Generating Classical Arabic Poetry using Pre-trained Models

Maryam ElOraby[*]
Mohammed Abdelgaber[*]
Nehal Elkaref
Mervat Abu-Elkheir
German University in Cairo, Egypt
{maryam.eloraby,mohammed.abdelgaber,nehal.elkaref}@student.guc.edu.eg
mervat.abuelkheir@guc.edu.eg

Abstract

Poetry generation tends to be a complicated task given meter and rhyme constraints. Previous work resorted to exhaustive methods inorder to employ poetic elements. In this paper we leave pre-trained models, GPT-J and BERTShared to recognize patterns of meters and rhyme to generate classical Arabic poetry and present our findings and results on how well both models could pick up on these classical Arabic poetic elements.

1 Introduction

Arabic poetry dates back to the sixth century, making it the earliest form of Arabic literature. It's often divided into two categories; classical and modern poetry. The classical Arabic poetry refers to the poetry written before the 20th century, more specifically poetry that adheres to the rules of classical prosody (العروض *al-'arūd*); following meters or patterns of syllabic pulses, and a rhyme (القافية *al-qafiya*). Modern poetry, on the other hand, has liberated itself gradually from these rules.

Classical poetry is also called vertical poetry in reference to the vertical parallel structure of its two parts known as hemistichs. A classic poem is versified where each verse consists of two halves, each is called شطر *shatr* 'hemistich'. Each verse in a poem follows a meter. Meters fall into fifteen different categories collected by the grammarian and prosodist *Al-Farahidi*. Later, one of his students, *Al-Akhfash*, discovered one more meter making them sixteen.

2 Related Work

Generating poetry is not a straightforward task as there are rules that need to be maintained to ensure the presence of poetic elements such as rhythm and rhyme, whereby these constraints tend to be added as a part of the architecture of the model.
To model constraints during training, Hopkins and Kiela (2017) converted their training corpus into its corresponding phonetic encoding and trained a Long-Short Term Memory (LSTM) trained on these encodings. They also introduced another approach that had a character-level LSTM model trained on a generic corpus of poetry an upon outputting a word, it gets approved or rejected by a Finite State Acceptor (FSA) classifier which ensures that only meter abiding words can be a part of the final poem.
Ghazvininejad et al. (2016) created *Hafez*, a program trained to generate topical poetry. Their system relied on Recurrent Neural Networks (RNNs) for coherence and finite state machinery to constraint rhyme and rhythm. From prosaic text, Van de Cruys (2020) generated English and French poetry by having gated recurrent units (GRUs; Cho et al. (2014)) in an encoder-decoder setting and an added layer of general attention (Luong et al., 2015). To ingrain their output with poetic elements, they applied a prior probability distribution to their network's probability output, where probabilities relating to words abiding by rhyme and topic constraints were boosted.
RNNs have also been used without constrain, for example, to build on encoder-decoder architecture Yan (2016) created a network that constructs a poem during each iteration, which gets fed to the network during the following iteration, hence, each poem takes part in constructing the next. On the other hand, Zhang and Lapata (2014) reserved one RNN for building hidden representations for a current line of poetry which was then fed to another RNN that sequentially predicted words of the next line of poetry. Pre-trained models have been put to use to the same task as well. For instance, Beheitt and Hmida (2022) trained GPT-2 (Radford et al., 2019) on Arabic news then fine-tuned the model on Arabic poetry. Hämäläinen et al. (2022) made

[*] Contributed to the work equally.

use of an encoder-decoder architecture to generate modern French poetry, where the encoder is initialized from a pre-trained RoBERTa (Liu et al., 2019) checkpoint while the decoder is based on a pre-trained GPT-2 checkpoint. They scraped a corpus of French poems, and used it to train their model for sequence-to-sequence generation, where it predicts a verse given a previous verse in a poem. They also conditioned beam search on rhymes during the generation phase.

In our work we dedicate another two pre-trained models of different architectures to explore how effective they are at recognising patterns of classical Arabic poetry through the poems they are trained to generate.

3 Data

Initially, we compiled datasets of *ashaar*[1] and the *Arabic Poetry Dataset*[2]. Combined, each data sample was comprised of a poem's *era/country of origin*, *verses*, *author*, *meter* and a poem's *topic(s)*. Additional scraping was done from *al-diwan*[3] to fill in missing values and to add a new column to the dataset that is rhyme.
Furthermore, the dataset was tweaked to only contain poems from eight eras, the *Abbasid*, *Ayubi*, *Ottoman*, *Umayyad*, *Andalusian*, *Mamluk* eras and the *Pre-Islamic Period*. We target these eras as they have more structure compared to the Free Verse poetry which has no musical pattern or rhyme. We also chose to centralise our poems around 15 out of the main 16 meters of Arabic poetry and thus we removed meters variants from our corpus.

3.1 Meters

We depict meter frequency in Figure 1 and we can observe that the meters الطويل *al-Ṭawīl*, الكامل *al-Kāmil*, البسيط *al-Basīṭ*, and الوافر *al-Wāfir* are the most dominant, while the least frequent meters are المضارع *al-Muḍāriʿ* and المقتضب *al-Muqtaḍab*. Although such imbalance could be problematic but it is also reflective of the nature of classical Arabic poetry, where the aforementioned four most occurring meters were predominately utilised for writing poetry compared to other meters (Golston and Riad, 1997). Furthermore, المضارع *al-Muḍāriʿ* and المقتضب *al-Muqtaḍab* were rejected by most theoreticians, beginning with *Al-Akhfash*, who regarded them as artificial, fictitious, and not used in real poetry (Frolov, 1996). Each of the mentioned meters has its own unique sequence of *taf'īlāt 'feet'* where a line of poetry follows this pattern of feet in each of its' hemistich.

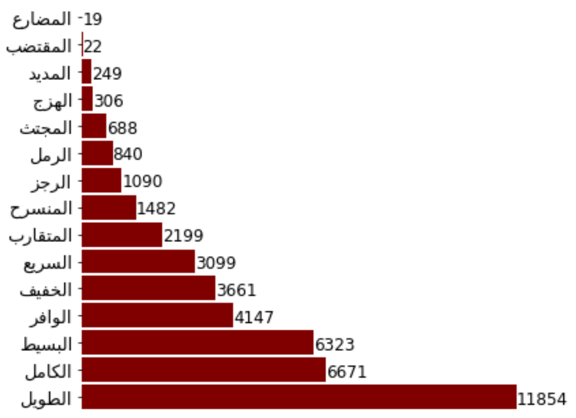

Figure 1: Meter Frequency

3.2 Topics

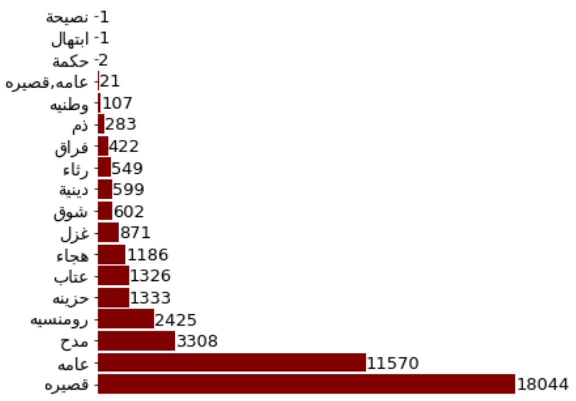

Figure 2: Topic Frequency

Poems in our targeted eras cover 17 different themes as shown in Figure 2. The most frequent topics are القصيرة *al-qṣyra* 'short' and العامة *al-'ama* 'generic'. As the the name of the former topic suggests, poems labelled as القصيرة *al-qṣyra* 'short' should have a small number of verses and accordingly we found that poems ranged from at least one verse to ten verses. However, we discovered that out of the entire 18K poems, 120 of them had more than 10 verses and four more had a substantially higher number of verses that reached over 50.

[1]https://huggingface.co/datasets/arbml/ashaar
[2]https://www.kaggle.com/datasets/ahmedabelal/arabic-poetry
[3]https://www.aldiwan.net

3.3 Rhymes

In vertical poems, a verse consists of two hemistichs. The second hemistichs of all verses within a poem are expected to end with the same letter. Figure 3 below contains present rhymes and their counts.

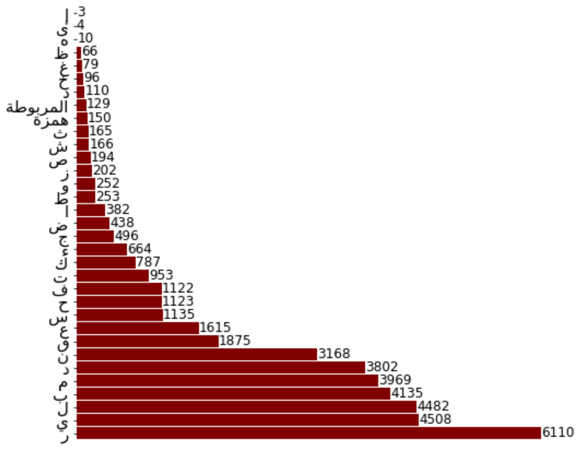

Figure 3: Rhyme Frequency

Rhymes could be consonants, short vowels or long vowels. Three of the Arabic short vowels written as diacritics have their long vowel version in letter form, and each pair is phonetically identical. In Table 1 we provide short vowels existing in our poems and their equivalent long vowel. In our dataset, a poem having a short vowel for a rhyme is labelled with its long vowel equivalent. Moreover, a short vowel and its corresponding long vowel could be used interchangeably within a poem. We can see an example of this in the following verses[4]:

وَلَمّا اِبتَلى بِالحُبِ رَقَ لِشكوَتي

وَما كانَ لَولا الحُبُ مِمَن يَرِقُ لي

أُحِبُ الَّذي هامَ الحَبيبُ بِحُبِه

أَلا فَاعجَبوا مِن ذا الغَرامِ المُسَلسَلِ

The second and fourth lines constitute the second hemistichs of a verse. And as can be seen, the second half of the first verse ends with (ي) while the second half of the last verse ends with a consonant however preceded by the diacritic (اِ).

[4]Poem by Bulbul Gram Ahajery from the Ayubi era https://www.aldiwan.net/poem14884.html

Long Vowel	Short Vowel	Pronunciation
ي	اِ	/i/
ا	اَ	/a/
و	اُ	/u/

Table 1: Short vowels and their equivalent long vowels

4 Models

We experiment with training two different transformer-based architectures: encoder-decoder model and a decoder-only model to generate poetry based once on a prompted meter, and again on a prompted topic.

4.1 BERTShared

Transformer-based encoder-decoder models have shown to significantly boost performance on a variety of Seq2Seq (sequence-to-sequence) tasks (Lewis et al., 2020; Raffel et al., 2020). However, the pre-training of encoder-decoder models is highly costly (Zhang et al., 2020). Rothe et al. (2020) proved the efficacy of warm-starting the encoder-decoder models with the checkpoints of publicly available pre-trained language models, such as BERT and GPT-2, for various Seq2Seq tasks.

Adopting this approach, we used CAMelBERT-CA (Inoue et al., 2021), a BERT checkpoint pre-trained on classical Arabic text, to warm-start both the encoder and decoder. This checkpoint was chosen since the subset of poems we chose to work with is known a priori to be written in classical Arabic. We specifically experimented with BERTShared architecture, in which the parameters of the encoder and decoder are shared, reducing the model's memory footprint by half (Rothe et al., 2020).

The input to the encoder is a vector sequence $\mathbf{X}_{1:n_x}$ of length n_x and at the decoder the model generates an output sequence $\mathbf{Y}_{1:n_y}$ of length n_y. The model defines a conditional distribution of target vectors $\mathbf{Y}_{1:n_y}$ given the input sequence $\mathbf{X}_{1:n_x}$:

$$p_{\theta_{enc},\theta_{dec}}(\mathbf{Y}_{1:n_y}|\mathbf{X}_{1:n_x}) \tag{1}$$

where the BERT-based encoder part encodes the input sequence $\mathbf{X}_{1:n_x}$ to a contextualized encoded sequence $\overline{\mathbf{X}}_{1:n_x}$:

$$f_{\theta_{enc}} : \mathbf{X}_{1:n_x} \rightarrow \overline{\mathbf{X}}_{1:n_x} \tag{2}$$

and the BERT-based decoder part models the conditional probability distribution of the target sequence $\mathbf{Y}_{1:n_y}$ given the sequence of encoded sequence $\overline{\mathbf{X}}_{1:n_x}$:

$$p_{\theta_{dec}}(\mathbf{Y}_{1:n_y}|\overline{\mathbf{X}}_{1:n_x}) \tag{3}$$

To generate poems using this architecture, we adopted the beam search multinomial sampling scheme, with a set maximum generation length of 130.

4.2 GPT-J

The performance of the transformer-based language models goes up according to Power-law with the number of model parameters, the size of the dataset, and the amount of compute (Radford et al., 2019). We use GPT-J, an open-source decoder-only transformer language model with 6B parameters (Wang and Komatsuzaki, 2021) which is four times the size of the largest GPT-2 model and two times the size of the largest GPT-Neo model (Black et al., 2021) parameters-wise.

In uni-directional models like GPT-J, when given an input sequence of tokens $\mathbf{w} = [w_1, w_2, ..., w_n]$ a probability $p(\mathbf{w})$ is assigned by the model to the sequence by factorizing it as the product of conditional probabilities:

$$p(\mathbf{w}) = \prod_t p(w_{\text{t}}|w_{\text{t-1}}, ..., w_1) \tag{4}$$

so the task becomes predicting the next token given the previously generated/input tokens.
Initial experiments done on the pre-trained GPT-J model showed the model is capable of generating coherent and grammatically correct Arabic sentences. This motivated us to use the model in Arabic poetry generation, adopting the Top-p method of sampling.

5 Experiment Setup

5.1 BERTShared Setup

In both experiments implemented using the BERTShared architecture, we tokenized our text using CAMeLBERT-CA's pre-trained WordPiece tokenizer. We also added two new tokens to the tokenizer to outline the structure of the vertical poems; a token to separate the two *shatrs* 'hemistichs' of a verse and another token to mark the start of a verse and separate the verses from each other.
In the first experiment we used meters as inputs and in the second we used topics. The poems are passed as the targeted outputs in both experiments, and 512 is used as a maximum output length since BERT trains positional embeddings for up to 512 positions. Furthermore, we split each poem in our dataset into chunks of 23 verses each.

Both of the BERTShared models were developed using the HuggingFace transformers library[5] and trained on a 16GB T4 NVIDIA Tesla GPU on a Google Colab notebook[6], using a batch size of 16. Both models were fine-tuned using Adam with the default learning rate of 5e-5, and a linear-rate warm-up of 3k.

5.1.1 Meters as Prompts

In the first experiment where we trained a BERTShared model with meters as inputs, we worked with a sample of 15,000 poems from our dataset due to memory limitations. The meter frequencies after sampling are shown in Figure 4.

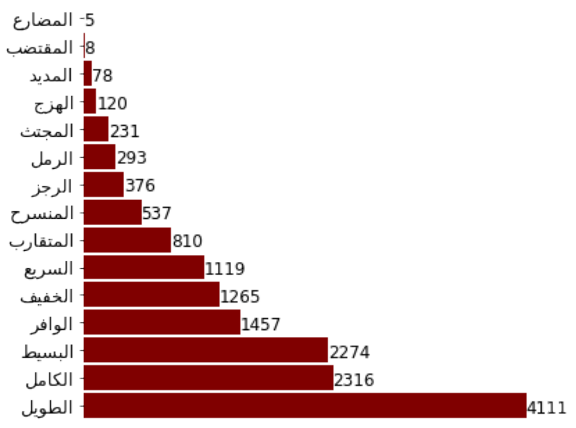

Figure 4: Meter frequency after sampling

Afterwards, we partitioned the dataset into 85% training and 15% validation using a seed of 42. Regarding the input length, we used 5 as a maximum length as we found that this length accounts for all meter labels.

5.1.2 Topics as Prompts

This experiment involved generating poems based on a prompted topic, thus topics were passed as inputs, with a maximum length of four. We excluded all samples tagged as العامة *al-'ama* 'generic' and القصيرة *al-qsyra* 'short' to focus on the specific topics rather than the general, miscellaneous ones. Second, the poems in our dataset were originally

[5]https://huggingface.co/docs/transformers/
[6]https://colab.research.google.com/

Grouped Topics Label	Topic Labels
(Sad) حزينه	(Sad) حزينه
	(Lament) رثاء
	(Separation) فراق
(Romantic) رومنسيه	(Romantic) رومنسيه
(Religious) دينية	(Religious) دينية
(Reproach) عتاب	(Reproach) عتاب
(Love) غزل	(Love) غزل
	(Longing) شوق
(Praise) مدح	(Praise) مدح
(Invective) هجاء	(Blame) ذم
	(Invective) هجاء
(Patriotic) وطنيه	(Patriotic) وطنيه

Table 2: Adopted grouping of poetry topics in BERTShared experiments.

labelled with 17 different topics, but some data samples were scarce. In attempt to balance out the number of samples per class, we ignored ابتهال *ibthāl* 'supplication', حكمة *ḥikma* 'wisdom', and نصيحة *naṣiḥa* 'advice' topics for being the rarest. Then we grouped some of topics together, in a manner slightly inspired by a grouping suggested by Alyafeai et al. (2022). The grouping for this experiment is shown in Table 2.

5.2 GPT-J Setup

Influenced by how character tokenizers perform better compared to the BPE morphological tokenizer in Arabic poem-meter classification task (Alyafeai et al., 2021), two models were developed using a character tokenizer, one of which uses meters while the other uses topics as prompts. Additional two models were implemented where the rhyme is passed once along with the meter and another with the topic to exert more control over the generation process.

Google's V3-8 TPU [7] was used to run the GPT-J models. Pre-training the model on Arabic text was not possible, as it requires at least a v3-256 TPU. Therefore, the GPT-J model pre-trained on the English-dominated *Pile dataset* (Gao et al., 2020) was fine-tuned on our dataset. The models with the highest validation score on the parti-

[7] https://cloud.google.com/tpu/docs/regions-zones

Grouped Topics Label	Topic Labels
(Sad) حزينه	(Sad) حزينه
	(Lament) رثاء
	(Separation) فراق
	(Reproach) عتاب
(Romantic) رومنسيه	(Romantic) رومنسيه
	(Love) غزل
	(Longing) شوق
(Praise) مدح	(Praise) مدح
(Blame) ذم	(Blame) ذم
	(Invective) هجاء

Table 3: Adopted grouping of poetry topics in GPT-J experiments.

tioned 90% training and 10% validation dataset were picked. Partitioning was done using a seed of 2022.

5.2.1 Data Preparation

Models fine-tuned on meters used 42,461 poems. However, models fine-tuned on topics used only 12,252 poems after going through a process of exclusion and grouping similar to what's done in BERTShared model.
All poems tagged as العامة *al-'ama* 'generic', القصيرة *al-qṣyra* 'short', ابتهال *ibthāl* 'supplication', حكمة *ḥikma* 'wisdom', and نصيحة *naṣiḥa* 'advice', were excluded. Then the rest of the poems were grouped as suggested by Alyafeai et al. (2022). Table 3 shows the final grouping used for GPT-J experiments.

The prompt format used to feed the poems to the model is:

```
[Tag]
Poem Text
<|endoftext|>
```

where *Tag* refers to the meter only, the topic only, the meter and rhyme or the topic and rhyme depending on which model is fine-tuned.
Each line of the poem *(bayt)* contains two verses separated by a forward slash (/) just like the following example:

وكاثرت متعة لذاته / فلم أر ذلك إلا متاعا

In the meter only and topic only models, rhyme was emphasized by inserting a hyphen (-) before

Hyperparameter	Value
lr	5e-5
end lr	1e-5
weight decay	0.1
batch size	16

Table 4: Hyperparameters set for all GPT-J models

Model Name	Total Steps	Warm up Steps
Meter only	1380	100
Topic only	300	30
Meter and Rhyme	1450	150
Topic and Rhyme	630	60

Table 5: Number of fine-tuning steps for each GPT-J model

the first letter of the rhyme.

5.2.2 Fine-tuning Hyperparameters

Table 4 shows the hyperparameters used for all the mentioned models. Higher and lower learning rates were used, but no sign of improvement was observed in the validation score. Table 5 shows the warm-up steps and the total steps for each model.

5.2.3 Inference Hyperparameters

To generate diverse poems, the inference hyperparameters used were:
Top-p = `0.9`, and Temperature = `0.9`

5.2.4 Omitted Models

Some initial model were implemented using AraGPT2 [8] BPE subword tokenizer. The poems showed a great tendency for repetition, as well as outputting invalid tokens and English letters.
An attempt was made to turn the AraGPT2 tokenizer into a character-level tokenizer by segmenting words into characters. This was done by inserting a hyphen (-) between every two letters. Another model was implemented using the new tokenizer and despite it achieving the best validation score of all models, the generated poems were incoherent and incomprehensible.

6 Evaluation

Some of the generated poems of our models are shown in Table 6. Because poems are essentially a form of art, no automated tool, or AI model could fully substitute the assessment of poetry by a human. Hence, we turned to four experts in classical Arabic poetry for an evaluation based on a number of dimensions as mentioned in 6.2. Additionally, we employed existing tools to test how much our model adheres to meters as will be explained in the following subsection.

6.1 Machine Evaluation

We first utilized the Arabic poetry classification model which Inoue et al. (2021) trained and made available on HuggingFace[9], to classify meters of the generated poems and assess the models' accuracy in capturing them.
We used each model to generate 10 poems per each of the 15 meters, consisting of a maximum of seven verses each. Then we passed the 300 poems to the poetry classification model, verse by verse. For each model and meter, we counted how many poems out of the 10 had all their verses adhering to their prompted meter. The results are presented in Figure 5. It shows that for الطويل *al-Ṭawīl* - the class with the most data samples - both models perform very well; the BERTShared model correctly captured the meter in the 10 poems it generated for this prompt, and the GPT-J model performed as equally for the 10 poems it generated for the same prompt. Both models could not capture the meters for any of المضارع *al-Muḍāriʿ* or المقتضب *al-Muqtaḍab*, the classes which had the least amount of samples in our dataset as shown in Figure 1. Furthermore, GPT-J model outputs display an overall linear correlation between the class size and the per-class accuracy. BERTShared, on the other hand, shows a good performance for some classes like الرمل *al-Ramal* that has 293 samples, but is under-performing, for instance, in السريع *al-Sarīʿ* meter of 1119 samples.

6.2 Human Evaluation

We sent out two surveys for our evaluators to assess the quality of poems with respect to meters and topics separately.

The survey analysing quality of topics of 16 poems, two poems from each topic group shown in Table 3 from each model. The evaluators were asked to answer the following questions from a scale of one to five, with one being the worst and five being the best:

[8] `https://huggingface.co/aubmindlab/aragpt2-base`

[9] `https://huggingface.co/CAMeL-Lab/bert-base-arabic-camelbert-ca-poetry`

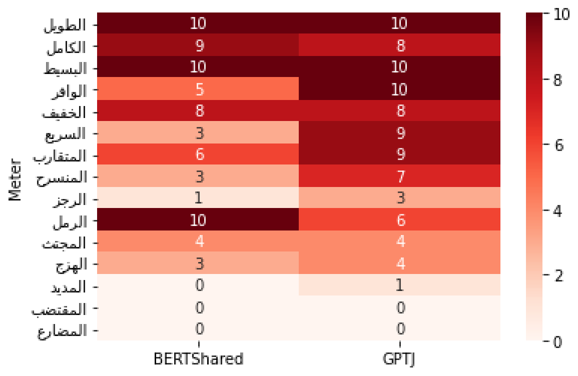

Figure 5: Per-meter accuracy of BERTShared and GPT-J. The y-axis is sorted by meter frequencies as in Figure 1.

1. How fluent is the generated poem?
2. How coherent is the poem with respect to as specified topic?
3. How consistent is the rhyme throughout the poem?
4. What meter does this poem follow?
5. How consistent is the meter throughout the poem?

Meanwhile, our second survey had a total of 18 poems, covering the following nine meters:

- الطويل *al-Ṭawīl*
- البسيط *al-Basīṭ*
- الوافر *al-Wāfir*
- الخفيف *al-Khafīf*
- المنسرح *al-Munsariḥ*
- الرمل *al-Ramal*,
- المتقارب *al-Mutaqārib*
- السريع *al-Sarīʿ*
- الكامل *al-Kāmil*

Our evaluators were required to answer the following questions and much like the first survey their answer should range from one to five:

1. How fluent is the generated poem?
2. How coherent is the poem with respect to as specified topic?
3. How consistent is the rhyme throughout the poem?
4. How much do verses follow the same rhythm?
5. How close are the verses to the specified meter?

Figure 6 shows each model's per-meter accuracy, how well the generated poems adhered to the prompted meter, as reported by the human evaluation. The results also vary between our models; GPT-J outperforms BERTShared in some meters but BERTShared does in some others. Overall, both models perform better the more data samples there are. Similarly, the per-topic accuracy for each model after averaging the evaluators' ratings is depicted in Figure 7.
Figure 8 reports the evaluators' ratings of the generated poems in terms of their fluency, thematic coherence, and rhyme consistency. On average, we see that BERTShared outperforms GPT-J in the fluency and grammatical correctness of the poems; and the coherency. However, GPT-J model shows a better ability of keeping the rhymes consistent throughout the verses of the poem.

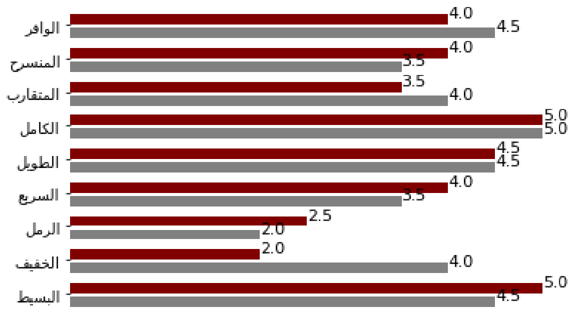

Figure 6: Per-meter accuracy of GPT-J (Red) and BERTShared (Grey) models as per human evaluation.

7 Conclusion & Future Work

We have presented two approaches to classical Arabic poetry generation. We have experimented with two transformer-based architectures: the encoder-decoder model initialized with BERT-based checkpoints, and the decoder-only architecture: GPT-J. We fine-tuned both models on a classical Arabic

Model	Prompt	Generated Poem
BERTShared	Meter: الوافر *al-Wāfir*	ألا من للموت أهون من حياة ومن لي أن أعيش بغير عيش عجبت لمن يموت وليس يدري وما في الموت موت بعد موت لعمرك ما الحياة وإن تناهت على ما قد مضى من الدنيا وعيش أرى الموت الشقاء لكل حي ولا خير الحياة لمن لحي
	Topic: (Romantic) رومنسيه	ألا أيها القلب الذي هو واجد على كل حال من سرورك موقد وإني لأرجو أن يدوم وصاله وما أنا إلا مغرم متزيد فيا ليت شعري ما الذي بي من الجوى إذا لم يكن لي في هواك مزيد أيا كبد حرى عليك تقطعت ويا لك من قلب عليك شديد فديتك لو كان الوصال بناظري إلى الله أشكو بعد ذاك الصدود ولو كنت أدري كيف أصغي لقولكم ولكن قلبي لا يزال يصيد
GPT-J	Meter: الطويل *al-Ṭawīl*	ومن جاءني بالخمس منه بحقه وكيف يحامي حاجب بالأجارب لقد حط في بعض اللقاء مكانه وما حط في بعض اللقاء مناسبي وقد جاءني بالخمس والجهل مذنب فما هو لي في بيت مجد وكاذب ولكن إذا ما أعطيت كف مجدها وليس لها غير الملاقاة ثاقب فلا برحت أيدي الليالي مخافة تقر لها عيني وتلوى الحواجب
	Topic: (Sad) حزينه	وعين على الأشجان بات ينيرها ومن دونه عود اللمى فيعيرها وحالف صبري في التجلد إنها إذا قام منها الوجد في الدهر حورها إذا نازحتها الدار بات يجودها بما جاد في الأيام منها أسيرها أتطلب دار لم ينل من بلادها ويعذب إلا في التراب بدورها إذا ما ارتدى بالماء لم يحو مقلة سوى ذلك الصافي ولا ذلك النورها

Table 6: Examples of generated poetry

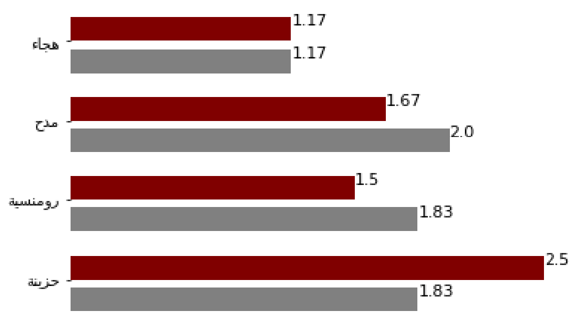

Figure 7: Per-topic accuracy of GPT-J (Red) and BERTShared (Grey) models as per human evaluation.

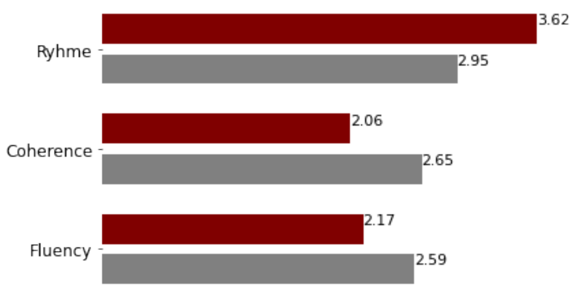

Figure 8: Rhyme consistency, fluency, and coherence ratings of the poems generated by GPT-J (Red) and BERTShared (Grey) models as per human evaluation.

poems corpus for two prompt-based generation tasks, and made use of two evaluation methods: one machine-based that focused on the models' ability to adhere to the prompted meters, and one human-based that focused on assessing the quality of the generated poems. The evaluators regarded the poems as interesting human evaluation revealed that BERTShared model performed slightly better in generating more fluent and coherent poems, but GPT-J model could capture the rhymes much better. In the future, we aim to incorporate human evaluation in the loop in a reinforcement learning environment, where the model should learn to generate the poetry based on corrected faulty poems.

Limitations

A limitation hindering both models are poems of topics labelled العامة *al-'ama* 'generic' and القصيرة *al-qṣyra* 'short' as they are the most occurring topics as show in Figure 2 yet they cover no distinct domain. Furthermore, we found no records online that could confirm that poets intended to write their poems following a certain theme, therefore we had to rely completely on *aldiwan*'s topic labelling not knowing what is based on or how accurate it is. Another is human evaluation, despite the presence of experts, there were too many poems to assess, and evaluators were not keen on the surveys especially meters evaluation as to them the number of meters to evaluate poems for is large.
In addition, GPT-J could not be pre-trained due to unavailability of the required hardware, so fine-tuning was used instead, which is suboptimal.

Acknowledgements

We gratefully acknowledge Google TRC program for providing the TPU machines that allowed us to train the GPT-J models. We are grateful, as well, to each of Abdel-Hamid Mohamed Taha, Mahmoud Salam Abo-Malek, Rayyan Mohamed and Mohamed Ashry for their contribution in evaluating this work.

References

Zaid Alyafeai, Maged Saeed AlShaibani, Mustafa Ghaleb, and Irfan Ahmad. 2021. Evaluating various tokenizers for arabic text classification. *CoRR*, abs/2106.07540.

Zaid Alyafeai, Maged Saeed AlShaibani, and Omar Hammad. 2022. Qawafi: Arabic poetry analysis using deep learning and knowledge based methods. `https://github.com/ARBML/qawafi`.

Mohamed El Ghaly Beheitt and Moez Ben Haj Hmida. 2022. Automatic arabic poem generation with gpt-2. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022)*, volume 2, pages 366–374.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Dmitry Frolov. 1996. The circles of al-Khalil and the structure of luzumiyyat of Abu'l-'Ala' al-Ma'arri. *Studies in Near Eastern Languages and Literatures. Memorial Volume of Karel Petraček, Praha*, pages 223–236.

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. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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.

Chris Golston and Tomas Riad. 1997. The phonology of classical arabic meter. *Linguistics: An Interdisciplinary Journal of the Language Sciences*, 35(1):111–132.

Mika Hämäläinen, Khalid Alnajjar, and Thierry Poibeau. 2022. Modern French Poetry Generation with RoBERTa and GPT-2. In *Proceedings of the International Conference on Computational Creativity, ICCC'22*.

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, pages 168–178.

Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

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

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

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.

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.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2238—2244.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 11328–11339.

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.