AraPoet: Controlled Arabic Poetry Generation with a Twist

Badr AlKhamissi and **Mona Diab**
Responsible AI, Meta
bkhmsi@fb.com, mdiab@fb.com

Abstract

Poetry is deeply embedded in Arabic culture. It has been the dominant form of art among Arabs for more than a millennia. In the era of pretrained large language models (LLMs), we ask ourselves whether finetuning such models on a publicly available Arabic poetry dataset can generate equally plausible poems that are indistinguishable from that of humans. Specifically, we finetune an Arabic monolingual (**AraT5**) and a multilingual (**mT5**) sequence-to-sequence pretrained models with sizes 220M and 1.2B parameters respectively, then combine them using the Twist Decoding method to leverage the best out of each model. We go a step further and attempt to control the style of the poem by conditioning the model on a set of attributes. Namely, the poet name, country, era, meter, theme, language type, and the number of verses of each poem for the model to generate.

1 Introduction

Poetry is an art form that is deeply human. It is used to convey what normal words fail to express. Their rhythmic nature are aesthetically pleasing and have been used to evoke meaning and emotions among people throughout history. The first attempts to develop systems that can automatically generate poetry dates back to the 1960s (Queneau, 1961). Early endeavors made use of simple combinatory processes, such as interchanging lines in a set of poems, to generate new ones (Gonçalo Oliveira, 2017). However, despite being a computational problem in nature, early approaches came mostly from different groups of poets and researchers from the humanities (Gonçalo Oliveira, 2017), such as the Portuguese Movement of Experimental Poetry (Barbosa et al., 2014) and the French Atelier of Literature Assisted by Maths and Computers (Oulipo, 1981). It was not until the works of (Gervas, 2000) and (Manurung, 2003) that the Artificial Intelligence (AI) community embraced "intelligent" poetry generation. Note that such computational systems require to deal with several layers of language such as phonetics, lexical choice, syntax and semantics in order to generate something of meaningful creative value. In other words, it requires a delicate use of language that explores the relationship between creativity and randomness.

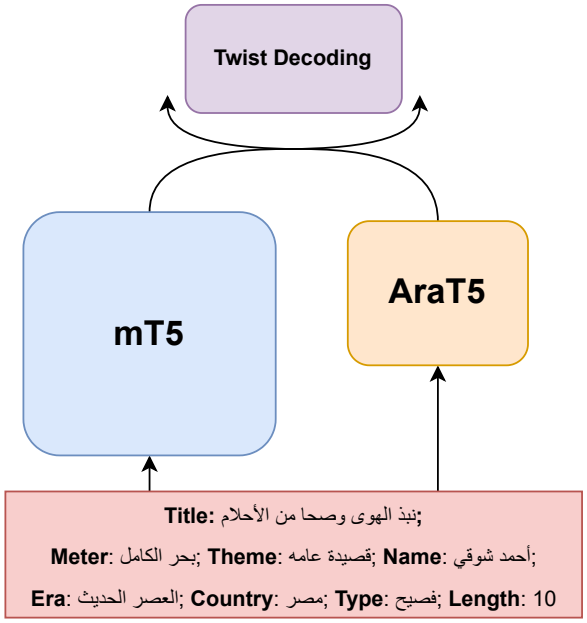

Figure 1: AraPoet Architecture. The same input is given to both models which are then combined during inference time using the Twist decoding method. The input shows the different conditions considered in this work to control the output of the generated poems.

In this work, we focus on controlled poetry generation in the Arabic language. Specifically, we finetune two pre-trained language models on the largest publicly available dataset for Arabic poetry. The first model, **AraT5** (Nagoudi et al., 2022), has been explicitly pretrained on Arabic text, while the other model, **mT5** (Xue et al., 2021), has been pretrained on multiple languages including Arabic. We then combine both models using the Twist decoding method to benefit from what each model has to offer simultaneously during inference time

Attribute	# Classes
Poet Name	7050
Meter	79
Country	22
Theme	18
Era	14
Language Type	4

Table 1: The number of classes of each attribute after preprocessing the ASHAAR dataset.

(Kasai et al., 2022).

To control the output of our model, we condition it on a set of attributes provided in the input to the encoder during finetuning, which a user can then use to prompt the model with a specific set of values to guide the generation to a specific style. Concretely, the attributes considered in this work are the (1) *title* of the poem which can be any free-form text, (2) the *meter* or the rhythmic structure of the poem (Al-shaibani et al., 2020), (3) the *theme* of the poem (e.g. romantic, political, etc.), (4) the *name* of the poet, (5) the *era* of the poem (e.g. Modern, Pre-Islamic, etc.), (6) the *country* of the poet, (7) the *language type* (e.g. Modern Standard Arabic, Dialectical, etc.) and (8) the *number of verses* to be generated. Table 1 shows the number of classes found in the ASHAAR dataset for each of the discrete attributes.

2 Dataset

Split	# of Poems	# of Tokens	# of Lines
Train	241,898	129,549,770	7,312,490
Test	12,732	7,199,613	402,369

Table 2: Dataset Statistics as measured by the **mT5** tokenizer.

In this work, we use ASHAAR[1], the largest publicly available dataset for Arabic poetry. The dataset contains $254,630$ poems, which we split by uniformly sampling 5% to be a held-out testing set that we use for evaluation, and the rest is used for training. Table 2 shows the number of samples in each of the training and testing sets. The dataset also comes with fine-grained attributes for each poem that we use to condition our models for controllable poetry generation. Table 1 shows each attribute along with it's corresponding number of classes. For example, the dataset has poems from $7,050$ poets. Note that we clean the values of each attribute by stripping the diacritics, and removing the prefix bHr[2] from the *meter* attribute. One limitation of this dataset is that it contains a lot of missing values that we put as "None".

3 Models

To train AraPoet, we finetune two T5-based models on the ASHAAR dataset. Specifically, we use the monolingual **AraT5** model that achieves state-of-the-art on many tasks in the Arabic Language Generation (ARGEN) benchmark, and the powerful multilingual $\textbf{mT5}_{\textbf{Large}}$ model due to its much larger capacity and support for the Arabic language. Further, we use **MARBERTv2** to classify the style of each generated poem to verify whether the output follows the provided prompt (see Section 6).

AraT5 Nagoudi et al. (2022) pre-train **AraT5** with a mix of Modern Standard Arabic (MSA) and Twitter data that make up 29B tokens combined. The Twitter data is approximately 28.39% dialectal and 71.61% MSA. The model follows the T5_{Base} architecture making up $\sim$ 220M parameters. We note that this model does not have diacritics in its pre-trained vocabulary. Therefore, we remove them from the training set when finetuning this model.

mT5 On the other hand, Xue et al. (2021) pre-train **mT5** with 57B Arabic tokens along with data from a 100 other languages. In this work, we use the large variant of this class of models which is made up of $\sim$ 1.2B parameters.

MARBERTv2 Abdul-Mageed et al. (2021) introduce **MARBERTv2**, which is an encoder-only model that was pre-trained on 1B Arabic tweets using the masked-language-modelling objective. Specifically, it follows the $\text{BERT}_{\text{BASE}}$ architecture (Devlin et al., 2019) with a total of $\sim$ 160M parameters.

4 Experimental Setup

Pre-processing For all models we only remove the elongations (i.e. tatweel), and for the **AraT5** model we further remove the diacritics.

Training Each model is finetuned for 10 epochs with a learning rate scheduler that first warms-up linearly for 1000 steps then decay linearly over the

[1] https://huggingface.co/datasets/arbml/ashaar

[2] This paper uses Buckwalter transliteration

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2
AraT5	64.28	59.97	64.28	13.59	4.25
mT5	57.20	51.98	57.20	9.31	2.17
Twist	58.13	53.07	58.13	9.42	1.95

Table 3: ROUGE and BLEU scores of each model variant.

course of training from $5e-5$ to 0. The input is padded or truncated to a maximum sequence length of 128, and the output generates a maximum of 256 tokens. The effective batch size used for training is 64. The weights of the pre-trained models are all obtained from the HuggingFace library (Wolf et al., 2019).

Generation To generate a high quality poem from the trained models one must tune the generation parameters appropriately, since greedy decoding fails to generate a plausible poem due to constantly repeating many words and sentences. Therefore, after evaluating our trained models qualitatively on various prompts, we found the following parameters to work well. First, we use beam search with a beam-width of 3, top-k sampling (Fan et al., 2018) with $k = 50$, nucleus sampling (Holtzman et al., 2020) with $p = 0.9$, penalized sampling (Keskar et al., 2019) to avoid generating duplicate substrings with a repetition penalty of 5, and finally a temperature of 0.7. Those parameters were kept constant for all generations including for the **`Twist`** decoding method.

Twist Decoding This method allows us to benefit from the complementary strengths of two models that do not share the same vocabulary or tokenization scheme (Kasai et al., 2022). Thus enabling us to use a high capacity but multilingual model with a smaller language-specific one.

5 Results

In this section, we report the ROUGE and BLEU scores of each variant to automatically evaluate the generated poems (see Table 3). It can be seen that **`AraT5`** performs the best in terms of the different metrics while **`mT5`** and **`Twist`** obtain similar results.

6 Analysis

Here, we ask whether our trained models generate poems that do indeed follow the style provided in the prompt in terms of the *meter*, *country*, *theme*, *era* and *language type* of the poem. To test this, we finetune **`MARBERTv2`** (Abdul-Mageed et al., 2021) for each attribute separately. Specifically, we only take the poems that do not have a missing value for the corresponding attribute and use it as the training set for finetuning. The same is done for the testing set. The number of classes used for each attribute is shown in Table 1. Further, Table 4 shows the testing accuracies of the generated poems for each model. Compare this to the ground truth poems of the testing set which can be considered as an upper bound. The sizes of the train and test sets used for the analysis is also shown in Table 4. We exclude the *poet name* from the analysis since the performance was below 1%.

7 Discussion

Despite performing better on the automatic evaluations, the **`AraT5`** model underperforms significantly when evaluating the generated poems qualitatively in terms of poeticness and relevance to the provided conditions. The analysis done in Section 6 confirms this quantitatively by measuring the degree to which the generated poems follow the style provided in the prompt. Both models appear to be generate grammatically sound outputs and coherent poems; that is they display a consistent theme, but only up to a specific number of lines. The combination of **`mT5`** and **`AraT5`** using the Twist decoding method perform better than either model alone when doing a qualitative evaluation. Further analysis is needed to confirm those in a quantitative manner.

8 Related Work

Constrained poetry generations using language models is not new. Previous work have used Markov models (Barbieri et al., 2012) and Recurrent Neural Networks (RNNs) (Potash et al., 2015) to predict the next token conditioned on a sequence of input tokens. Yan (2016) proposed a RNN with a polishing schema, by iteratively refining the poem composition to satisfy certain requirements. On the other hand, Ghazvininejad et al. (2016) uses a

Attribute	Twist	mT5	AraT5	Ground Truth	| Train |	| Test |
Meter	**44.19%**	43.01%	22.50%	87.78%	145,666	4564
Country	26.29%	**27.43%**	21.57%	50.64%	60,722	3306
Theme	**39.54%**	39.42%	39.30%	49.73%	64,174	3346
Era	34.45%	**35.23%**	32.78%	51.07%	139,998	7423
Language Type	83.01%	**83.57%**	76.19%	88.75%	174,156	9219
Average	45.50%	**45.73%**	38.47%	65.59%	-	-

Table 4: The testing accuracies of the generated output and the ground truth as described in Section 6. The numbers in **bold** are the highest accuracies for each attribute among the generated outputs.

Finite-State Acceptor to guide the RNN language model to control the rhyme and meter. Whereas Malmi et al. (2016) uses the RankSVM algorithm coupled with a deep neural network to predict the next full line from a pool of human-produced lyrics while taking into account the rhyme, structure and semantic similarity. In another line of work, Yan et al. (2013) approached poetry generation through a generative summarization framework that incorporate certain features as constraints to be optimized (see Gonçalo Oliveira (2017) for a survey).

In more recent work, Talafha and Rekabdar (2019b) use two Bidirectional Gated Recurrent Unit (Bi-GRU) based models, one for composing the first line of the poem and another encoder-decoder variant with hierarchical attention for producing other lines. Similar to this work, they use their models to generate Arabic poems. The same authors proposed another model that combines phonetic and semantic embeddings for each word as its representation (Talafha and Rekabdar, 2019a). Hakami et al. (2021) finetuned a GPT-based model, pre-trained on English corpora, on a dataset of Arabic verses. In the same spirit, Beheitt and Ben Hajhmida (2022) pre-trained then finetuned a GPT-2 model on an Arabic corpus and on a Arabic poetry dataset respectively to achieve better quality.

9 Conclusion & Future Work

In this work, we present AraPoet, a method for the automatic but controlled generation of Arabic poetry. To achieve that goal, we train the largest single model for Arabic poetry generation by finetuning **mT5$_{\text{LARGE}}$** ($\sim$ 1.2B parameters) (Xue et al., 2021) on the largest repository of Arabic poems, the ASHAAR dataset. We make use of the metadata that comes with each poem to condition our model with a set of values that controls the style of the generated poem. Further, we finetune as well **AraT5** (Nagoudi et al., 2022) since it was solely pretrained on an Arabic corpus. Then, we leverage the best of both models through the Twist decoding method (Kasai et al., 2022). We compare the three variants using automatic and human evaluation. Finally, we conduct an analysis to see whether the generated poems follow the style provided in the prompt. This is done by finetuning **MARBERTv2** (Abdul-Mageed et al., 2021) on each attribute (e.g. meter, theme, etc.) then observing the testing accuracy of the generated poems on each attribute.

In future work, we would like to explore training the data further than only 10 epochs and making use of the full dataset by not truncating the poems to a maximum length that has been done due to time constraints. This should lead to considerable improvements in the quality of the generations. To better follow the style provided in the prompt, we would like to explore adding adversarial loss to the objective function in which a discriminator (such as **ARBERT** or **MARBERT**) would learn to distinguish between the classes of each attribute and then used to refine the output of the generations. Finally, given the final trained models, we would like to explore having a final stage of finetuning using reinforcement learning from human feedback (Christiano et al., 2017) that has been shown to follow human preferences by using it as a reward signal. Such a method would be very suitable for controlled poem generation since we do not have yet suitable metrics that we could optimize that captures the underlying aesthetics of poems.

Limitations

One limitation of this submission is that it lacks human evaluations. However, we are currently working with experts on evaluating a sample of the generated poems which will be ready for the camera-ready version if accepted.

References

Muhammad Abdul-Mageed, AbdelRahim Elmadany, and El Moatez Billah Nagoudi. 2021. ARBERT & MARBERT: Deep bidirectional transformers for Arabic. 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 7088–7105, Online. Association for Computational Linguistics.

Maged S. Al-shaibani, Zaid Alyafeai, and Irfan Ahmad. 2020. Meter classification of arabic poems using deep bidirectional recurrent neural networks. *Pattern Recognition Letters*, 136:1–7.

Gabriele Barbieri, Francois Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. volume 242.

Pedro Barbosa, Ana, Hatherly, DE E.M., Melo e Castro, Rui Torres, and Sandy Baldwin. 2014. Essays from portugal on cyberliterature & intermedia by pedro barbosa, ana hatherly, and e.m. de melo e castro edited by rui torres and sandy baldwin.

Mohamed El Ghaly Beheitt and Moez Ben Hajhmida. 2022. Automatic arabic poem generation with gpt-2. pages 366–374.

Paul Francis Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *ArXiv*, abs/1706.03741.

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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Pablo Gervas. 2000. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. In *University of Birmingham*, pages 93–100.

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

Hugo Gonçalo Oliveira. 2017. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 11–20, Santiago de Compostela, Spain. Association for Computational Linguistics.

Asmaa Hakami, Raneem Alqarni, Mahila Almutairi, and Areej Alhothali. 2021. Arabic poems generation using lstm, markov-lstm and pre-trained gpt-2 models. pages 139–147.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Hao Peng, Ximing Lu, Dragomir Radev, Yejin Choi, and Noah A. Smith. 2022. Twist decoding: Diverse generators guide each other.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Eric Malmi, Pyry Takala, Hannu (TT) Toivonen, Tapani Raiko, and A. Gionis. 2016. Dopelearning: A computational approach to rap lyrics generation. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ruli Manurung. 2003. An evolutionary algorithm approach to poetry generation.

El Moatez Billah Nagoudi, AbdelRahim Elmadany, and Muhammad Abdul-Mageed. 2022. AraT5: Text-to-text transformers for Arabic language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 628–647, Dublin, Ireland. Association for Computational Linguistics.

Oulipo. 1981. *Atlas de litterature potentielle*, volume Number vol. 1 in Collection Idees. Gallimard.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. GhostWriter: Using an LSTM for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924, Lisbon, Portugal. Association for Computational Linguistics.

R. Queneau. 1961. 100.000.000.000.000 de poemes.

Sameerah Talafha and Banafsheh Rekabdar. 2019a. Arabic poem generation incorporating deep learning and phonetic cnnsubword embedding models. *International Journal of Robotic Computing*.

Sameerah Talafha and Banafsheh Rekabdar. 2019b. Arabic poem generation with hierarchical recurrent attentional network. pages 316–323.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. 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.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *IJCAI*.

Rui Yan, Han Jiang, Mirella Lapata, Shou de Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *IJCAI*.