meter, we hire a professional female teacher who is specialized in English and music.

Since none of our annotators speak English as a native language, we have one PhD student and one faculty member conduct a small-scale comparative study in their native language, German, annotating 30 BWS tuples for rhyme and human likeness. The results in Table 7 confirm the trends we saw in human-evaluation in §6.2. In terms of rhyme, humans perform best, followed by BYGPT5, BYT5, MT5, and finally GPT-2. On human likeness BYGPT5 is outperformed by humans but performs similar to GPT-2. The encoder-decoder models MT5 and BYT5 perform the worst, likely for similar reasons outlined in §6.2.

C German Token-level Attributions

Figure 8 shows token-level attribution scores for German quatrains. By and large, we observe the same trends as in §7.4 on English, i.e., GPT-2 places less attention on style and the emphasized parts of the text are less granular.

D Example Quatrains

In Table 8 we list additional example quatrains in German and English, generated with BYGPT5 (base).

[7] https://github.com/quadrismegistus/prosodic
[8] https://github.com/sravanareddy/rhymedata

Language	Rhyme label	Rhyme freq.	Meter label	Meter freq.	Alliteration label	Alliteration freq.
German	ABCD	19.73%	iambus	61.66%	low	50.95%
	ABAB	15.60%	alexandrine	18.06%	medium	39.23%
	AABB	13.07%	trochee	17.05%	high	9.81%
English	AABB	19.17%	iambus	83.96%	medium	54.92%
	ABCD	16.40%	anapest	7.61%	low	28.94%
	ABBC	13.07%	trochee	4.13%	high	16.14%

Table 6: Distribution of alliteration levels, as well as most frequent meters and rhyme schemes in QuaTrain.

Model	Rhyme	Human Likeness
Human	**0.84**	**0.89**
ByGPT5	0.57	0.55
GPT-2	0.35	0.55
ByT5	0.45	0.19
mT5	0.28	0.32

Table 7: Min-max normalized and averaged BWS scores annotated by two native German speakers. The SHR is $\rho = 0.83$ for rhyme and $\rho = 0.61$ for human likeness.

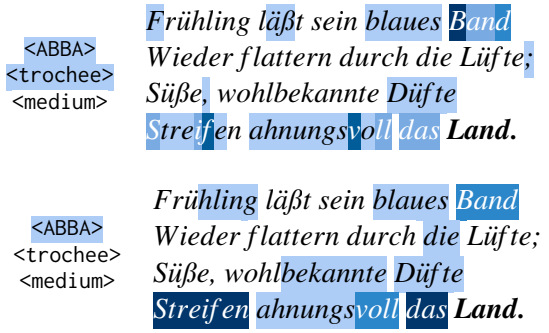

Figure 8: A well-known stanza by Mörike (1832) with visualized attention from ByGPT5 (top) and GPT-2 (bottom) when generating the last syllable.

German		English	
Ein Reiter steht am Hafen,	A	With languid smile, the stealing tear retires,	A
Der schaut die Flut nicht an,	B	And the slow fading light on trembling fires!	A
Er hört die Schiffer schlafen	A	Now she receives the golden circlet round,	B
Im stillen Ozean.	B	And fills the woven chambers with a sound;	B
Schweigend stehn die Burgen nieder,	A	The first who learned the lesson there	A
Und die Lüfte sind verhallt,	B	Had learned to scoff and scorn to sneer,	A
Und die Trommeln klingen wider,	A	And that the learned might have been	B
Und die Büchsen knallen halt.	B	A shameless woman and a queen.	B
Der Greis erbebt, die Hand erstarrt,	A	Then came the labour of the day within,	A
Die Kinder schauern vor dem Sterben;	B	The gray beginning of the week,	B
Die Stimme bricht, die Thräne fallt,	C	And down we went with hope and terror sin,	A
Sie sieht ihm nach mit nassem Beben.	B	And not a word to say and speak.	B
Die Strömung wiederum durch alle Glieder dringt,	A	For the stars are in the sky;	A
Und alles, was da lebt und wallt und leuchtet, singt.	A	And the stars have gone to die	A
Da steigt ein Palmenstrauch aus dem erhabnen Lichte,	B	With their songs of joy and fear,	B
Er schwimmt auf einer Fluth und singet in Gedichte.	B	With their music and their cheer.	B

Table 8: Additional example poems generated with ByGPT5 (base) in German and English.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
9

☒ A2. Did you discuss any potential risks of your work?
It is likely that the risks associated with using a poetry generation system trained on texts that are already hundreds of years old may be manageable and may not require explicit addressing.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

3,4

☑ B1. Did you cite the creators of artifacts you used?
3, Abstract A

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All artifacts are publicly available under permissive licenses without restrictions. Which is why we didn't explicitly discuss terms of use.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We do not explicitly discuss intended use of our artifacts, as the use case does not differ from the artifacts we derive them from.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We collect our data from existing datasets which already have quality control performed on them, to which we reference.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
3,4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

3,4,5,6

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We report number of parameters used in our models, but not the overall computational budget. Our models were trained as low priority slurm jobs which were often interrupted which would make such estimates inaccurate. We do however discuss computational performance in Section 9.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
3,5

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5,6

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
All these things are documented in our code artifact.

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

6

☒ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
We detail our annotation process in our own words, not by copying the text of instructions verbatim.

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Appendix C

☒ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
The data we collect from our annotators does not contain sensitive information. We told our annotators what we're aiming for, but didn't explicitly write about these instructions in our paper.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Appendix C