International Journal of Transdisciplinary AI
Vol. 2, No. 1 (2020) 1-25

DOI: 10.35708/TAI1869-126247

# Generating Metrically Accurate Homeric Poetry with Recurrent Neural Networks

Annie K. Lamar[1] and America Chambers[2]

[1] Department of Classics, Stanford University, Stanford, CA, USA
kalamar@stanford.edu
[2] Department of Computer Science, University of Puget Sound, Tacoma, WA, USA
alchambers@pugetsound.edu

Received (11/15/2019)
Revised (05/09/2020)
Accepted (06/20/2020)

**Abstract** We investigate the generation of metrically accurate Homeric poetry using recurrent neural networks (RNN). We assess two models: a basic encoder-decoder RNN and the hierarchical recurrent encoder-decoder model (HRED). We assess the quality of the generated lines of poetry using quantitative metrical analysis and expert evaluation. This evaluation reveals that while the basic encoder-decoder is able to capture complex poetic meter, it under performs in terms of semantic coherence. The HRED model, however, produces more semantically coherent lines of poetry but is unable to capture the meter. Our research highlights the importance of expert evaluation and suggests that future research should focus on encoder-decoder models that balance various types of input – both immediate and long-range.

**Keywords:** poetry generation; neural network; recurrent neural networks; poetic meter; ancient languages

## 1 Introduction

Homeric poetry is poetry traditionally ascribed to the bard Homer and includes the *Iliad* and the *Odyssey*. Homeric poetry plays a central role in the field of Classics and strongly influences the study and understanding of Classical literature, mythology, and ancient culture, values, and militarism. In this paper, we investigate the use of recurrent neural networks for generating metrically accurate Homeric poetry – providing a new pathway for Classicists to analyze the metrical and semantic considerations made in the production of such epic oral poetry.

Generating metrically accurate Homeric poetry is a creative and challenging task. Homeric poetry is written in a strict and complex meter known as *dactylic hexameter*. Most poetry generation models focus either on syllabic-based meters or simple iambic meters. In contrast to such poetic meters, lines of dactylic hexameter contain varying numbers of syllables and words. An additional challenge is the relatively small amount of Homeric poetry available. The entirety of the Homeric canon is about 27,000 lines.

In the next section, we provide a more detailed description of dactylic hexameter. We also discuss the long history of analysis that Homeric poetry has enjoyed. We then provide a description of our data sets and the recurrent neural network architectures that we use. Finally, we present the results of having expert Classicists evaluate the poetry generated by our system.

## 2 Homeric Poetry

### 2.1 Homer and Oral Poetry

The *Iliad* and the *Odyssey* are two epic poems traditionally ascribed to the bard Homer and dated to the eighth century BCE. The *Iliad* tells the story of the Trojan War and the *Odyssey* recounts the hero Odysseus' long homecoming following the war. Homeric poetry is written in a dialect of ancient Greek called Homeric Greek.

Although we refer to Homer as the author of these texts for convenience, the two works are most likely the product of an extended oral tradition rather than a single poet named Homer. That is, the two works are likely the result of many (unnamed) poets who memorized, performed, and innovated on a set of standard poems over a long period of time.

### 2.2 Homeric Meter

Homeric poetry is written in dactylic hexameter, a metrical system in which each line consists of six metrical feet. A single foot can consist of one of three options: a dactyl, a spondee, or an anceps. A dactyl is a foot composed of one long and two short syllables. A spondee is composed of two long syllables. An anceps is composed of one long syllable and one other syllable of either type (long or short). Each line of dactylic hexameter must end with an anceps.

Figure 1 shows the first line of Homer's *Iliad* with the dactylic hexameter marked above. The feet are separated by a vertical bar. A long dash indicates a long syllable, while 'u' indicates a short syllable. In addition, the long syllables in the actual text are bolded. The pronunciation and translation are also shown in English underneath.

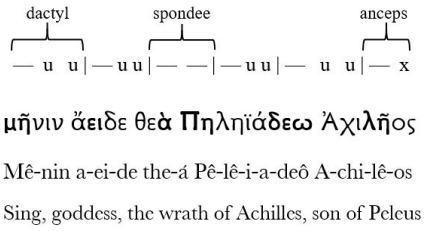


Figure 1: First line of the *Iliad* with metrical markings.

## 3 Related Work

### 3.1 Related Work in Antiquity

Poets have been attempting to imitate Homeric poetry in its style, content, and meter since ancient times. Plato writes in the *Republic* that "praisers of Homer... say that this poet educated Greece, and that in the management and education of human affairs it is worthwhile to take him up for study and for living, by arranging one's whole life according to this poet" [6,27]. Tragic poets contemporary to Plato were indeed considerably influenced by Homeric epic. Homeric motifs and style appear in the tragedies and satyric dramas of Aeschylus, Euripides and Sophocles [28].

Roman poets, including Livius Andronicus, Naevius, Ennius, and Virgil, likewise imitated Homeric epic. The structure of Virgil's *Aeneid* demonstrates this point: the first half of the Virgilian epic reflects the structure of Homer's *Odyssey* while the second half reflects that of the *Iliad* [8]. Homeric poetry influences even (relatively) modern writers; although providing an exhaustive list is beyond the scope of this paper, a few representative examples are James Joyce's *Ulysses*, Margaret Atwood's *The Penelopiad*, William Faulkner's *As I Lay Dying*, and Audrey Niffenegger's *The Time Traveler's Wife*.

In addition, scholars have been attempting to understand and quantify Homeric poetry for millenia. Greek grammatical scholarship from as early as the sixth century BCE has been focused on the reproduction of grammar and syntax of Homeric texts; The sixth century BCE scholar Theagenes of Rhegium, heralded as the first grammarian, is noted for his defense of Homeric grammar and mythology [23]. A couple centuries later, Plato devotes much of his *Republic* to understanding the nature of poetry and uses Homer as a model. In the modern era, we can look to Nietzsche and Freud for philosophical applications of Homer, and Butler and Coleridge for humanistic interpretations of the texts. Homeric texts were read and preserved by a wide range of Mediterranean peoples, from the coast of modern-day Spain, to northern Africa, and into the Near East. Because the transmission of Homeric texts has been considered critical by such a range of cultures and eras, clearly the text communicates universal ideas about the nature of humanity. For this reason, it is worthwhile to apply the tools of modernity to continue attempts to understand and reproduce Homeric poetry.

### 3.2 Poetry Generation in Modern Times

In the early twenty-first century most poetry generation was rule-based or statistical in nature.

Oliveira et al. created PoeTryMe, a platform for automatic poetry generation that used grammar rules and templates [22]. Tosa et al. created a rule-based haiku generation system that used arbitrary phrases chosen by a user to produce a haiku with the correct number of syllables on each line [20]. Tosa et al. improved on this rule-based system by incorporating more Japanese cultural characteristics into the generated haikus [21].

Jiang et al. used statistical methods and the principles of machine translation to generate Chinese couplets [13]. He et al. also generated Chinese Classical poems using statistical machine translation models where to generate each output sentence, a model specifically trained for an input sentence is used for generation [10].

Other approaches to poetry generation include genetic algorithms [18], [3], [25] and the application of text summarization techniques [26].

Since 2014, research on automatic poetry generation has been dominated by neural methods. Zhang et al. produced joint character-level recurrent neural networks to better capture poetic content and form [31]. Wang et al. used a bidirectional long short-term memory (LSTM) model to generate Chinese poetry [24]. Ghazvininejad et al. used a dictionary to impose rhyme constraints on poetry generated with a recurrent neural network [16]. In contrast, Lau et al. learned rhyme patterns automatically and proposed a pipeline of language, rhyme, and meter models to generate Shakespearean poetry [12]. Of particular relevance to this paper is Lau et al.'s findings that a vanilla LSTM is able to capture English iambic meter implicitly. In contrast to [12], we attempt to generate dactylic hexameter, a poetic meter more complex than iambic meters.

Our work in this paper expands upon our previous investigation of the use of encoder-decoder RNNs for generating metrically accurate Homeric poetry [15]. In addition to expanding on the experimental details and results, we also experiment with a new model (hierarchical recurrent encoder decoder) that is specifically designed for generating text that is semantically coherent.

## 4 Datasets

The entirety of the available Homeric canon contains 27,342 lines from Homer's *Iliad* and Homer's *Odyssey*. The text from both works is taken from Oxford's 1920 edition of the *Homeri Operi* [2].

Our training set consists of input output pairs where the input is one line of the poem and the output is the next line of the poem (see Figure 2). For example, given the first line of the *Iliad* as input, the model should generate the second line of the *Iliad* as output, so on and so forth.

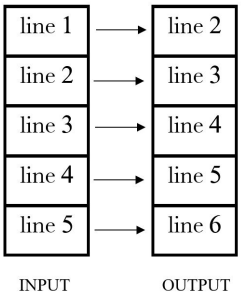


Figure 2: The training set

We generate 30 test sets by following the procedure outlined below 30 times:

- Randomly select a passage of 5 lines from the training set.
- For each passage:
  - Randomly select one line from the passage.
  - Remove the selected line (and its corresponding output) from the training set and place it in the test set.
  - Remove the preceding line (and its corresponding output) from the training set.
  - All remaining lines remain in the training set.

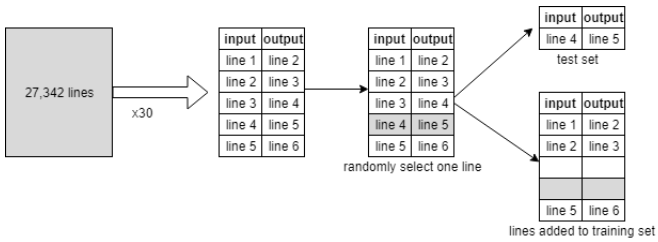


Figure 3: Procedure to generate test sets

This procedure is shown in Figure 3. Note that while the selected line (the shaded row in Figure 3) is added to the test set, the preceding line is not included in either the test set or the training set. The preceding line is removed from the training set so that the selected line (i.e. line 4) does not appear in the training set in any form: neither as an input nor as an output.

Once we have constructed our training and test sets, we further sub-divide the training set to create a validation set. The validation set was constructed by taking the last 10 lines from every 100 lines, resulting in a validation set that was approximately 10% the size of the training set and the reflected the diversity found in the training set.

## 5 Model Architecture

In this section, we provide a brief overview of the architecture of the recurrent neural network (RNN) models used in this paper.

### 5.1 Encoder-Decoder RNNs

An encoder-decoder RNN [14] is a neural network architecture that maps sequences to sequences. In this case, given a sequence of words (i.e. one line in a poem) the model generates an output sequence of words (i.e. the next line in the poem).

An encoder-decoder RNN works in two stages. The first stage involves mapping the input sequence to a high-dimensional space. This high-dimensional space represents a mathematical encoding of the *meaning* of the sentence that is language independent. The second stage then maps from points in this high-dimensional space back to the space of sequences.

Let $x = [x_1\ x_2 ... x_n]$ be the words in a given line of poetry. The encoder RNN incrementally builds an encoding $\mathbf{h}$ of the input sentence word-by-word using the following algorithm:

// Build the partial encodings
**for** $t = 1$ to $n$ **do**
  $\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$
**end for**

//Build the overall encoding
$\mathbf{h} = g(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$

where $\mathbf{h}_0$ is typically a vector of all zeros. The partial encodings $\mathbf{h}_t$ capture the meaning of the sentence up to the given word. The final encoding $\mathbf{h}$ is then a function of all of the partial encodings with $g(\cdot)$ commonly chosen to be $g(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n) = \mathbf{h}_n$. That is, $\mathbf{h}_n$ is assumed to be a sufficient encoding of the entire sentence.

Two common choices for the function $f(\cdot)$ are the long short-term memory (LSTM) model [7] and the gated recurrent unit (GRU) [14]. Both functions keep track of an internal state as a form of short-term memory. At each step, the new state is derived from the previous one in a process of forgetting and then updating in response to the new word $\mathbf{x}_t$.

Given the final encoding of the input sentence (which we will now denote as $\mathbf{h}_x$), the next line of the poem $\mathbf{y} = [y_1\ y_2\ ...]$ is generated using another RNN (the decoder). At each step, a word $y_t$ is sampled conditioned on the previously generated word $y_{t-1}$, the partial encoding $\mathbf{h}_{t-1}$ of the *decoder*, and the encoding of the input sentence $\mathbf{h}_x$. Note that in this context, the partial encoding $\mathbf{h}_{t-1}$ serves as a summary of the meaning of the previously generated words. Algorithmically, we have

**while** $y_{t-1}$ does not equal `EOS` **do**

Compute $p(y_t|y_{t-1}, \mathbf{h}_{t-1}, \mathbf{h}_x)$
Sample $y_t \sim p(y_t|y_{t-1}, \mathbf{h}_{t-1}, \mathbf{h}_x)$
**end while**

where `EOS` is a specially designated end-of-sentence symbol. The distribution $p(y_t|y_{t-1}, \mathbf{h}_{t-1}, \mathbf{h}_x)$ is, again, commonly chosen to be an LSTM or GRU function whose output is passed through the softmax function to produce a valid probability distribution over the words in the vocabulary.

The first model we use in this paper is an encoder-decoder RNN with two additional augmentations. First, we use a bidirectional encoding – i.e., partial encodings are computed from the beginning of the sentence forward and from the end of the sentence backwards. These two encodings are then concatenated together to produce the final encoding $\mathbf{h}_x$ for the input sentence. We also use an attention mechanism where the generation of each word $y_t$ is now conditioned on a weighted sum of *all* of the partial encodings of x rather than conditioning on $\mathbf{h}_n$ alone. A bidirectional encoding and attention mechanism have both been shown to significantly improve performance regardless of the natural-language application [4, 17, 29].

### 5.2 Hierarchical Recurrent Encoder-Decoder

The model described above has no explicit mechanism for generating semantically cohesive lines of text which is important when generating poetry. For this reason, we also experiment with a hierarchical recurrent encoder-decoder (HRED) model [9] which has an extra encoding step designed to keep track of the current semantic state of a dialogue (or poem).

The encoder RNN operates as normal – given an input $x$, it produces an encoding $\mathbf{h}_x$ of the input sentence. HRED also employs a bidirectional encoding but uses the GRU function instead of LSTM.

This encoding $\mathbf{h}_x$ is then passed as input to a *second* encoder RNN called the context RNN. The context RNN aggregates the encodings $\mathbf{h}_x$ of each successive input sentence. The context RNN acts as a form of global memory, summarizing the semantic content of the dialogue (or poem) up to the current point in time.

The decoder RNN then generates each word $y_t$ conditioned on the previously generated word $y_{t-1}$, the partial encoding $\mathbf{h}_{t-1}$ of the decoder, and the encoding of the context RNN. That is,

$$y_t \sim p(y_t|y_{t-1}, \mathbf{h}_{t-1}, \mathbf{h}_c)$$

where $\mathbf{h}_c$ is the encoding produced by the context RNN. In this way, each generated word is conditioned on all of the previous input sentences up to that point in time which provides a form of semantic coherence.

### 5.3 Word Embeddings

Finally, instead of representing each word in the vocabulary as a binary one-hot vector (i.e., a vector of all zeros whose length is the size of the vocabulary

with a single non-zero entry indicating the given word), we learn GloVe word embeddings [11].

A GloVe word embedding is a mapping from words in the vocabulary to a high-dimensional vector space. The mapping is learned from the co-occurrence of words in the poem. The learned word vectors provide a much richer representation than binary one-hot vectors – e.g., the Euclidean distance between two word vectors provides a measure of the semantic similarity of the words.

Our vocabulary is the set of all words found in the *Iliad* and the *Odyssey*. We learn GloVe word embeddings of dimension 50 from the text of both poems.

## 6 Experiments

For our first model, we use the OpenNMT-py library [5]. The encoder RNN is a 2-layer bidirectional LSTM with encoding dimension of 200 (i.e. the forward and backward encoding each have dimension 100). The decoder RNN is also a 2-layer bidirectional LSTM but with dimension 500. We use an Adam optimizer with dropout probability of 0.3 and an initial learning rate of 0.01. The model was trained for 64,000 iterations at which point the accuracy on the validation set stopped improving.

For our second model, we use the code provided by the author [1]. We used an encoder with dimension of 100 (i.e. the forward and backward encoding each have dimension 50). The context RNN has encoding dimension 500. The model uses an Adam optimizer with initial learning rate of 0.002. The model was trained for 25,000 iterations and the accuracy was evaluated on the validity set every 2,500 iterations.

Once trained, we then applied both models to each of the 30 lines in the test set. For the first model, the input is the selected line of poetry (e.g. line 4 as shown in Figure 3) from which the model generates a new line of poetry (line 5). For the second model, the input is the previous **two** lines of poetry (e.g. line 3 and line 4) from which the model generates a new line of poetry (line 5). In this case, the previous two lines of poetry form a short context that is used to predict the next line in the poem. (Note that in this case we remove the previous two lines of poetry from the training set).

We assess the first model in two ways: (1) a quantitative metrical evaluation, and (2) an overall evaluation by Classicists to determine how well the generated lines fit into their original passages. We then provide a qualitative analysis of the lines generated by the second model and discuss the strengths and weaknesses of the model.

### 6.1 Parameter Experimentation

The first model, the basic encoder-decoder RNN, provided many options for tuning the model both at training time and at generation time. At training time, we experimented with two different functions for generating probabilities over the target vocabulary. At generation time, we experimented with different ways of sampling words from the vocabulary and different ways of controlling the length of the generated line of poetry. These parameters are explained in greater detail below.

*Generator Functions* We experimented with both a softmax and sparsemax function to generate probabilities over the target vocabulary at training time. The softmax function maps the output of the decoding process, a real-valued vector, to a normalized probability distribution over the vocabulary. In this case, all words in the vocabulary have non-zero probability of being generated. The sparsemax function [19] projects the output of the decoding process, again a real-valued vector, onto the K-dimensional probability simplex (where K is the size of the vocabulary). The authors note that this projection is likely to fall near the border of the simplex producing a probability distribution where many words have zero probability. Ideally, the sparsemax function would concentrate probability mass on the likeliest next words in the poem thereby improving the overall metrical and semantic quality of the generated lines.

*Random Sampling* At generation time, the decoder generates each word by sampling from a probability distribution defined over the vocabulary. We experimented with 3 different schemes for sampling from this distribution: sampling from among the $k$ words with highest probability (for $k = 2, 5, 10$) or sampling from the full probability distribution. Sampling from among the $k$ words with highest probability reduces the chance of randomly sampling a low-probability word.

*Length Penalty* Finally, we experimented with three options for controlling the length of the generated line of poetry: (1) no length penalty, (2) average length penalty, and (3) the length penalty described by [30]. The average length penalty divides the log probability of the words in the generated output by the number of words, thereby penalizing longer sentences. Although we cannot force the model to generate lines consisting of exactly six metrical feet, the length of the generated line of poetry (i.e. the number of words) is correlated with the metrical accuracy.

### 6.2 Results of Parameter Experimentation

To determine which parameters resulted in the best output, we generated five lines of Homeric poetry using each combination of parameters (detailed in the section above). The following metrics were used to evaluate the quality and accuracy of the meter of the generated lines:

1. The number of lines (out of 5) that had six total feet.
2. The number of lines (out of 5) that had an anceps.
3. The number of lines (out of 5) that were in perfect dactylic hexameter.
4. The percentage of feet in all 5 lines that were scannable (i.e. a correct dactyl, spondee, or anceps).

Table 1 shows the results of our parameter experiments. A random sampling value of $k = -1$ indicates sampling from the full distribution. All models were trained with an Adam optimizer and a starting learning rate of 0.01.

Overall, the softmax function outperforms the sparsemax function, performing better in terms of percent of correct feet and the number of perfect lines.

Interestingly, none of the other parameters showed any impact on the performance of the model. As such, we chose the most general parameter settings.

Our final model uses the softmax function with no length penalty and random sampling from the full distribution. We trained this model for 64,000 steps. Training was stopped when the validation accuracy stopped improving. The final model accuracy on the validity set was 97.77%, with perplexity of 1.08 and cross-entropy loss of 0.08. This model was used to produce the results for the remainder of the paper.

| Model | Len. Penal. | Rand. Samp. | Lines with 6 Feet | Lines with Ancipites | % Feet Correct | Perfect Lines |
|---|---|---|---|---|---|---|
| Softmax | None | -1 | 4 | 5 | 93.5 | 3 |
| | | 2 | 4 | 4 | 90.6 | 2 |
| | | 5 | 2 | 5 | 100 | 2 |
| | | 10 | 3 | 5 | 89.2 | 2 |
| | Wu | -1 | 4 | 5 | 93.5 | 3 |
| | | 2 | 4 | 4 | 90.6 | 2 |
| | | 5 | 2 | 5 | 100 | 2 |
| | | 10 | 3 | 5 | 89.2 | 2 |
| | Avg | -1 | 4 | 5 | 93.5 | 3 |
| | | 2 | 4 | 4 | 90.6 | 2 |
| | | 5 | 2 | 5 | 100 | 2 |
| | | 10 | 3 | 5 | 89.2 | 2 |
| Sparsemax | None | -1 | 2 | 5 | 88.9 | 1 |
| | | 2 | 4 | 5 | 93.1 | 3 |
| | | 5 | 2 | 5 | 88.9 | 1 |
| | | 10 | 2 | 5 | 88.9 | 1 |
| | Wu | -1 | 2 | 5 | 88.9 | 1 |
| | | 2 | 4 | 5 | 93.1 | 3 |
| | | 5 | 2 | 5 | 88.9 | 1 |
| | | 10 | 2 | 5 | 88.9 | 1 |
| | Avg | -1 | 2 | 5 | 88.9 | 1 |
| | | 2 | 4 | 5 | 93.1 | 3 |
| | | 5 | 2 | 5 | 88.9 | 1 |
| | | 10 | 2 | 5 | 88.9 | 1 |

Table 1: Results of parameter experimentation.

## 7 Results

In this section, we describe the results of both models. For the basic encoder-decoder RNN, we present (1) a quantitative metrical evaluation, and (2) an overall evaluation by Classicists to determine how well the generated lines fit into their original passages. For the second HRED model, we provide a qualitative analysis by the author of this paper (a Classicist) and discuss the strengths and weaknesses of the model.

### 7.1 Metrical Evaluation

All 30 machine-generated lines in the test set are presented in the Appendix along with the number of feet in the line, of those feet how many are correct (i.e. correct dactyls, spondees, or anceps), whether the line ends in an anceps, and whether the line is in perfect dactylic hexameter.

The 30 machine-generated lines ranged from a minimum of 5 feet to a maximum of 8 feet. All of the generated lines ended in an anceps. Of the total *feet* in the 30 generated lines, 93% were correct dactyls, spondees, or anceps. Of the total generated *lines*, 46% were in perfect dactylic hexameter.

Of the the 53% of lines which contained a metrical error, approximately half (56%) contained 6 feet but had at least one foot that was unscannable and the other half (44%) did not contain 6 feet.

Of the 7% of feet that were metrically incorrect, 67% were incorrect because the syllables could not scan correctly. For example, consider the second foot of line 10 in Table 4, which is the word ἵνα. The word ἵνα may only be scanned as two short syllables. However, two short syllables cannot form a proper foot in dactylic hexameter. The remaining 33% of the metrically incorrect feet were incorrect because they contained only one syllable; that is, the line scanned properly except for one extra syllable. For example, consider line eight in Table 4, which scans spondee-dactyl-spondee-dactyl-incorrect-anceps. The penultimate foot of line eight is the word ὡς, a one-syllable word that cannot form an entire foot.

Metrical analysis suggests that the presence of spondees increases the chance that a line will be metrically inaccurate. If we do not consider anceps, 81% of the feet within the metrically correct lines were dactyls, and only 19% were spondees. Conversely, only 53% of the feet within the metrically *incorrect* lines were dactyls. We also noted that most metrical errors occurred towards the middle of the line. The fourth foot was most likely to contain a metrical error.

An example of a generated line with perfect meter is:

ῥηιδίως δαναῶν ἐπιβαινέμεν ἀλλὰ καὶ αυτω

which is composed of a dactyl, dactyl, dactyl, dactyl, dactyl, anceps. Notably, this line demonstrates a phenomenon known as *epic correption* between the last two feet. Epic correption is when a syllable that is typically a long syllable becomes short because the next word begins with a vowel. In this example, epic correption occurs between καὶ and αυτω. This line can be translated as "But the son of Danaoi easily having climbed up, even he..." This is typical syntax for ancient Greek poetry, which often uses the structure of "But X having happened, Y..." where the action is finished on the next line.

An example of a generated line with poor meter is

ὀφθαλμοῖσι τε κατείρυσται καὶ ἐπαρτέες εἰσὶν ἑταῖροι

which begins with a spondee, then contains a dactyl, and after the second foot becomes unscannable. The translation is also poor, but can be roughly rendered as "both the eyes provoked[3] and enemies are equipped."

### 7.2 Evaluation by Classicists

To evaluate how well the generated lines fit into the original passages from which they were chosen, we created a survey that displayed the 30 5-line passages of Homeric poetry. Evaluators were instructed that each passage contained *at most* one machine-generated line of poetry (i.e. a passage may not contain any machine-generated lines). The survey contained 27 passages with a machine-generated line and 3 passages with no machine-generated line. For each of the 30 passages, evaluators were asked to identify which line (if any) was machine generated. If they identified a line as machine generated, they were further asked to mark why. There were a total of 10 evaluators all of whom are Classics graduate students. Evaluators were paid $10.00 each for evaluating all 30 passages. An example of a passage from the survey is shown in Figure 4.

We first present the results on a passage level. On the passage level, there are 300 total evaluation instances (10 evaluators and 30 passages). In the contingency table shown in Table 2, a true positive occurs when an evaluator tags any line in a passage as machine-generated and that passage contained a machine-generated line. A true positive does not require that an evaluator selected the correct line as machine generated. A true negative occurs when an evaluator correctly states that a passage contains no machine-generated lines. A false positive occurs when an evaluator tags a line in a passage as machine generated, but that passage contains no machine-generated lines. A false negative occurs when an evaluator states a passage contains no machine-generated lines, but the passage does contain a machine-generated line.

[3] "Provoked" is conjugated incorrectly and eyes cannot be the subject of the verb.

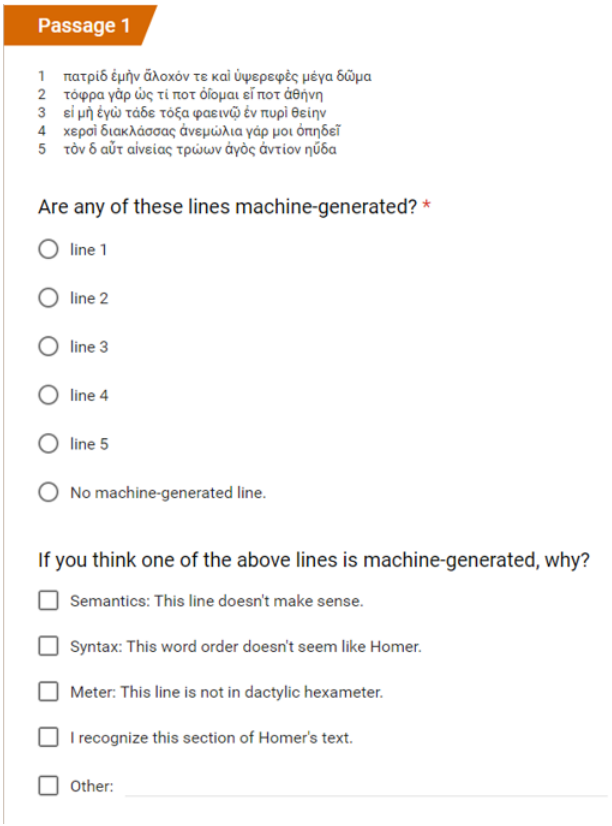


Figure 4: A sample passage from the survey

| | Tagged As MG | Tagged No MG |
|---|---|---|
| Contained MG | 206 | 64 |
| Not Contain MG | 11 | 16 |

Table 2: Contingency table for **passage** evaluation

75% of the time evaluators correctly determined if a passage contained a machine-generated line or not [4]. The precision and recall were 94% and 76% respectively with an $F_1$ score for passage-level evaluation of 0.836. Overall, when an evaluator identified a passage as containing a machine-generated line, they

[4] Although they did not necessarily correctly identify which line was machine generated.

were almost always correct. However, such passages were identified only 76% of the time.

We now present our results at the line level. We consider only results for the 27 passages which contained a machine-generated line. At the line level, there are 1,350 total instances (10 evaluators · 27 passages · 5 lines). Note that these instances are not independent because evaluators were only able to choose a maximum of 1 in every 5 lines. In the contingency table shown in Table 3, a true positive occurs when an evaluator tagged a line as machine-generated and the line was machine generated. A true negative occurs when an evaluator indicated a line was not machine-generated (i.e. they did not mark the line in the survey) and the line was not machine generated. A false positive occurs when an evaluator tagged a line as machine generated and it was not. A false negative occurs when evaluators did not mark a line as machine generated, but the line was machine generated.

| | Tagged As MG | Tagged No MG |
|---|---|---|
| MG Line | 171 | 99 |
| Not MG Line | 35 | 1045 |

Table 3: Contingency table for **line** evaluation

90% of the time evaluators correctly identified a line as machine generated or not. Note that randomly guessing one of the six options (either choosing one of the five lines or "No machine-generated line") would result in an accuracy of 16.67%. The precision and recall were 63% and 83% respectively with an $F_1$ score for line-level evaluation of 0.71.

Evaluators were also asked to mark the reason why they thought a line was machine generated. They were given five options: (1) Semantics: This line doesn't make sense, (2) Syntax: This word order doesn't seem like Homer, (3) Meter: This line is not in dactylic hexameter, (4) I recognize this section of Homer's text, and (5) Other. Evaluators were allowed to select more than one option. If the evaluator marked "Other," they were asked to type in their own response. Figure 5 below shows the reasons that evaluators marked a line as machine-generated when the line was in fact machine-generated. A majority of the time, evaluators correctly marked a line as machine-generated because semantically the line did not make sense.

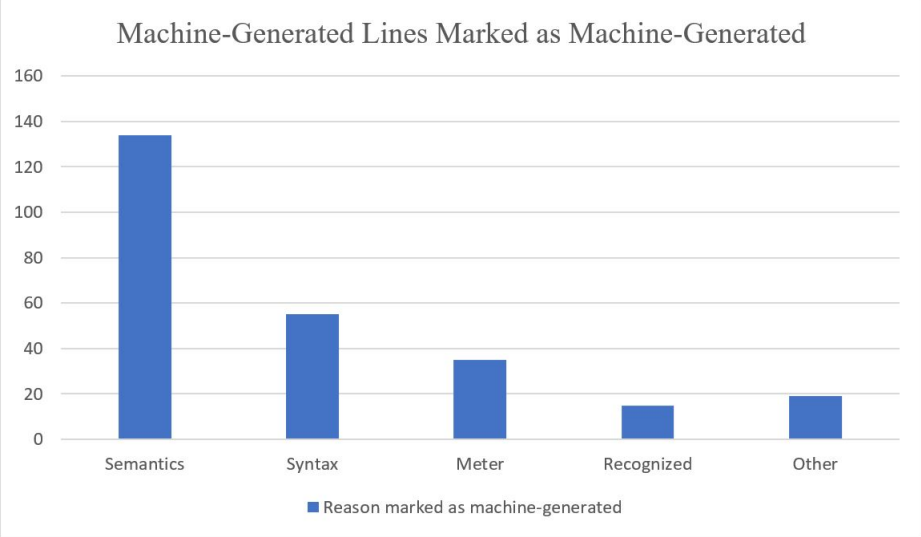


Figure 5: Reasons evaluators selected for why they believed a line was machine-generated.

When evaluators marked their reasoning as "Other," they often presented reasoning related to context – i.e. how the line "fit" with the surrounding lines of poetry. For example, one evaluator writes "[s]emantically, it doesn't seem to necessarily fit with the other lines of text. It may be true Homer, while one of the surrounding lines is machine-generated." Other evaluators write "just guessing but it doesn't seem to fit," "The line doesn't make sense in this context," "Doesn't fit with rest of passage," "Doesn't seem to fit," and "Doesn't fit with rest of text." Another evaluator notes a machine-generated line "seems to interrupt meaning between lines 3 and 5."

Another common thread in "Other" responses is evaluators' insider knowledge of Homeric poetry being used to identify machine-generated lines even when those lines were semantically and metrically sound. For example, one evaluator notes that "[l]ines ending in glaukopis Athene tend to be self-contained units where Athena performs the action of a verb." Here, the student was only able to identify the line as machine-generated because they were aware of how the Greek formula *glaukopis Athene* is used in Homeric poetry. Another evaluator writes "almost never see 'pallas' by itself, normally is 'pallas athene.' In this case, the evaluator was able to tell the line was machine-generated because of their familiarity with the Homeric formula *Pallas Athene*.

Other times, evaluators were able to identify machine-generated lines because of their knowledge about Homeric characters roles within texts. For example, consider the lines below:

αὐτὰρ ἐπεὶ δὴ δούρατ ἀλεύαντο μνηστήρων
ἷξε τόθ ἔσαν οἶκον δὲ καλυψώ δῖα θεάων

But when they had avoided the spears of the suitors,
He came then to the house of Calypso heavenly among
goddesses.

The first line is true Homeric poetry from Homer's *Odyssey*. The second line is machine-generated. In the *Odyssey*, the suitors (μνηστήρων) are a group of characters who never come in contact with the character Calypso (καλυψώ). Evaluators noticed this. They wrote reasons such as "Calypso should not be involved in this context with suitors" and "Calypso and the suitors? I'd watch that show, but nope!"

The same type of error occurs in a second passage involving the characters Penelope and Hector. Penelope is a character who remains on the island of Ithaca for the duration of the storyline told by Homer's *Iliad* and *Odyssey*. Hector is a character who remains at Troy during most of the *Iliad* and is killed partway through the epic. These two characters clearly could not have ever occupied the same location at any point in the Homeric canon. However, a line including Hector is generated following a line about Penelope. In the below lines, the first three are Homeric poetry while the last line is machine-generated:

θηρσὶ καὶ οἰωνοῖσιν ἕλωρ γένετ οὐδέ ἑ μήτηρ
οὐδ ἄλοχος πολύδωρος ἐχέφρων πηνελόπεια
ἕκτορα ἄντα κατ ὄσσε παρίστατο δάκρυ χέουσα

Nor did his mother deck him for burial and weep
over him,
Nor his father, we who gave him birth, no, nor did
his wife,
wooed with many gifts, constant Penelope.
Face-to-face with Hector she stood and both eyes shed
tears.

The reasons graduate students provided for identifying the latter line as machine-generated included "Hector should not be involved in this context" and "Get out of here, Hector."

Sometimes evaluators marked true lines of Homer as machine-generated. Figure 6 shows the reasons graduate student provided for marking a line as machine-generated when the line was not machine-generated.

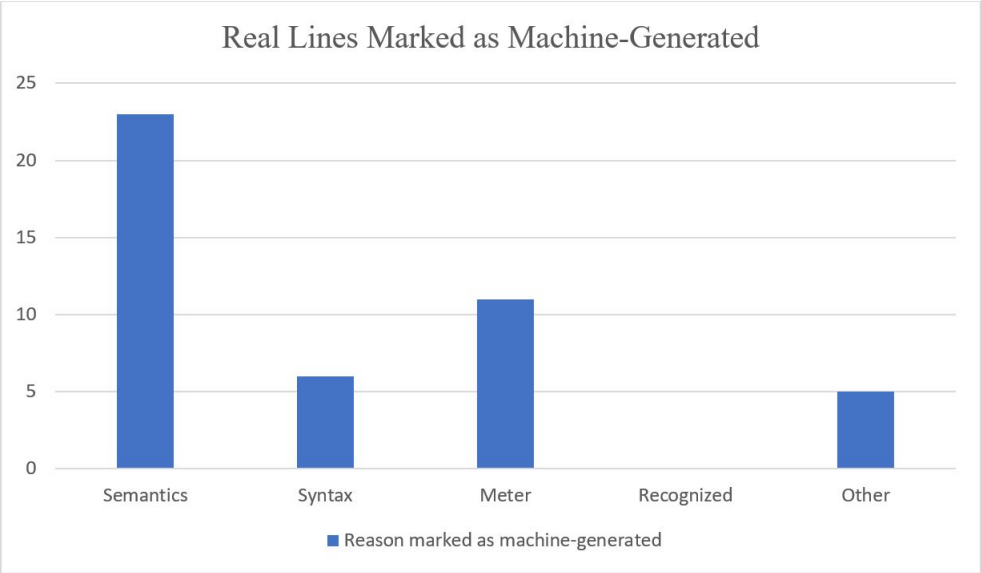


Figure 6: Reasons evaluators selected for why they believed a line was machine-generated.

The most common reason that a true line of Homeric poetry was marked as machine-generated was, again, semantics. This reasoning has no correlation to the placement of the machine-generated line in the passage being placed immediately before or after the line selected by the evaluator. Evaluator's confusion was likely because they were reading passages out of context.

Evaluators also marked meter as the reason they believed a real line was machine-generated. The comments left in the "Other" category suggest that evaluators may have marked "Meter" because the Homeric line contained rare (but correct) metrical patterns. For example, an evaluator marked this line of Homeric poetry as machine-generated:

ὣδε δέ τις εἴπεσκεν ἀχαιῶν τε τρώων τε

The above line is scanned dactyl-spondee-dactyl-spondee-spondee-anceps. This evaluator commented that "it is rare for a spondee to appear in the fifth foot of dactylic hexameter." The presence of a spondee in the penultimate foot of a hexmetric line is unusual, although not unheard of. The presence of a spondee in the antipenultimate foot also makes the penultimate spondee more common and characterizes the line as 'spondeic.'

In general, the evaluators were able to identify machine-generated lines of Homeric poetry primarily for semantic and syntax reasons. If we consider only the lines identified by evaluators as machine-generated based upon metrical reasons, the evaluator accuracy drops to 12.96%. Based on these results, we conclude that we are able to successfully generate poetry in dactylic hexameter.

### 7.3 Evaluation of the Hierarchical Recurrent Encoder Decoder Model

For the second model, HRED, we present a qualitative analysis of the 30 generated lines from the test set. All 30 machine-generated lines are given in the Appendix.

The most striking aspect of the 30 lines generated by HRED is that none of them come close to dactylic hexameter – an observation that one can make even without understanding the Greek, simply by comparing the lines in Table 4 and Table 5 in the Appendix. The lines generated by HRED are much longer and are easily identified as not belonging in the *Iliad* or the *Odyssey*.

We suspect that HRED fails to produce lines in dactylic hexameter because the decoder RNN is not conditioned on – i.e., it is not directly influenced by – the encoding of the input sentence. Instead, the decoder RNN is conditioned on, and thus directly influenced by, the encoding produced by the *context* RNN which is an aggregation of multiple input sentences. Thus, the strict meter of the input exerts a weak and indirect influence on the generated output.

Semantically, however, the lines generated by HRED are overall more coherent than those generated by the first model. Consider the first machine-generated line:

> ἵπποισιν καὶ ὄχεσφιν παρὰ νηῶν ἂψ ἀπονοστήσειν προτὶ ἴλιον ἠνεμόεσσαν
> πρόσθεν γάρ μιν μοῖρα δυσώνυμος ἀμφεκάλυψεν

This can be translated as:

> They went away with horses and carriages alongside the ships towards windy Ilium, for previously [powerful] fate enfolded them.

In this example, we see that HRED did quite well choosing semantically related words. For example, it refers to Ilium as a windy city and it connects the idea of Fate as a powerful force [5] – all concepts that show up in the *Iliad* and the *Odyssey*. Furthermore, it is able to choose related nouns such as “horses" and “carriages" and to recognize that ships travel *toward* a geographic destination.

Another interesting property of the lines generated by HRED is that many of them contain *preposition stacking*. Homeric poetry often contains multiple prepositional phrases as shown in the example above: “with horses", "alongside the ships", "towards...Ilium". In contrast, however, the lines generated by HRED contain fewer *stacked participles* – e.g., “having gone away", “having boarded the ship" – favoring more concrete, aorist (i.e. simple), or present tense verbs.

Finally, many of the lines lose coherence towards the end. For example, the third generated line in the test is:

> πόλλ ὀλοφυρόμενος σχεδόθεν δέ οἱ ἦλθεν ἀθήνη ἀνδρὶ δέμας νέῳ μήλων
> οἷοί τε ἀνάκτων παῖδες ἔασι

[5] The phrase μοῖρα δυσώνυμος which was translated as “powerful fate" is syntactically incorrect. The word δυσώνυμος has the incorrect gender.

This can be translated as:

> Lamenting much, Athena went nearby to the young man in-such-a-way...of sheep...Children of the gods, such as these, are...

Here, we are using the ellipsis to denote disconnected phrases. This sentence begins coherently but it eventually devolves into a series of unrelated phrases. This was a common pattern in many of the lines generated by HRED.

Overall, it is clear that the strength of the HRED model lies in its ability to produce semantically coherent lines of poetry. However, this seems to come at the cost of metrical accuracy.

## 8 Conclusion

In this paper, we investigated the use of encoder-decoder RNNs for generating metrically accurate Homeric poetry. We experimented with two models: a basic encoder-decoder RNN and the hierarchical recurrent encoder-decoder model. The basic encoder-decoder RNN model was able to produce metrically accurate poetry however it under-performed in terms of generating semantically cohesive lines of poetry that fit the broader context of the narrative. In contrast, the HRED model produced more semantically coherent lines of poetry but was completely unable to reproduce dactylic hexameter.

We hypothesize that this stark difference between the two models is a result of the type of connection that exists between the input (i.e. the input line of poetry) and the generated output. In particular, the former model directly connects the encoding of the input sentence to the generation of the output sentence, ostensibly resulting in a stronger reproduction of dactylic hexameter. The latter model instead chooses to aggregate successive inputs into a global *context* encoding that is connected to the generation of the output sentence. This context encoding loses the finer detail of the meter but is better able to summarize and keep track of the semantic topic.

The models are similar in lesser ways – e.g., both models are able to capture unique aspects of Homeric poetry such as epic correption or stacked prepositions.

## 9 Future Work

In the future, we would like to focus on creating an encoder-decoder model that leverages the strengths of both the basic encoder-decoder RNN and the HRED model. In particular, developing a model that connects the encoding of the input sentence *and* the context encoding to the decoder so that the generated line of poetry is directly influenced by both.

We would also like to train a basic encoder-decoder RNN that is able to look at the previous *two* lines of poetry in a similar manner to HRED. This would allow us to determine if HRED is able to produce more semantically coherent lines of poetry because of the context encoder or simply because it sees more preceding lines of poetry.

In addition, for both models, there is no explicit mechanism for constraining where a Homeric character can appear in a passage. As such, we are interested

in developing a model that allows us to control where characters appear within the poem. Since an encoder-decoder RNN produces a probability distribution for the next word in the poem, we plan to investigate methods for biasing these distributions towards words that are likely given both the location of the line within the poem as well as the location of the word in the line.

## 10 Appendix

Below are two tables showing the 30 generated lines for the basic encoder-decoder model and the hierarchical recurrent encoder-decoder.

| Number | Generated Line | Feet | Correct Feet | Anceps | Perfect line |
|---|---|---|---|---|---|
| 1 | τυδείδῃ δ ἄρα θυμὸν ἐνὶ στήθεσσιν ὄρινε | 6 | 6 | Y | Y |
| 2 | νηὶ θοῇ ἐπίηρα πόδες τέρας οὐδέ τις ὕπνος | 6 | 6 | Y | Y |
| 3 | τόφρα γὰρ ὡς τί ποτ ὀίομαι εἴ ποτ ἀθήνη | 5 | 5 | Y | N |
| 4 | ῥηιδίως δαναῶν ἐπιβαινέμεν ἀλλὰ καὶ αὖτως | 6 | 6 | Y | Y |
| 5 | ὀφθαλμοῖσι τε κατείρυσται καὶ ἐπαρτέες εἰσὶν ἑταῖροι | 8 | 7 | Y | N |
| 6 | πρὶν τοιόσδ ἐστι διαμπερές αὐτὰρ ἀχαιοὶ | 6 | 5 | Y | N |
| 7 | ἐλέησεν ἀχαιοῖσιν πυρὸς θέτις θεοειδής | 6 | 4 | Y | N |
| 8 | ἴσχειν ἐν μεγάροισιν ἀκούσαμεν ὡς υἱόν | 6 | 5 | Y | N |
| 9 | αὐτίκ ἄρ ἐγγύθεν ἔσκε προσηύδα θοῦρον ἄρηα | 7 | 6 | Y | N |
| 10 | εἴσατο ἵνα μή με παρέστη ἄλις ὣς τιμήσαντο | 7 | 6 | Y | N |
| 11 | ὣς ἄρα μὲν μάλα πάντες ἀεικέα μήδετο ἔργα | 6 | 6 | Y | Y |
| 12 | ἀλλ ἄγε δὴ στέωμεν καὶ ἀλεξώμεσθα μένοντες | 6 | 5 | Y | N |
| 13 | ἐλθόντ εἰς ἴδην ὅθι πάσχετε παλλὰς ἀνώγει | 6 | 6 | Y | Y |
| 14 | νηυσὶν ἐπ ὠκεανοῖο μάχην τελαμωνιάδαο | 7 | 6 | Y | N |
| 15 | καὶ τοὺς ἄλλῃ ἅμα μνηστῆρας ἔργου | 5 | 5 | Y | N |
| 16 | ὣς ὁ νηὸς ἄγοντες ἐρεικόμενος περὶ δουρί | 6 | 6 | Y | Y |
| 17 | ἦ ἴδον ἠέ τι μᾶλλον ἐμεῦ πολὺ κέρδιον εἴη | 6 | 6 | Y | Y |
| 18 | τοῖς ἐπεί τοι πρὸς νῆας ἀρίστη φαίνετο βουλή | 6 | 6 | Y | Y |
| 19 | ἀλλ ἤδη τρώεσσιν πειρήσομαι ἀλεγεινῆς | 6 | 4 | Y | N |
| 20 | αἵματός εἰς ἀγαθοῖο φίλον τέκος οἵ ἀγορεύεις | 6 | 6 | Y | Y |
| 21 | δεῖξε καὶ ὤμοιιν πέλαγος προτὶ ἄστυ φέρεσκον | 6 | 6 | Y | Y |
| 22 | τόφρ εἴ τις πρῶτα θεῶν ἐθέλωμι τέτυκται | 6 | 5 | Y | N |
| 23 | αὐτὰρ ἐγὼ δίχα πάντας ἐυκνήμιδας ἑταίρους | 6 | 5 | Y | N |
| 24 | ἀργείων ὕπατον αἴαντα ταχὺν καὶ χεῖρας ὕπερθεν | 7 | 6 | Y | N |
| 25 | ἀλλ ἄγε δὴ οἶκον δὲ μετ αὐτοῦ μίμν ἐθέλῃσιν | 6 | 6 | Y | Y |
| 26 | ἀλλ ἄγ ἐμοὶ δότε τόξον ἐύξοον ὄφρα γένηται | 6 | 5 | Y | N |
| 27 | ὡς δ ἄγε τηλέμαχος μάλα γὰρ γλαυκῶπις ἀθήνη | 6 | 6 | Y | Y |
| 28 | τροίῃ φεύγων ὁ δ ἄντρου οἵσιν ἔθηκαν | 6 | 5 | Y | N |
| 29 | ἶξε τόθ ἔσαν οἶκον δὲ καλυψώ δῖα θεάων | 6 | 6 | Y | Y |
| 30 | ἕκτορα ἄντα κατ ὄσσε παρίστατο δάκρυ χέουσα | 6 | 6 | Y | Y |

Table 4: Thirty machine-generated lines from the encoder-decoder model.

| Number | Generated Line |
|---|---|
| 1 | ἵπποισιν καὶ ὄχεσφιν παρὰ νηῶν ἆψ ἀπονοστήσειν προτὶ ἴλιον ἠνεμόεσσαν πρόσθεν γάρ μιν μοῖρα δυσώνυμος ἀμφεκάλυψεν |
| 2 | χῶρον μὲν πρῶτον αὐτὰρ ἔπειτα κλήρους ἐν κυνέῃ χαλκήρει πάλλον ἑλόντες ὁπ- πότερος δὴ πρόσθεν ἀφείη χάλκεον ἔγχος |
| 3 | πόλλ ὀλοφυρόμενος σχεδόθεν δέ οἱ ἦλθεν ἀθήνη ἀνδρὶ δέμας νέῳ μήλων οἷοί τε ἀνάκτων παῖδες ἔασι |
| 4 | οὐδ ἄλοχος πολύδωρος ἐχέφρων πηνελόπεια ἐν λεχέεσσιν ἐὸν πόσιν ὡς ὀφθαλμοὺς τὸ γὰρ γέρας ἐστὶ θανόντων |
| 5 | ὅθι πλεῖστα δόμοις ἐν κτήματα κεῖται ὃς μενελάῳ δῶκε δύ ἀργυρέας ἀσαμίνθους δοιοὺς δὲ τρίποδας δέκα δὲ χρυσοῖο τάλαντα |
| 6 | δ ἀνὰ θυμὸν ἅ περ τελέεσθαι ἔμελλον τοῖσι δὲ καὶ μετέειπε γέρων ἥρως ἀλιθέρσης μαστορίδης ὁ γὰρ οἷος ὁμηλικίην ἐκέκαστο |
| 7 | ὣς ἐρέουσιν ἐμοὶ δὲ τότ ἂν πολὺ κέρδιον εἴη ἄντην ἢ ἀχιλῆα κατακτείναντα νέεσθαι ἠέ κεν αὐτῷ ὀλέσθαι πρὸ πόληος |
| 8 | δαῖτα δώσει δέ μοι ὅς κ ἐθέλῃσιν οὐ γὰρ ἐπὶ σταθμοῖσι μένειν ἔτι τηλίκος εἰμί ὥστ πάντα πιθέσθαι |
| 9 | λυσόμενος παρὰ σεῖο φέρω δ ἀπερείσι ἄποινα ἀλλ αἰδεῖο θεοὺς ἀχιλεῦ αὐτόν τ ἐλέησον μνησάμενος σοῦ πατρός ἐγὼ δ περ |
| 10 | ἵππους οἳ δὲ τάχ αὐτοὶ ἐπειγόμενοι περὶ νίκης ἐνθάδ ἐλεύσονται τότε δὲ γνώσεσθε ἕκαστος ἵππους ἀργείων οἳ δεύτεροι οἵ τε πάροιθεν |
| 11 | ἐμμεμαὼς ἐπόρουσεν ἐρυσσάμενος ξίφος ὀξὺ σμερδαλέα ἰάχων ὃ δὲ χερμάδιον λάβε χειρὶ αἰνείας μέγα ἔργον ὃ οὐ δύο γ ἄνδρε φέροιεν |
| 12 | ἔρχεσθε πρὸς δώμαθ ἵν αἰδοίη βασίλεια τῇ δὲ παρ ἠλάκατα δ αὐτὴν ἥμεναι ἐν μεγάρῳ ἢ εἴρια χερσίν |
| 13 | τυρῶν τέ μιν ἔνδον ἥμενοι ἦος ἐπῆλθε φέρε δ ὄβριμον ἄχθος ὕλης ἵνα οἱ ποτιδόρ- πιον εἴη |
| 14 | τῶν ἄλλων δαναῶν μετ ἀμύμονα πηλείωνα ἴθυσεν δὲ διὰ προμάχων συὶ εἴκελος ἀλκὴν καπρίῳ ὅς τ ἐν ὄρεσσι κύνας τ αἰζηοὺς |
| 15 | ἕκτορά τ ἀμφὶ μέγαν καὶ ἀμύμονα πουλυδάμαντα αἰνείαν θ ὃς τρωσὶ θεὸς ὣς τίετο δήμῳ τρεῖς τ πόλυβον καὶ ἀγήνορα δῖον |
| 16 | ὣς ἔφαθ ἡ δ ἐχάρη καὶ ἀπὸ λέκτροιο γρηὶ βλεφάρων δ ἀπὸ δάκρυον ἧκεν καί μιν φωνήσασ ἔπεα πτερόεντα προσηύδα |
| 17 | καὶ γάρ τίς θ ἕνα μῆνα μένων ἀπὸ ἧς ἀλόχοιο ἀσχαλάᾳ σὺν νηὶ ὅν περ ἄελλαι τε θάλασσα |
| 18 | ἔνθα μὲν ἑπτάετες μένον ἔμπεδον εἵματα δ αἰεὶ δάκρυσι τά μοι ἄμβροτα δῶκε καλυψώ ἀλλ ὅτε δὴ μοι ἐπιπλόμενον ἔτος ἦλθεν |
| 19 | δ ἄρα τούς γε οἵ τε καὶ ἄλλους ἀνθρώπους πέμπουσιν ὅτις σφέας εἰσαφίκηται καὶ τὰ μὲν εὖ κατέδησεν ὑπ αἰθούσῃ ἐριδούπῳ |
| 20 | ἀλλ οὐ γάρ τις πρῆξις ἐγίγνετο μυρομένοισιν ἀλλ ὅτε δή ῥ ἐπὶ νῆα θοὴν καὶ θῖνα θαλάσσης ἦομεν ἀχνύμενοι θαλερὸν κατὰ δάκρυ χέοντες |
| 21 | αὐτὰρ ἐπεὶ δὴ δούρατ μνηστήρων τοῖς δ ἄρα μύθων ἦρχε πολύτλας δῖος ὀδυσσεύς ὦ φίλοι ἤδη μέν κεν ἐγὼν εἴποιμι καὶ ἄμμι |
| 22 | λαιμὸν ἐπὶ οἷ τετραμμένον ὀξέι δουρί ἐκλίνθη δ ἑτέρωσε κάρη ἐπὶ δ ἀσπὶς ἐάφθη καὶ κόρυς ἀμφὶ δέ οἱ θάνατος χύτο θυμοραιστής |

| | |
|---|---|
| 23 | μεσσηγὺς νηός τε καὶ αὐτοῦ τηλεμάχοιο τὸν δὲ θεοκλύμενος ἑτάρων ἀπονόσφι καλέσσας ἔν τ ἄρα οἱ φῦ χειρὶ ἔπος τ ἔφατ ἔκ τ ὀνόμαζε |
| 24 | μή τι πατέρων γένος οἳ τὸ πάρος περ τ ἠνορέῃ τε πᾶσαν ἐπ αἶαν τὸν δ αὖ τηλέμαχος πεπνυμένος ἀντίον ηὔδα |
| 25 | ὀππότ ἐγώ περ ἴω μετὰ μῶλον ἄρηος αὐτὰρ ἐπεί κ ἀπὸ ναῦφι μάχην ἐνοπήν τε δίηται ἀσκηθής μοι ἔπειτα θοὰς ἐπὶ νῆας ἵκοιτο |
| 26 | μύθων τε ἔμεναι τε ἔργων ὡς ἂν ἔπειτ ἀπὸ σεῖο φίλον τέκος οὐκ ἐθέλοιμι οὐδ εἴ κέν μοι θεὸς αὐτὸς |
| 27 | πατρίδ ἐμὴν ἄλοχόν τε καὶ ὑψερεφὲς μέγα δῶμα αὐτίκ ἔπειτ ἀπ ἐμεῖο κάρη τάμοι ἀλλότριος φὼς εἰ μὴ ἐγὼ τάδε τόξα φαεινῷ ἐν πυρὶ |
| 28 | χειρί τέ μιν κατέρεξεν ἔπος τ ἔφατ ἔκ τ ὀνόμαζεν ἥ δὴ ἀλιτρός γ ἐσσὶ καὶ οὐκ εἰδώς οἷον δὴ τὸν μῦθον ἀγορεῦσαι |
| 29 | δὲ ὃ δ ἄρ τέκε παῖδα αὐτὰρ ἔμ ἀγχίσης πρίαμος δ ἔτεχ ἕκτορα δῖον ταύτης τοι γενεῆς τε καὶ αἵματος εὔχομαι εἶναι |
| 30 | ὦ γέρον ἦ μάλα δή σε νέοι τείρουσι σὴ δὲ βίη λέλυται χαλεπὸν δέ σε γῆρας ὀπάζει ἠπεδανὸς δέ νύ τοι θεράπων δέ τοι ἵπποι |

Table 5: Thirty machine-generated lines from the HRED model.

## Acknowledgments

The authors of this paper would like to thank the following people for their help evaluating machine-generated Homeric poetry: Emory B. Brigden from the University of Puget Sound; Konnor Clark, Edgar A. Garcia, and Megan O'Donald from the University of Washington; Rachel E. Dubit, Grace Erny, Nick Gardner, Dillon Gisch, and Brian D. Le from Stanford University; Kathryn H. Stutz from John Hopkins University.

## References

1. `https://github.com/julianser/hed-dlg`
2. Homeri Opera in Five Volumes. Oxford University Press (1920)
3. C. Zhou, W.Y., Ding, X.: Genetic algorithm and its implementation of automatic generation of chinese songci: Genetic algorithm and its implementation of automatic generation of chinese songci. Journal of Software **21**(3), 427 – 437 (2010)
4. D. Bahdanau, K.H.C., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473
5. G. Klein, Y. Kin, Y.D.J.S., Rush, A.: Opennmt: Open-source toolkit for neural machine translation. In: Proc. of ACL 2017, System Demonstrations. pp. 67 – 72 (2017)
6. Griswold, C.L.: Plato on rhetoric and poetry. In: Zalta, E.N. (ed.) The Stanford Encyclopeida of Philosophy (2016)
7. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735 – 1780 (1997)
8. Honoratus, M.S.: Servii Grammatici qui feruntur in Vergilii carmina commentarii. Cambridge University Press (2011)

9. I. Serban, A. Sordoni, Y.B.A.C., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. pp. 3776 – 3783 (2016)
10. J. He, M.S., Jiang, L.: Generating chinese classical poems with statistical machine translation models. In: Proc. of the Twenty-Sixth AAAI Conf. on Artificial Intelligence. pp. 1650 – 1656 (2014)
11. J. Pennington, R.S., Manning, C.: Glove: Global vectors for word representation. In: Empirical Methods in Nat'l Lang. Processing. pp. 1532 – 1543 (2014)
12. J.H. Lau, T. Cohn, T.B.J.B., Hammond, A.: Deep-speare: A joint neural model of poetic language, meter and rhyme. In: Proc. of the 56th Annual Meeting of the Assoc. for Compt'l Linguistics. pp. 1948 – 1958 (2018)
13. Jiang, L., Zhou, M.: Generating chinese couplets using a statistical mt approach. In: Proc. of the 22nd Int'l Conf. on Compt'l Linguistics. pp. 377 – 384 (2008)
14. K. Cho, B. van Merrienboer, C.G.D.B.F.B.H.S., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: Proc. of the 2014 Conf. on Empirical Methods in Nat'l Lang. Processing. pp. 1724 – 1734 (2014)
15. Lamar, A., Chambers, A.: Generating homeric poetry with deep neural networks. In: Proc. of the 1st Int'l. Conf. on Transdisciplinary AI (2019)
16. M. Ghazvininejad, X. Shi, Y.C., Knight, K.: Generating topical poetry. In: Proc. of the 2016 Conf. on Empirical Methods in Nat'l Lang. Processing. pp. 1183 – 1191 (2016)
17. M. T. Luong, H.P., Manning, C.: Effective approaches to attention-based neural machine translation. arXiv:1508.04025 (2015)
18. Manurung, H.M.: An Evolutionary Algorithm Approach to Poetry Generation. Ph.D. thesis, University of Edinburgh (2003)
19. Martins, A., Astudillo, R.: From softmax to sparsemax: A sparse model of attention and multi-label classification. In: Proc. of the 33rd Int'l. Conf. on Machine Learning. vol. 48, pp. 1614 – 1623 (2016)
20. N. Tosa, H.O., Minoh, M.: Hitch haiku: An interactive supporting system for composing haiku poem. In: Entertainment Computing – ICEC 2008. pp. 209 – 216 (2009)
21. N. Tosa, X.W., Nakatsu, R.: New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. In: Entertainment Computing – ICEC 2009. pp. 191 – 196 (2009)
22. Oliveira, H.G.: Poetryme: A versatile platform for poetry generation. In: Proc. of the European Conf. on A. I. 2012 Workshop on Compt'l Creativity, Concept Invention, and General Intelligence. pp. 1 – 21 (2012)
23. Porter, J.I.: Homer, skepticism, and the history of philology. In: Humphreys, S.C., Wagner, R.G. (eds.) Modernity's Classics, pp. pp. 261 – 292. Springer (2013)
24. Q. Wang, T. Luo, D.W., Xing, C.: Chinese song iambics generation with neural attention-based model. In: Intl. Joint Conf. on Artificial Intelligence. pp. 2943 – 2949 (2016)
25. R. Manarung, G.R., Thompson, H.: Using genetic algorithms to create meaningful poetic text. Journal of Experimental  Theoretical Artificial Intelligence pp. 43 – 64 (2010)
26. R. Yan, H. Jiang, M.L.S.L.X.L., Li, X.: i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In: Proc. of the Twenty-Third Intl' Joint Conf. on Artificial Intelligence. pp. 2197 – 2203 (2013)

27. Robin Waterfield, T.: Plato Republic. Oxford University Press (1993)
28. Sandys, J.E.: History of Classical Scholarship From the Sixth Century B.C. to the End of the Middle Ages. Cambridge University Press (1903)
29. Schuster, M., Paliwal, K.: Bidirectional recurrent neural networks. IEEE Transaction on Signal Processing **45**(11), 2673 – 2681 (1997)
30. Y. Wu, M. Schuster, Z.C.Q.V.L., Norouzi, M.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144
31. Zhang, X., Lapata, M.: Chinese poetry generation with recurrent neural networks. In: Proc. of the 2014 Conf. on Empirical Methods in Nat'l Lang. Processing. pp. 670 – 680 (2014)