Prose2Poem: The Blessing of Transformers in Translating Prose to Persian Poetry

REZA KHANMOHAMMADI, University of Guilan, Iran
MITRA SADAT MIRSHAFIEE, Alzahra University, Iran
YAZDAN REZAEE JOURYABI, Shahid Beheshti University, Iran
SEYED ABOLGHASEM MIRROSHANDEL, University of Guilan, Iran

Persian poetry has consistently expressed its philosophy, wisdom, speech, and rationale based on its couplets, making it an enigmatic language on its own to both native and non-native speakers. Nevertheless, the noticeable gap between Persian prose and poems has left the two pieces of literature mediumless. Having curated a parallel corpus of prose and their equivalent poems, we introduce a novel Neural Machine Translation approach for translating prose to ancient Persian poetry using transformer-based language models in an exceptionally low-resource setting. Translating input prose into ancient Persian poetry presents two primary challenges: In addition to being reasonable in conveying the same context as the input prose, the translation must also satisfy poetic standards. Hence, we designed our method consisting of three stages. First, we trained a transformer model from scratch to obtain an initial translations of the input prose. Next, we designed a set of heuristics to leverage contextually rich initial translations and produced a poetic masked template. In the last stage, we pretrained different variations of BERT on a poetry corpus to use the masked language modelling technique to obtain final translations. During the evaluation process, we considered both automatic and human assessment. The final results demonstrate the eligibility and creativity of our novel heuristically aided approach among Literature professionals and non-professionals in generating novel Persian poems.

CCS Concepts: • Social and professional topics → Assistive technologies;
Additional Key Words and Phrases: Machine translation, transformers, Persian poetry, low-resource language

ACM Reference format:
Reza Khanmohammadi, Mitra Sadat Mirshafiee, Yazdan Rezaee Jouryabi, and Seyed Abolghasem Mirroshandel. 2023. Prose2Poem: The Blessing of Transformers in Translating Prose to Persian Poetry. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 22, 6, Article 170 (June 2023), 18 pages.
https://doi.org/10.1145/3592791

1 INTRODUCTION

The dawn of Persian poetry dates back to almost 12 centuries ago when Rudaki, the founder of classical Persian/Farsi language and poetry, inaugurated a new literary pathway. Persian poetry is
known for its use of figures of speech such as metaphor, simile, paradox, and hyperbole, which can make it difficult to comprehend [35, 45]. While Persian prose is straightforward, poetry employs irregular and anomalous speech to convey conceptual ideas, and breaking literary rules can make a poem more impressive. Generating Persian poetry is challenging, as ancient poems are scarce and more intricate than English poems [13]. Fewer studies have addressed the poetry generation task in Persian, whereas various Recurrent and Transformer-based approaches and baselines have been introduced in English [16, 28, 46]. Fortunately, pretrained Language Models have recently stepped forward in Persian Natural Language Processing [14] and have paved the way for advanced methodologies to address the generative aspect of Persian poetry [21].

In this work, we take the first step in translating prosaic Persian text to ancient Persian poetry by introducing a novel machine translation approach called Prose2Poem. Conventional deep neural machine translation techniques utilize a sequence-to-sequence (Seq2Seq) architecture, namely encoder-decoder networks, to encode input text as latent features from which the decoder can extract the output text. Such networks require a parallel corpus to learn from; however, no such parallel prose–poem dataset was available at the time of conducting this research. In our initial study, we collected 5,191 parallel prose–poem pairs, which were then used to develop encoder-decoder models (train set = 3,471, test set = 1,720). Nonetheless, the complex nature of Persian poetry makes it highly difficult for such networks to converge and generalize. Specifically, despite having access to a large quantity of data, these networks are unable to convert prose into a standard poetry that captures the same context as the input. However, despite our dataset containing only 5,191 parallel prose–poem samples, there exists a relatively broader span of raw ancient poems available (321,581 couplets). This large collection of couplets lacks parallel prose, leaving us with a small parallel corpus and a large collection of poems (details of our utilized datasets are included in Section 4). This study leverages the smaller parallel set to learn context translation using an attention-based seq2seq architecture and the bigger monolingual set for standard poem generation purposes using Masked Language Modelling (MLM). To integrate the two, we devised a set of heuristics to mimic the unique process of the generation of Persian poems. Furthermore, we designed our heuristics to be adaptable to generate translated poems in six different formats of ancient Persian poetry. The difference between these formats lies in the location of rhymes in the hemistichs of a couplet. Namely, these formats are Robaei, Ghazal, Ghasideh, Masnavi, Ghet’e, and Dobeiti.

Our contributions in this work are as follows:

1. We established a robust baseline for Persian prose to poem translation by introducing a novel low-resource Neural Machine Translation (NMT) method that is capable of effectively handling the unique characteristics of Persian poetry.
2. We introduced a parallel prose–poem corpus and a synonym–antonym dataset in Persian.
3. We made curated datasets and all implementation publicly available\(^1\) to aid further studies.

The rest of this article is organized as follows: First, in Section 2, we discuss the previous literature of Persian poetry and the bottlenecks of processing this natural language. Next, we break down our proposed NMT method in Section 3. In Section 4, we evaluate the results of our experimentation and provide several examples illustrating how our method performs on real-world circumstances. Finally, we conclude the article in Section 5.

## 2 PREVIOUS WORK

The applicability of Natural Language Processing (NLP) techniques has long been investigated in poetry. Still, poetry is a difficult phenomenon to computationally process because of its

\(^1\)https://github.com/mitramir55/Prose2Poem.
complexity. Poems are extremely rich in context [24] and very confusing for the machine to represent them [36]. Predictably, a vast majority of initial studies had to first analyze poetry’s core values such as Rhyme [22, 23], Meter [43], Word Similarities [49], and Stresses [1] to gradually push the field toward better Poetic Machine Intelligence.

Poetry’s generative aspect has also inspired a considerable amount of literature where various approaches have been explored over time [2, 3, 8, 16, 28, 46, 48], including statistical inference, Long Short-Term Memory (LSTM) networks and Variational Autoencoders. Additionally, as many other NLP research lines have witnessed, the revolution of Attention-based Language Models and Transformers has also paved the way for further advancements in this field. For instance, Santillan and Azcarraga [40] leveraged transformers to first generate a list of candidate poems. The list is filtered afterwards by calculating the cosine similarity between candidate poems and a Doc2Vec [29] model to choose the most cohesive candidate as output. Also, using transformers, de la Rosa et al. [10] demonstrated the advantages of cross-lingual transfer between Spanish, German, and English in predicting the metrical pattern of poems. Other studies have addressed poetry generation in other interesting settings. Liu et al. [31] and Xu et al. [47] introduced approaches to generating poetry based on visual clues in English and Chinese, whereas [33] developed the same concept while seeking to better satisfy rhythmical expectations.

Generally, translating prose to poem has attracted minor attention, and very few previous studies have addressed this problem. Ghazvininejad et al. [15] was among the first studies to focus on translating prose into poetry. They introduced an initial model using the encoder-decoder architecture and two biased decoding models where they encouraged the decoding of certain words. Similarly, Dunder et al. [12] also addressed poetry translation but among two different languages, Croatian and German. They examined and demonstrated the usability of statistical and neural machine translation techniques in a low-resource setting. As another study, Chakrabarty et al. [7] proposed parallel corpora for six language pairs, performed a wide range of empirical investigations such as multilingual finetuning, and compared their performance in poetry translation. Gokirmak [17] also recently studied the applicability of back-translation [41], transformer-based language models, and GPT-2 [38] in translating prose to poem. The study utilized a Czech, a Turkish, and an English corpus with around 183k, 670k, and 191k poems, respectively. These numbers are significantly larger than the single available dataset we used in our study, which sentences our work to be conducted under extremely low-resource setting. Fortunately, a plethora of prior studies have investigated means of integrating monolingual data into machine translation tasks. For instance, Cold Fusion [42], Shallow Fusion [18], and Deep Fusion [18] have engaged monolingual language models with NMT architectures to enhance translation quality. These articles inspired us to initiate a similar approach on Persian language.

3 PROPOSED METHOD

The performance of machine translation models is heavily reliant on parallel datasets, and with no such Persian datasets available, we found this an excellent opportunity to gather a parallel set of prose and poems to tackle our problem. Although valid poetry sources are exceptionally rare, the Didarejan\footnote{https://didarejan.com/} website has matched simple Persian prose to semantically similar Ghazal poems written by ancient Persian poets, such as Hafez, Saadi, and Molana (Rumi). Our first step toward developing a machine translation method involves collecting these parallel prose and poems for training encoder-decoder models.

We break down our translation task into three stages. First, in stage 1, we generate an Initial translation of the input prose using conventional NMT approaches. Due to data scarcity, outputs
at this stage contain poetic flaws and shortcomings. To complement our initial results, we process initially translated outputs for two additional stages. To accomplish this, we designed a heuristic in stage 2 to map initial translations to a set of masked sequences from which pretrained MLMs will then generate Final Translations in stage three (Figure 1). Since machine translation tasks deal with text generation, generative language models (e.g., GPT2) seem to be a perfect component to be incorporated, whereas this study outlines the applicability of MLMs (e.g., BERT [1]) in generating poetry. Combining this technique with a beam search algorithm lets us control a generative setting in which limitations and customizations can be applied to produce outputs that better suit poetic needs.

3.1 Stage 1: Initial Translation

We initially tackle the translation task by training the following three types of Seq2Seq models on our parallel corpus: (a) Vanilla Seq2seq, (b) Seq2seq + Attention, (c) Transformer. Given set $X_{1:a} = \{x_1, \ldots, x_a\}$ as an input sequence (prose) with $a$ length of words, we have defined stage 1 Seq2seq models as functions that translate $X_{1:a}$ input words to $\delta_{1:m}$ target words (Equation (1)) using the encoder-decoder architecture,

$$\text{Stage 1: } X_{1:a} \rightarrow \delta_{1:m}. \quad (1)$$

The most striking observation about these Seq2seq models is their effectiveness in expressing prose context as poetry. In other words, Seq2seq generated poems bear a close semantic resemblance to both the input prose and the ground-truth poem. In addition, ancient poems are generally context rich, and so the ability to include context within generated poems is extremely valuable. Additionally, these models learn to generate poems with fair lengths and acceptable poetic meters. The outputs, however, often do not adhere to particular literary rules and cannot be considered standard Persian poetry. Below is a list of their shortcomings:

- Very few generated samples in set $\delta_{1:m}$ seem to have formed rhythmic literature.
- Despite exhibiting an appropriate level of context transformation from $X_{1:a}$ to $\delta_{1:m}$, Seq2Seq models did not seem to narrate that information and arrive at a precise meaningful point in our experiments.
- Set $\delta_{1:m}$ forms irrational associations of words.

Consequently, Seq2seq architectures produce incoherent and unintelligible results with weak human evaluation scores on our dataset, making them an inefficient alternative. Training machine translation models with small corpora have the disadvantage that even robust architectures fail to generalize well and overfit the data. As our setting is extremely low resourced, the model cannot simultaneously learn from insufficient data to map input features to output features and also produce well-written ancient poems, which alone is an arduous task. Alongside our trained
NMT models, we also experimented with two available pretrained Persian GPT2 models from Bolbolzaban [5] and HooshvareTeam [21] that were already trained on the Ganjoor dataset. Still, these approaches either failed to converge or were stuck in local minima or produced low-quality poems. Having mentioned the Seq2seq models’ inefficiencies in poem generation, the scarcity of our parallel data, the uniqueness of Persian poetry’s nature, and the availability of a relatively larger number of monolingual data on the target side, we introduced the following two additional stages.

### 3.2 Stage 2: Keyword Partitioning Heuristic

As shown in Figure 1, the Keyword Partitioning Heuristic receives Initial Translation’s contextually rich outputs and generates masked sequences. In several ancient poems, poets have written more than one couplet to express the ideas they had in mind. Hence, we design the heuristic adaptable to generating multiple masked sequences. Given the initial translation of an input prose, we define heuristic $h$ (Equation (2)) as a function that generates $c$ masked sequences $M_{1:c}$,

$$h : \delta \rightarrow M_{1:c}.$$  \hspace{1cm} (2)

Generated masked sequences contain a set of tokens that are either keywords or masks. These keywords represent the context of initial translations that, as previously mentioned, are well captured by encoder-decoders in stage 1. Besides, masked tokens are placeholders that will be further predicted using MLMs in stage 3. This is while stage 2 is all about forming a poetic template that well captures initially translated poems’ context and provides a proper infrastructure for MLM to generate final poems. Evidently, the quality of our final outcome is heavily dependent on the quality of our keyword/mask partitioning. Below, we define a list of criteria we expect $h$ to consider:

- **Keywords Presence**: A fair number of keywords should appear in a masked sequence. This way, keywords better guide stage 3 MLM to generate a high-quality poem through mask prediction.
- **Keywords Adjacency**: To encourage the generative aspect of our work, keywords should be rationally separated, and enough mask spacing should be present among adjacent keywords.
- **Keywords Association**: Persian poetry’s literature is fulfilled with highly literary collocations. Hence, to understand and capture these collocations, $h$ should avoid partitions in which keywords are nonliterary associated.

Henceforth, we step through different components of $h$ to generate $c$ masked sequences while striving to meet the previously mentioned criteria. An illustration of the following components is provided in Figure 2.

#### 3.2.1 Keyword Extraction.

Primarily, we extract keywords from the contextually rich $\delta$ set. We used an unsupervised keyword extraction package called YAKE! [6] to automatically detect keywords in our text. This unbiased method is neither task nor language specific, making it suitable for our work. Stage 1 encoder-decoder models have successfully learned how to decode couplet-like outputs that is composed of two hemistichs. We feed the I set to YAKE! to capture a list of couplet-level keywords based on the big picture that the poem draws. Subsequently, capturing as much context as possible, we feed each hemistich individually to the model and extract hemistichs-level keywords. Finally, we combine the two lists and define the final set of keywords as $K_{1:n}$. Since some keywords can be extracted more than once among the two levels, we define set $F_{1:n}^k$ to keep track of each keyword’s frequency of appearance.

#### 3.2.2 Keyword Augmentation/Truncation.

To overcome the challenge of keywords presence, we augment the $K$ set to highlight context information. We gathered a synonyms–antonyms dataset.
Fig. 2. An overview of our final approach. Stage 1 generates an initial translation of the input prose. This initial translation performs well in translating the context, but it underperforms in generating a valid, standard poem. Next, the heuristic designed in this work (Stage 2) uses the initial translation to generate masked sequences. These masked sequences contain poetically associated words that convey the same message as the initial translation. Finally, Stage 3 uses these masked sequences, and a standard poem is generated as the final output through MLM and beam search decoding.

for all 4,756 words in the Ganjoor dataset. Given function $\text{Syn}(K_i, j)$, which returns the $j$th most frequent synonym of $K_i$, we define the synonyms set ($S$) and the Final Keywords set ($\Psi$) as follows:

$$S = \left\{ \text{Syn}(K_i, j) \mid 1 \leq i \leq n, 1 \leq j \leq F^k_i \right\},$$  \hspace{1cm} (3)

$$\Psi_{1:n'} = K \cup S.$$  \hspace{1cm} (4)
Having defined the $\Psi$ set, we now define $c$, which was previously used to declare the number of masked sequences we expect stage 2 to return ($M_{1:c}$). $c$ can also be viewed as the number of final couplets that we will be expecting stage 3 MLM to mask-predict and return as the final poem. We calculate $c$ by dividing the number of final keywords (i.e., $n'$) of the Ganjoor dataset by 10, since almost all hemistichs have a word count less than 10,

$$c = \left\lfloor \frac{n'}{10} \right\rfloor.$$

(5)

Furthermore, we make the heuristic adaptable to the number of keywords it lies between mask tokens to form a sequence. Since members of $M_{1:c}$ are in couplet level, we can also specify the keyword load in each of the right- and left-side hemistichs. Accordingly, we define $r$ and $l$ as the number of keywords to be indicated in each masked sequence’s right- and left-side hemistichs, respectively. Since each couplet contains two hemistichs only, using Equation (6), the heuristic calculates the overall number of keywords it needs to form $c \times 2$ masked hemistichs,

$$o = c \times (r + l).$$

(6)

3.2.3 Keyword Distribution. By now, we have collected the final set of keywords representing prose context. In this part, we will distribute them among mask tokens so that proper $M$ sequences are formed. The properness of these masked sequences heavily rely on how the predicted (keyword) and unpredicted (mask) tokens are arranged next to each other. As $M$ will be fed into a MLM to predict its masks, we design masked templates in favor of the MLM to produce better text as the final output. Based on our experiments, the adjacency of the keywords inside masked sequences has the highest impact on the quality of final outputs. Hence, we designed this subsection to overcome the keyword adjacency challenge.

Since more than 98% of all hemistichs available in the Ganjoor dataset have a word count of less than or equal to 10, we designate a maximum length of 10 tokens for each masked hemistich to tackle our problem. Next, we calculate index-based word frequency distributions $F_{1:10}^n$ using NLTK [4] to measure the presence of the Ganjoor dataset vocabulary in each of the 1 to 10 indices. To form logically ordered masked sequences, we intend to distribute keywords among those indices where each keyword (a) has previously been most appeared in and (b) is logically separated from the rest of the keywords. To do so, we iterate final keywords ($\Psi_{1:n'}$) and calculate a series of equations to obtain their final positional indices ($PI_{1:n'}$). This means that we will take a number of steps to figure out which of the indices between 1 to 10 is most suitable for each of the final keywords. These steps begin with calculating $PS_{1:10}$ (Equation (7)) per keyword (key) to calculate 10 positional scores. Finally, as mentioned in Equation (8), the hemistich positional index that the keyword will be finally placed in is equal to the argument (index) containing the maximum positional score,

$$PS_{1:10} = \{score_{id}^\text{idx}(key, i)|1 \leq i \leq 10\},$$

(7)

$$PI_{n'} = \text{argmax}(PS_{1:10}).$$

(8)

The most significant calculation here is measuring how likely it would be for the key to appear in a specific index ($i$). In Equation (8), we define the $score_{id}^{\text{idx}}$ scoring function to measure the key’s belonging to index $i$,

$$score_{id}^{\text{idx}}(key, i) = \frac{\text{score}^n\left(\text{n}^r_d(i), \text{n}^l_{agg}(i), 0\right) - \text{score}^n\left(\text{n}^l_d(i), \text{n}^r_{agg}(i), 1\right)}{id_{agg} + 1} \times \log_{10}\left(F_{i}^n\left(\text{Key}\right)\right).$$

(9)

Given $n^r(i)$ and $n^l(i)$ as the nearest right- and left-side neighbor indices of $i$, we define $n^r_d(i)$ and $n^l_d(i)$ as the distances between $n^r(i)$ to $i$ and $n^l(i)$ to $i$. Furthermore, $n^r_{agg}(i)$ and $n^l_{agg}(i)$ are defined
as the number of previously suggested keyword positions ($PI_{1:n'}$) for the $n'(i)$ and $n'(i)$ indices, respectively (Figure 3). Altogether, the $score^{idx}$ scoring function deteriorates by the number of previously suggested positions at index $i$ ($idx_{agg}$) and improves with two factors, where one is the frequency of $key$ in index $i$ (indicated as $F^n_i(key)$) and the other is the absolute difference between the right-side ($score^n(\ldots, \ldots, 0)$) and the left-side ($score^n(\ldots, \ldots, 1)$) neighbor scores. The intuition here is that the less aggregated a side is, the more appealing that index ($i$) is for $key$ to be placed in.

$$score^n(x, y, z) = \frac{x + 1}{(score_{agg}(i, z) + 1) \times (y + 1)}.$$  

To score a neighbor side, we define $score_{agg}$ (Equation (10)) where it improves the further away its nearest neighbor (i.e., $n'(i)$ or $n'(i)$) is and degrades by the more keyword aggregated it is. We use Equation (11) to define $score_{agg}$ as a measurement of how aggregated the right ($j = 0$) or left ($j = 1$) neighbor side of $i$ is.

$$score_{agg}(x, y) = \begin{cases} 1 & \{ j \mid 1 \leq j \leq n', PI_j < x \} & y = 0 \\ \{ j \mid 1 \leq j \leq n', PI_j > x \} & y = 1 \end{cases}.$$  

Having iterated all $\Psi_{1:n'}$ elements, we obtain the full $PI_{1:n'}$ set, which contains indices that each $\Psi$ set member would best fit in. Last, we create the $IK_{1:10}$ set where it has a list of all suggested keywords for each of the 10 indices. The more uniformly distributed the keywords are among the indices, the more likely it is to form logically ordered associated masked keyword sequences. However, comparing the size of the final keywords set $\Psi$ and the overall required number of keywords $o$, we choose to either re-augment or truncate the $\Psi$ set to make the two equal when they are in different lengths. Given $D = |o - n'|$, the former process (re-augmentation) refers to a process during which we augment the D least aggregated indices’ top frequent keywords using the $Syn(K_i, 1)$ function (expanding the $IK_{1:10}$ set). In contrast, the latter process (truncation) points to a process during which we remove the D most aggregated indices’ least frequent keywords (shrinking the $IK_{1:10}$ set). Having performed either of the two process, the $IK_{1:10}$ set is now expected to contain $o$ keywords distributed among indices. Hereafter, we will use the $IK_{1:10}$ set to extract the most contextually similar combination of keywords. However, to perform this selection, we need to introduce the Word Associations Graph, which will further be used for contextual relevance scoring.

### 3.2.4 Word Associations Graph

To make the heuristic knowledgeable of poetic word associations, we seek to capture a pattern among available ones. However, associated words in Persian
poetry cannot be easily extracted. Figures of speech have long played prominent roles in the significance of Persian poetry. Based on aesthetic values, they add non-literary adjustments, which may, in certain scenarios, make the poem undergo meaning divergence. Hence, defining a valid association as “a set of contextually similar words” is inadequate. Instead, we use the graph data structure to model associations indicated in our dataset’s poems to capture unbiased knowledge. This way, we can simultaneously investigate both the spatial attitudes and the context of agreement among associated words.

Using the NetworkX [19] python package, we construct a graph with $V$ vertices, each representing a word in the Ganjoor dataset’s vocabulary. Then we iterate couplets to extract unique word combinations of two and draw an edge between them at their earliest co-occurrence. Next, given $v_1$ and $v_2$ as two connected vertices and $v'_1$ and $v'_2$ as their indices (ranging from 1 to 20), we define edge attribute $E(v_1, v_2)$ using Equation (12), which scores the two vertices co-occurrence by their spatial distance,

$$E(v_1, v_2) = 1 - \frac{|idx_{v_1} - idx_{v_2}|}{20}.$$  

(12)

While some figures of speech, including paradox, make semantic similarity an invalid measure of how poetically two words are associated in Persian poetry, several other figures of speech, such as synonym, heavily rely on it. Consequently, it would be beneficial to encourage contextually similar word pairs and penalize contextually dissimilar ones. To do so, we train a Word2Vec model [34] on the Ganjoor dataset with a window size of 5 to map words onto 64-dimensional spaces.

Since the two $v_1$ and $v_2$ words can co-occur multiple times, we calculate $E(v_1, v_2)$ each time. Moreover, we define $\bar{E}(v_1, v_2)$ as the mean of all previously calculated edge attributes and $\text{sim}$ as $v_1$ and $v_2$ word vectors’ cosine similarity. Finally, we define their final association score as the arithmetic mean of $E(v_1, v_2)$ and $\text{sim}$. Final association scores will further be obtained from the graph to identify how much poetically associated word pairs are. The more similar and spatially close two words are, the more likely they are to be poetically associated.

3.2.5 Partitioning. Thus far, we have coped with Presence and Association-related challenges, which form components of a standard poem. However, the adjacency of keywords in masked sequences effectively contributes to the quality of final poems as well. This step is designed to extract a set of masked sequences with fairly distributed keywords in-between mask tokens such that are (a) adequate in number, (b) literary/contextually associated, and (c) logically separated among mask tokens. After adding a mask token to each $IK_{1:10}$ index-based keyword sets, we score possible occurrences using Equation (13) to satisfy these criteria,

$$\text{score}^p(d, g) = \text{sigmoid}\left( \log_2(d) \times g \right).$$  

(13)

To score each candidate combination, we step through every two adjacent keywords ($v_1$, and $v_2$) and accumulate their pairwise spatial differences, referred to as $d$. Using the Word Association Graph, we also define $g$, which is the mean of all pairwise association scores ($E(v_1, v_2)$). Given $d$ and $g$, $\text{score}^p(d, g)$ is calculated for every possible masked candidate. Finally, a total of $c \times 2$ top masked hemistichs are selected as final candidates and zipped indexwise to form couple-level masked sequences.

3.2.6 Poetry Format Adaptation. Persian poems can be categorized into different formats such as Ghazal, Masnavi, Dobeiti, or Ghet’e, where each has its unique way of locating rhymes inside couplets. As these locations are pre-determined in each format, the challenge becomes finding words that rhyme with each other so that we can place them in those specific locations. Thus, we gathered a list of words that rhyme together and used them to make poems adaptable to user’s format of
interest. More specifically, a total of 102 lists were gathered from different resources, each containing 2 to 146 words that rhyme together, with a total vocabulary size of 2,051 words. To select rhymes for masked sequences, we chose an initial rhyme and further explored other words that rhyme with it. It is noteworthy that this selection is highly based on how well associated a rhyme candidate is with the masked template. More specifically, we chose the candidate with the highest mean of pairwise association scores concerning previously selected rhymes and indicated keywords inside the masked sequence. This process enabled the heuristic to select contextually relevant rhymes per each masked sequence and take the final step toward better-formed $M_{1:c}$ set members.

### 3.3 Stage 3: Final Translation

Having generated a set of masked templates containing poetically arranged keywords, we define stage 3 as a function that maps masked sequences to final poems (Equation (14)),

\[
\text{Stage 3 : } M_{1:c} \rightarrow Y_{1:c}.
\]  

To do so, we use state-of-the-art MLMs to mask-predict the template and build a full couplet by filling in the blanks. As discussed previously, instead of using encoder-decoder models and letting them be fully in charge of translating prose to standard Persian poems, we asked it to solely generate a contextually rich initial translation of the input prose and let stages 2 and 3 be in charge of poetry generation. Besides, since we have a relatively larger amount of unlabeled monolingual poems, we pretrained different variations of BERT on the Ganjoor dataset to let it update its general knowledge of the Persian to learn latent ancient poetry features. In terms of masked language modelling, while available BERT models used a 15% probability of masking input tokens during their initial training phase, we set BERT models to randomly mask 40% during the pretraining phase to expect more of the model by giving it less and expose it to similar context to what it will see during the poetry generation stage. We pretrained available Persian MLMs such as BERT, RoBERTa, ALBERT, and DistilBERT on the Ganjoor dataset. In terms of length, we experimented with both hemistich-level and Couplet-level pretraining, among which the latter performed significantly better in predicting a contextually rich set of masked sequences.

Since MLMs predict a probability distribution over each word to predict a mask, we need a strategy to decode the final poem by selecting between a set of candidates per mask. We approach this using a beam search decoder with a customised heuristic inspired by Langkilde and Knight [27] and Khanmohammadi and Mirroshandel [25],

\[
\text{score}^b(g_4, g_3, g_2) = \frac{4 \times g_4 + 3 \times g_3 + 2 \times g_2}{10}.
\]

Given $g_4$, $g_3$, and $g_2$ as the 4-gram, trigram, and bigram of each mask-predicted candidate word and its previously indicated words, the beam search decoder constantly ranks the $k$ (Beam Depth) most probable sequences using Equation (15). Predicted sequences using the beam search algorithm are our approach’s final decoded poems ($Y_{1:c}$).

### 4 EXPERIMENTS AND RESULTS

Our curated parallel corpus contains 5,191 pairs of prose and poems. First, we left 1,720 pairs for evaluation and augmented the remaining (train set) up to 28,820 pairs using the Word Substitution technique to train Seq2seq (stage 1) models for 13 epochs with an embedding dimension of 256 and batch size of 64. Seq2seq models utilize Tensorflow GRU layers with 1,024 units that were initialized with Glorot uniform initializer (also known as Xavier uniform initializer). Adam and Sparse Categorical Crossentropy were also chosen as the optimizer and the loss function, respectively. The
Table 1. Comparison between Different Approaches

| Stage | Approach                  | SA     | PL     | B-1   | B-2   | B-3   | R-1   | R-2   | R-L   |
|-------|---------------------------|--------|--------|-------|-------|-------|-------|-------|-------|
| 1 IT  | Vanilla Seq2Seq          | 0.15   | 55.14  | 0.1721| 0.0576| 0.1930| 0.1010| 0.0001| 0.1326|
| 2 IT  | Seq2Seq + Attention      | 0.36   | 29.96  | 0.2784| 0.2011| 0.1508| 0.3961| 0.2508| 0.4398|
| 3 IT  | Transformer              | 0.57   | **12.18**| 0.2777| 0.2617| 0.1675| 0.6676| 0.4839| 0.6643|
| 4 FT  | h + BERT (h)             | 0.60   | 66.63  | 0.8435| 0.3323| 0.2092| 0.8776| 0.3725| 0.8540|
| 5 FT  | h + DistilBERT (h)       | 0.61   | 69.14  | 0.8507| 0.3463| 0.2009| 0.8792| **0.3882**| 0.8587|
| 6 FT  | h + ALBERT (h)           | 0.59   | 69.39  | 0.8326| 0.3004| **0.2244**| 0.8580| 0.3459| 0.8334|
| 7 FT  | h + RoBERTa (h)          | 0.65   | 61.86  | **0.8700**| **0.3512**| 0.2010| **0.8976**| 0.3857| **0.8769**|
| 8 FT  | h + BERT (c)             | **0.72**| 60.89  | 0.8386| 0.2991| 0.2032| 0.8738| 0.3376| 0.8504|
| 9 FT  | h + DistilBERT (c)       | 0.67   | 82.64  | 0.8444| 0.3040| 0.2102| 0.8854| 0.3374| 0.8637|
| 10 FT | h + ALBERT (c)           | 0.69   | 57.37  | 0.8385| 0.2585| 0.2224| 0.8721| 0.3002| 0.8473|
| 11 FT | h + RoBERTa (c)          | 0.67   | 68.98  | 0.8509| 0.3150| 0.2132| 0.8894| 0.3610| 0.8696|

(SA = Semantic Affinity, PL = Perplexity, B = BLEU, R = ROUGE).
The bold numbers show the best value in each column.

Monolingual pretraining phase of MLMs (stage 3) was carried out using 321,581 extracted couplets from the Ganjoor dataset\(^3\) (train = 289,422 test = 32,159). We pretrained BERT, DistilBERT [39], ALBERT [26], and RoBERTa [32] for five epochs using experimentally determined values of 0.40 and 0.15 masked token ratios in couplet-level (c) and hemistich-level (h) MLMs. To augment keywords in the heuristic (stage 2), we collected a dataset of synonyms and antonyms for each word in the vocabulary, totaling 4,756 words with an average synonym and antonym count of 7.12 and 2.34, respectively.

We used both automatic and human evaluation metrics to evaluate our experimental approaches. BLEU [37] and ROUGE [30] are among the most conventional metrics in evaluating text generation tasks. As part of human evaluation, two groups of 600 and 100 random poems were assessed by both poetry professionals and non-professionals, respectively. Moreover, we used Perplexity (PL) to measure the generalization performance of language models. We also used another metric to measure the semantic affinity between generated and ground-truth poems through thematically classifying them. To do so, we gathered a dataset of 1,319 couplets (augmented up to 2,567) and thematically labelled them as Divine (Elahi), Ethical (Akhlaghi), Amorous (Asheghaneh), or Philosophical (Erfani). Generally, this metric depicts what percentage of translated and ground-truth poems describe the same concept, enabling a high-level understanding of how semantically similar the two are.

As indicated in Table 1, we experimented with different Initial Translation (IT) and Final Translation (FT) approaches. According to the first three rows, seq2seq architectures suffer in low-resource settings where scarce data prevents them from generalizing, which leads to poor semantic affinity, BLEU, and ROUGE scores. Using the best-performing IT approach, namely the Transformer architecture, couplet- and hemistich-level models are developed. In contrast to IT approaches, heuristically aided FT models (rows 4 to 11) have overcome the data scarcity challenge by exploiting monolingual data and quality-control heuristics. A manifestation of this is witnessed when transitioning from IT to FT approaches, where scores have improved significantly. Generally, the combination of H and RoBERTa (rows 7 and 11) has yielded relatively higher scores among BLEU and ROUGE metrics in both couplet and hemistich levels. While couplet-level approaches such as h + BERT and h + ALBERT have achieved better results in terms of semantic affinity and perplexity. However, such unique tasks cannot be simply evaluated by automatic metrics, and more intelligence is needed for this purpose [20].

\(^3\)www.ganjoor.net.
Following the work of Van de Cruys [44], we asked a group of professionals (P) and non-professionals (NP) to annotate leading approaches’ generated poems by their Fluency (F), Coherence (C), Meaningfulness (M), Poeticness (PO), and Translation Quality (T) from 1 to 5. The professionals were further asked to rate the generated poems on a scale of 1 to 3 based on how much they replicated previous literature and how unoriginal they were. As shown in Table 2, leading MLMs, performed considerably better than transformer-based encoder-decoder models in most metrics.

A number of samples are shown in Table 3 that demonstrate different aspects of our work (more examples are provided in the Appendix). Since our poetry dataset includes ambiguous poetical and philosophical texts, they may convey different meanings at the same time. Hence, an exact translation is infeasible. Additionally, there are various figures of speech that further complicate matters. Thus, providing English equivalents of these samples was only possible. It is noteworthy that readers should not expect the input and output English equivalents to be precisely the same. There are many ways to illustrate prose’s message in poetry, and there is no rule of thumb that can be followed to do so. Even professionals cannot always translate prose directly into poetry. Some words may also seem irrelevant but are figuratively known as emblems that convey certain historical or cultural meanings. Hence, valid judgments of our method’s applicability cannot be made from English equivalents that are solely indicated to aid non-native readers to gain insight into Persian prose/poems. Hereafter, we will discuss certain aspects of these examples.

Interestingly, aesthetic values can almost be seen among every FT outcome. For instance, in sample 1, (Lawn) and (covered) are contextually related. Additionally, (spirit), (mentality), and (soul) are also close in meaning. The two words and can also be known as synonyms in Persian literature. Alliteration is another figure of speech that consists of words that are similarly written but may or may not differ in a phoneme. In sample 2, the FT consists of the word (wise) and (world) as two words that are similarly written but differ when are pronounced. The same can also be seen in sample 3 where ((has been repeated but pronounced similarly, whereas one means docile and the other means inhabitant. Furthermore, a paradox can be seen in the generated poem of sample 4 where (intellect) and (foolishness) are antonyms. Phonotactics are known as another figure where a certain character is repeated for several times. In this manner, in sample 5, (pronounced as /sh/) is repeated. These effects demonstrate how different aspects of this work have successfully contributed to poetical values. However, albeit minor, there are also evidences that depicts how some examples suffer from an increase of prepositions that results in the lack of coherency.

5 CONCLUSION
Persian poetry is well known for its mellifluous poetic style, captivating readers with its unique method of narrating personal, emotional, and historical topics. In computational terms, this mesmerizing phenomenon has solely been investigated as a text generation task, whereas, in this work, we took the initial steps of modeling Persian poetry as a translation task, where an input Persian prose can be translated to ancient poetry. Our approach involved gathering different datasets and
### Table 3. Random Prose and Their Respective Processed Text in Each Step

| Type | Text: |
|------|-------|
| **Sample 1** | |
| **Prose** | It is only by following the almighty that you can achieve your ultimate destiny. Because the heart is separate from the beauty. |
| **Poem (G)** | ناراذگاری گریزی هر چیزی از آن‌ها لازم نیست |
| **IT** | خود را می‌رسانند |
| **Heuristic** | چگونه از تغییراتGT‌های أساسی‌تری استفاده کنیم؟ — چگونه استفاده از فضاهای آزادی‌تری برای تغییرات GT؟ |
| **PT** | Your mark can be seen in creations that are occult to my sight. The divine will is my ultimate future, and I will sacrifice my life for it. |
| **Sample 2** | |
| **Prose** | We will become enlightened friends with the enemy if enmity breaks both nations’ hearts. |
| **Poem (G)** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **IT** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **Heuristic** | چگونه تغییرات GT‌های روزانه‌تری استفاده کنیم؟ — چگونه استفاده از فضاهای آزادی‌تری برای تغییرات GT؟ |
| **PT** | Since the wisest see in the universe creation and truth, the world of lovers is better. |
| **Sample 3** | |
| **Prose** | Even if sedition revolts the two worlds, we will be waiting for the one. |
| **Poem (G)** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **IT** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **Heuristic** | چگونه تغییرات GT‌های روزانه‌تری استفاده کنیم؟ — چگونه استفاده از فضاهای آزادی‌تری برای تغییرات GT؟ |
| **PT** | The universe belongs to all, which lets the creation make lovers decent. |
| **Sample 4** | |
| **Prose** | By pulling back the curtains, the words of each individual will be curative. |
| **Poem (G)** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **IT** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **Heuristic** | چگونه تغییرات GT‌های روزانه‌تری استفاده کنیم؟ — چگونه استفاده از فضاهای آزادی‌تری برای تغییرات GT؟ |
| **PT** | With regard to wisdom, you were not demure, and among all individuals, you are ignorant. |
| **Sample 5** | |
| **Prose** | The cupbearer took off the hair of the intoxicated. Now, he intoxicates the believers to delude them. |
| **Poem (G)** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **IT** | گردازش فلسفه‌های معاصر را به همراه می‌برد |
| **Heuristic** | چگونه تغییرات GT‌های روزانه‌تری استفاده کنیم؟ — چگونه استفاده از فضاهای آزادی‌تری برای تغییرات GT؟ |
| **PT** | The king elevates the sultan, which opens up wisdom to its sovereign. |

An English equivalent of the initial input and the final outcome is provided among all samples. (G = Ground truth).
experimenting with encoder-decoder models, MLM, and the beam search algorithm. The two primary challenges in trying to capture centuries of advancements in Persian poetry, are the insufficiency of available parallel data and the highly technical nature of Persian poems. Hence, we pretrained masked language models using relatively larger monolingual (poem-only) data and incorporated it with a set of heuristics to capture techniques used in generating Persian poetry. By assessing generated poems using official metrics and human assessors, we were able to establish a solid baseline for the Persian prose to poem translation task.

Future work is expected to integrate more robust methodologies with more technical literary knowledge for prose to poem translation while less supervision and engineering are applied. Most importantly, curating large parallel datasets would certainly enable computational methodologies to gain greater poetic insight, generalize better, and generate higher quality outputs.
## A Appendix

| Table 4. More Examples on Prose and Their Respective Processed Text in Each Step |
| --- |
| **Type** | **Text** |
| **Sample 1** |  |
| Prose | Since the angels saw you passing by, intoxicated, the tumult of resurrection rose.  |
| Poem (G) | مستنگانگیت و در خانواده نمای می‌ترسد به تمامی زندگانهای خودشان |
| IT | به اتفاق این تونسی قشنگتر بوده است که زندگی شن در محیط معمول  |
| Heuristic |  |
| PT | You passed by my life, intoxicated, and the world of angels is where I grieve.  |
| **Sample 2** |  |
| Prose | Although speaking is like pure water and lucrative, but less of it is better.  |
| Poem (G) | بایا بایها بایی بایی به اتفاق این تونسی قشنگتر بوده است که زندگی شن در محیط معمول  |
| IT | به اتفاق این تونسی قشنگتر بوده است که زندگی شن در محیط معمول  |
| Heuristic |  |
| PT | Your grace is fairly popularized and your words are true like water.  |
| **Sample 3** |  |
| Prose | The devil seeks to govern thousands of people, and anyone who is ruled by him also becomes the devil.  |
| Poem (G) | خالیت دوستی می‌در نهایت این مایلی زندگی معمول  |
| IT |  |
| Heuristic |  |
| PT | A look at creation reveals the almighty power and the horizon reveals my true world.  |
| **Sample 4** |  |
| Prose | The king told the servant to stop defining others and define himself.  |
| Poem (G) | کتکی از زمین برداشته از یک دست و دست می‌زند  |
| IT |  |
| Heuristic |  |
| PT | In this world, what matters the most is belief since the rest is of no importance. It states the right while your words narrate the others.  |
| **Sample 5** |  |
| Prose | Making mistakes as a believer in God is much better than the obedience of the rest, whose faith is ragged and of no value.  |
| Poem (G) |  |
| IT |  |
| Heuristic |  |
| PT | What I believe in most is the almighty, from whom, if I escape, I will suffer as much penalty as he will have mercy on me if I worship him.  |

An English equivalent of the initial input and the final outcome is provided among all samples. (G = Ground truth).
REFERENCES

[1] Manex Agirrezabal, Iñaki Alegria, and Mans Hulden. 2016. Machine learning for metrical analysis of English poetry. In Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING’16). The COLING 2016 Organizing Committee, Osaka, Japan, 772–781.

[2] Manex Agirrezabal, Bertol Arrieta, Aitzol Astigarraga, and Mans Hulden. 2013. POS-tag based poetry generation with WordNet. In Proceedings of the 14th European Workshop on Natural Language Generation. Association for Computational Linguistics, 162–166.

[3] Brendan Bena and Jugal Kalita. 2020. Introducing Aspects of Creativity in Automatic Poetry Generation. arXiv:2002.02511 [cs.CL]. Retrieved from https://arxiv.org/abs/2002.02511.

[4] Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O’Reilly Media, Inc.

[5] Bolbolzaban. 2020. GPT2-Persian. Retrieved from http://bolbolzaban.com/.

[6] Ricardo Campos, Vitor Mangeravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. Information Sciences 509 (2020), 257–289. https://doi.org/10.1016/j.ins.2019.09.013

[7] Tuhin Chakrabarty, A. Saakyan, and Smaranda Muresan. 2021. Don’t go far off: An empirical study on neural poetry translation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Online and Punta Cana, Dominican Republic. Association for Computational Linguistics, 7253–7265.

[8] Huimin Chen, Xiaoyuan Yi, M. Sun, Wenhao Li, Cheng Yang, and Z. Guo. 2019. Sentiment-controllable chinese poetry generation. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI’19).

[9] Sean M. Conrey. 2007. A canvas for words: Spatial and temporal attitudes in the writing of poems. New Writ. 4, 1 (2007), 79–90. https://doi.org/10.2167/new406.0

[10] Javier de la Rosa, Alvaro Pérez, Mirella De Sisto, Laura Hernández Lorenzo, Aitor Díaz, Salvador Ros, and Elena Gonzalez-Blanco Garcia. 2021. Transformers analyzing poetry: Multilingual metrical pattern prediction with transformer-based language models. Neural Comput. Appl. (11 2021), 1–6. https://doi.org/10.1007/s00521-021-06692-2

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805.

[12] Ivan Dunder, Sanja Seljan, and Marko Pavlovski. 2020. Automatic machine translation of poetry and a low-resource language pair. In Proceedings of the 43rd International Convention on Information, Communication and Electronic Technology (MIPRO 20). 1034–1039.

[13] Fadaee Elaheh. 2011. Poetry in translation: A comparative study of Silverstein’s monolingual and bilingual (English to Persian) poems. Int. J. Engl. Lit. 2, 3 (2011), 75–82.

[14] Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani, and Mohammad Manthouri. 2020. ParsBERT: Transformer-based model for Persian language understanding. arXiv:2005.12515 [cs.CL]. Retrieved from https://arxiv.org/abs/2005.12515.

[15] Marjan Ghazvininejad, Yejin Choi, and Kevin Knight. 2018. Neural poetry translation. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). Association for Computational Linguistics, 67–71. https://doi.org/10.18653/v1/N18-2011

[16] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: An interactive poetry generation system. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL’17). System Demonstrations. Association for Computational Linguistics, 43–48.

[17] Memduh Gokirmak. 2021. Converting prose into poetry using neural networks. Master thesis. Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, Prague.

[18] Caglar Gulecehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Hui-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. arXiv:1503.03535 [cs.CL]

[19] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using networks. In Proceedings of the 7th Python in Science Conference, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). 11–15.

[20] Lifeng Han. 2018. Machine translation evaluation resources and methods: A survey. arXiv:1605.04515 [cs.CL]. Retrieved from https://arxiv.org/abs/1605.04515.

[21] HooshvareTeam. 2021. ParsGPT2 the Persian Version of GPT2. Retrieved from https://github.com/hooshvare/parsgpt.

[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: Long Papers). Association for Computational Linguistics, Vancouver, Canada, 168–178. https://doi.org/10.18653/v1/P17-1016

[23] Harsh Jhamtani, Sanket Vaibhav Mehta, Jaime Carbonell, and Taylor Berg-Kirkpatrick. 2019. Learning rhyming constraints using structured adversaries. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 170:1–6. https://doi.org/10.1007/s00521-021-06692-2

ACM Trans. Asian Low-Resour. Lang. Inf. Process., Vol. 22, No. 6, Article 170. Publication date: June 2023.
Prose2Poem: The Blessing of Transformers in Translating Prose to Persian Poetry
[46] S. Xie. 2017. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation.

[47] Linli Xu, Liang Jiang, Chuan Qin, Zhe Wang, and Dongfang Du. 2018. How images inspire poems: Generating classical Chinese poetry from images with memory networks. arXiv:1803.02994 [cs.CL]. Retrieved from https://arxiv.org/abs/1803.02994.

[48] Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Wenhao Li. 2018. Automatic poetry generation with mutual reinforcement learning. In Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 3143–3153. https://doi.org/10.18653/v1/D18-1353

[49] Lingyi Zhang and Junhui Gao. 2017. A comparative study to understanding about poetics based on natural language processing. Open J. Mod. Ling. 07 (01 2017), 229–237. https://doi.org/10.4236/ojml.2017.75017

Received 24 November 2021; revised 25 February 2023; accepted 11 April 2023