POETRY THROUGH PROPOGATION, GENERATING HAIKUS WITH DEEP LEARNING RECURRENT NEURAL NETWORKS

Evan Banerjee
Student# 1009682309
evan.banerjee@mail.utoronto.ca

Diego Ciudad Real Escalante
Student# 1009345308
diego.ciudadrealescalante@mail.utoronto.ca

Noah Monti
Student# 1009452398
noah.monti@mail.utoronto.ca

Ji Hong Sayo
Student# 1007314728
ji.sayo@mail.utoronto.ca

ABSTRACT

This is our progress report for the APS360 Final Project. Total Pages: 9

1 INTRODUCTION

2 ILLUSTRIATON

The high-level description of the model can be seen below in 1 . The model consists of Gated Recurrent Unit that is trained on poems sourced from the web. These poems also build up the dictionary of words used by the model to generate haikus. Finally, the diagram also shows the expected usage of this model. A user would prompt the model with the starting words of a poem, press enter, and the model would generate a haiku with the prompt as its starting point.

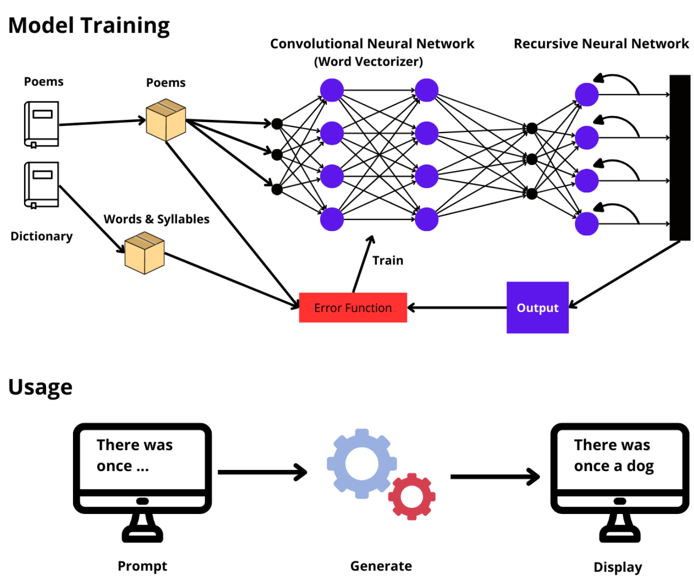

Figure 1: Top-Level Model Description

3 BACKGROUND

4 PROJECT DESCRIPTION

Project Description "Genuine poetry can communicate before it is understood" —T.S. Eliot In this project we are working to build a deep learning model to generate short poems of various topics. Our model is a haiku generator which can be prompted with the first line of a haiku and generate the content of subsequent lines, built on an LSTM (long-short-term-memory) architecture. Poetry has long been regarded as a deep expression of emotion and human experience. Given that the model we construct will possess neither emotions nor intuition, we are interested to see whether our construction will be able to imitate human poetry, and if so to what degree. If the model is capable of producing compelling poems as output, this would contradict the idea that poetry requires emotion to produce, with implications for our understanding of the nature of creative work in general.

Several ML approaches can be taken to text-generation. For instance, we compare our model to markov chains as a baseline. However, deep learning offers advantages that other statistical approaches cannot. As compared to simpler statistical models, neural nets can learn more abstract and general characteristics in the poems used for training. Given that a hallmark of good poetry is abstraction and unity around a sustaining theme, we believe deep learning is uniquely well-suited for the problem of poetry generation.

5 INDIVIDUAL CONTRIBUTIONS AND RESPONSIBILITIES

Thus far our team has been performing effectively—we have been able to meet both internal and external deadlines. Currently, we've been collaborating over github, with most team members working in jupyter notebooks we upload to the shared github page for review. Team communication takes place over whatsapp, and both systems have been working well thus far.

On a technical level, we are quite pleased with our progress thus far, which has been facilitated by the fact that training on text is less memory-intensive than training on images, which allows for more rapid prototyping without consuming very large amounts of computing power. We have already implemented both 'baseline' and prototype models. What remains is hyperparameter tuning, improvements to the model and testing, all of which we feel will be completed well within the timeframe of the project. In summary, we are on track to meet the goals outlined in our proposal, and may even implement some additional testing and functionality beyond what was outlined in our initial proposal.

5.1 REDUNDANCIES

The most crucial remaining aspect of our project is training the finalized model with a consolidated dataset. This is also a potentially sensitive step, as connection losses or malfunctions could result in significant investments of time and computational resources being wasted. Given this, we plan to have two team members responsible for training. If the primary team member runs into difficulties, they will quickly alert the team via WhatsApp, so that the secondary group member can take over and begin training on a separate system, thus reducing the risk of a technical or personal problem derailing the project.

Several of our group members have separately implemented or are working to implement different model architectures. While we have a consensus on which architecture we plan to use for the final model, this experimentation provides a strong element of redundancy. If unexpected issues emerge with the architecture we've agreed on, we can discuss as a group and fall back on one of the experimental models.

5.2 TEAM CONTRIBUTIONS AND COMMUNICATION

As shown in the task tracker (Figure **??**) and the communications table (Figure **??**), all team members have been responsive and met the necessary deadlines for their work.

6 NOTABLE CONTRIBUTION

6.1 DATA PROCESSING

The main data used for this project consists of the Gutenberg Poetry Corpus dataset in github (biglam, 2022). This dataset is publicly available and comes from a web scraper run in 2018. The web scraper collected poems from the Gutenberg Project, and online repository for public domain literature. The data comes in the form of a JSON file which has the following fields: ”s”, a line of poetry, and ”gid” an id of the book in the gutemberg project database. To clean the data we stripped every line of the poem, and segmented them into ”poems” of 10 lines. We decided to split the data every 10 lines since the main reason for using this data set is to draw from more historical and poetic languge. Therefore, the actual length of the text we trained our model on is not as relevant since we work out the structure of the haiku through the training.

Another data source we used is the CMU pronouncing dictionary (University, 2014). We use this dictionary to count the amount of syllables in each line of the haiku. Given that this is an open source project with a python library, data cleanup was minimal for this dataset.

After downloading the data, the cleaning process for the haiku dataset consists is run by a python script. This python script takes in each line of the JSON file, and removes uncommon characters such as the return-carriage character (`\r`), tabs (`\t`), and em-dashes (—). Next it converts all the characters to lowercase. And finally, it appends the haiku to a text file in which the end of a poem section is denoted by two new line characters. The text file is then used at training when loading the data. Below is an example of the raw input data, and its processed form.

Original (Raw text):

{ ”s”: ”That was our bench the time you said to me”, ”gid”: ”442” }
{ ”s”: ”The long new poem–but how different now,”, ”gid”: ”442” }
{ ”s”: ”How eerie with the curtain of the fog”, ”gid”: ”442” }
{ ”s”: ”Making it strange to all the friendly trees!”, ”gid”: ”442” }
{ ”s”: ”There is no wind, and yet great curving scrolls”, ”gid”: ”442” }
{ ”s”: ”Carve themselves, ever changing, in the mist.”, ”gid”: ”442” }
{ ”s”: ”Walk on a little, let me stand here watching”, ”gid”: ”442” }
{ ”s”: ”To see you, too, grown strange to me and far. . . .”, ”gid”: ”442” }
{ ”s”: ”I used to wonder how the park would be”, ”gid”: ”442” }
{ ”s”: ”If one night we could have it all alone–”, ”gid”: ”442” }

Finalized (After cleaning):

That was our bench the time you said to me
The long new poem but how different now
How eerie with the curtain of the fog
Making it strange to all the friendly trees
There is no wind, and yet great curving scrolls
Carve themselves, ever changing, in the mist
Walk on a little, let me stand here watching
To see you, too, grown strange to me and far
I used to wonder how the park would be
If one night we could have it all alone

Another use of the data is in building the vocabulary before training. To do this we run the entirety of the text file word-by-word into a python dictionary that maps each word to an index. This lets us represent the words encountered in a numeric form that is easier to deal with. Finally, we must say that because the nature of this project is generative, testing new data simply means prompting the model differently. In adition to being the largest collection of poems we could find, this data set provides the additional benefit of having a wide range of styles. This allows us to make more diverse poems as seen below in the qualitative results section.

6.2 Baseline Model

Our first main baseline model is a markov chain of depth four. This was trained on the same data used in the primary model. The main structure is shown in figure 0:

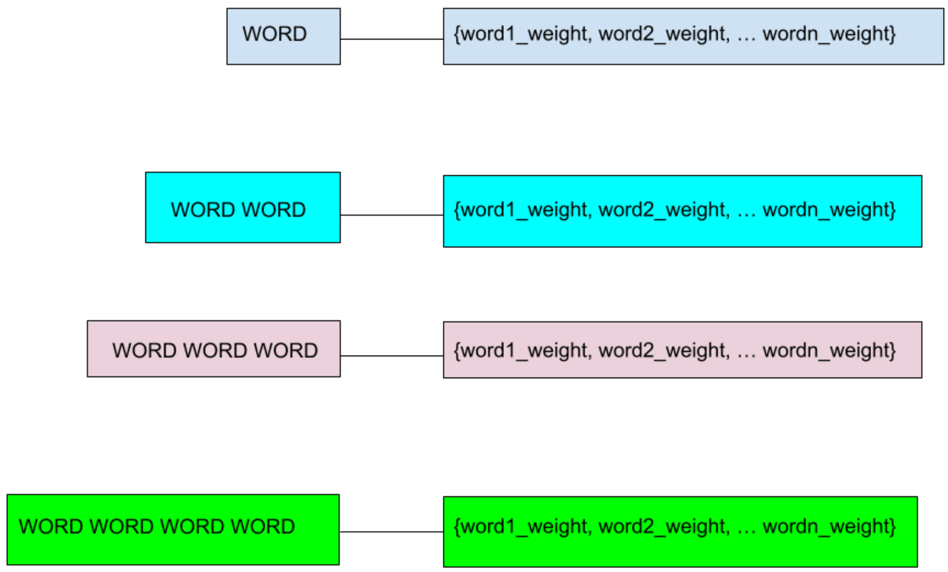

Figure 2: general weighting scheme for markov chain

Each individual word in the corpora is assigned a weighting scheme for every word (including itself) based on how frequently those other words come after the first word.
Then every pair of words is assigned a weighting scheme for every word in the corpora based on how likely the word appears after the pair.
Then every triple of words is assigned a similar weighting scheme.
This repeats until you have a weighting scheme for every n-gram sequence of words up to the depth provided by the function.

Throughout this process we treated newline characters and ”end of poem” lines (EOP) as distinct words. The model would take in a starting string of words and would try to autocomplete the poem. It chose words by taking a weighted random choiced based on the weights calculated for the markov datastructure

We chose to use a depth of 4 because lower depths had far more incoherent outputs while higher depths seemed to either return no output or just directly copy poems for the most part. Here are some example inputs, and their corresponding outputs:

Show me:

show me a garden that's
bursting into life

Give:

give your
puppies a little extra
thankful that god took me out
this funny as hell
steve's hair evolving

Teach:

teach people
how to get folder icons
to show in snowboard

It should be noted that these poems were hand-chosen from the markov chain for being more coherent. Often, the markov chain will simply copy off another poem, return no output, or only return a couple of words, such as the input "teach" generating the output "teach great place to live".

Even in the best case, this model struggles to correctly write lines with the correct number of syllables.

We created another baseline model from a very simple LSTM RNN. the structure for this one was a much simpler version of our primary model.

It had a 128 dimensional embedding scheme with 2 hidden layers of dimension 256 and a learning rate of 0.001. It was trained with a batch size of 64 for 30 epochs. The output was mostly incoherent so we decided to use the Markov chain as a baseline. All the source code for the first RNN and the Markov chain is on github, under the jupyter notebook `RNN_Model_1`.

6.3 Primary Model

The overall model architecture is described in the Figure 3 flowchart.

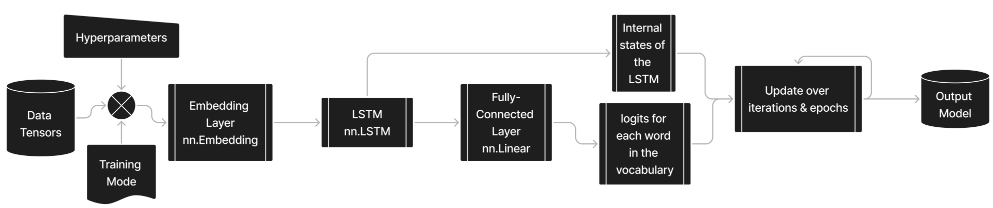

Figure 3: Model Architecture

For our haiku generation model, we continued with a Long Short-Term Memory (LSTM) architecture to capture the contextual nuances, and move towards generating coherent and accurate haikus.

The architecture of the model is as follows:

First, the input data is passed through an embedding layer. It transforms this data into vector representations that capture the relaitonships between words throughout the data. The layer takes in the number of unique words in the dataset, including our special tokens. These special tokens consist of the following:

- Padding token - used to make sequences uniform in length within a batch.
- Unknown token - represents a word that is not in the vocabulary.
- End-of-Sequence token - signifies where the model should stop generating text.

The embedding dimsensions are also inputs to this layer, which are the size of each word in the embedding vector, as is the padding index - the index reserved for our padding tokens that ensure padded positions don't contribute to the learning process.

Next, the output of the embedding layer is passed through the LSTM layer. This layer serves to process the sequence of embeddings to capture contextual information across words. It takes in the numbers of features in the hidden stats of the LSTM, the number of stacked LSTM layers, and the batch size. The resuling output of this layer is a tensor that has the output features from the LSTM, as well as the hidden states of the LSTM for each layer.

Finally, the output of the LSTM layer is passed through a fully-connected layer. This maps the LSTM outputs to our vocabulary set, and produces logits for each word in this vocabulary. The input feautres are equal to the LSTM's hidden dimensions, and the output features are equal to the size of the vocabulary.

In our current configuration, we used 128 embedding dimensions, 256 hidden dimensions, and 2 LSTM layers.

With our current training data, we have a vocabulary size of 50554 words.
The embedding layer has $50554 \cdot 128 = 6470912$ parameters.
Next the the LSTM layer has $2 \cdot 4 \cdot (256 \cdot (128 + 256) + 256) = 788480$ parameters. The four is to account for the input gate, forget gate, cell gate, and output gate in an LSTM.
Finally, the fully-connected layer has $256 \cdot 50554 + 50554 = 12992378$ parameters.

Therefore, the total number of parameters in the current model is 20251770.

As for our training hyperparameters, we chose to train over 50 epochs, with a learning rate of 0.001, a batch size of 64, an Adam optimizer[1], and a Cross Entropy Loss criterion[2].

Quantitative Results:
We tested out model with 300 one or two word prompts to the model, and counted the syllables in the output haiku.

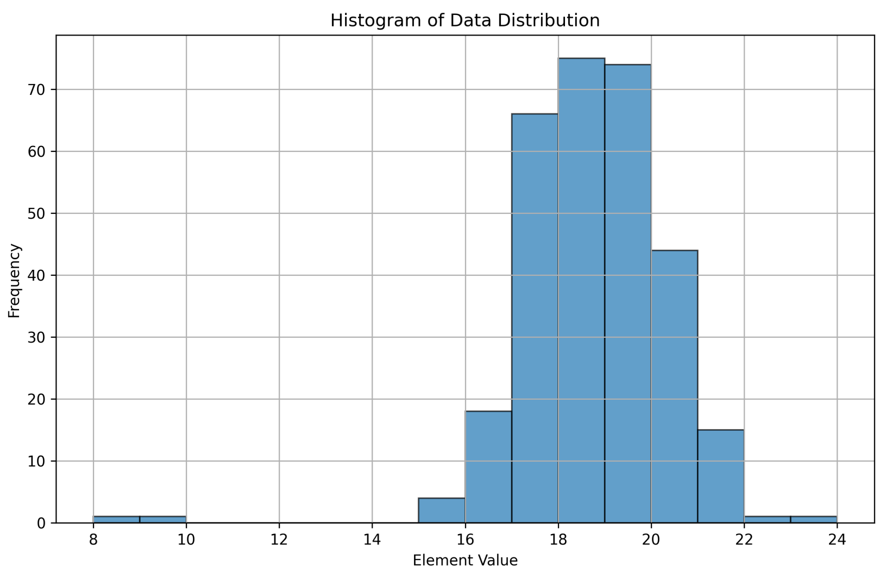

Figure 4: Syllables per haiku from 300 outputs

[1]Chosen from findings in (Kingma & Ba, 2017)
[2]Chosen from approach in (TensorFlow Team, 2023)

As we can see in Figure 4, the vast majority of generated haikus are between 17 and 19 sylalbles, with a few outliers. These can be explained by either the model incorrectly counting syllables from words that are not in the CMU Pronouncing Dictionary (University, 2014), or the syllable counting algorithm miscalculating the output. We are aiming to have our model only output 17 syllable haikus where possible - there are more methods we will be integrating into the model.

Qualitative Results:
The model generates a mix of interesting or funny hakius, and incoherent outputs:

- *”river as thou fledst*
  she glows within thy dreams where
  babe sad favorite”
- *”water from the breeze*
  there love dejected lies not
  here and could he dream”
- *”grass and the moon is*
  the smoke of a broken tree
  is great for the sun”
- *”ocean says she has*
  to stir stainless nightingale
  but well they judge me”
- *“car wolves reach of sighs*
  and coat an hurrah like strife
  truest mother round”
- *“long does fairer by*
  their heart yet cracks and half so
  late they bent him off”
- *“when the death is no*
  knight ere for my dower and the
  vine and hints of that”
- *”wake when spite of love*
  when that their light are red and
  many a star sweet”

As we can see, the model is able to generate haikus that could pass for human-written, however, it still generates incoherent haikus. While the decision to make train the haikus in poetry rather than haikus resulted in more poetic language, it also increased the frequency of rarely used words.

The main challenges we faced making this model were related to ensuring syllable structure and was maintained and the syllable counts were accurate, and slow training times.

Because of irregularities in the English Language, we found it difficult to come up with a reliable method to count the number of syllables in a given word. Our current solution involves referencing the CMU Pronouncing Dictionary (University, 2014) to get the syllable count of a given word. However, this method is not perfect, as the CMU Pronouncing Dictionary does not contain all words that may show up in our vocabulary. This causes issues when a word is not in the dictionary, so we opted for heuristic algorithms to fix this, although this is still not perfect in practice, it significantly reduced the number of times the 5-7-5 meter was broken.

Despite gaining more resources for training through Vertex AI, this is not a sustainable way of training the model as it quickly runs out of free compute credits. However, using even just the free Tier drastically increased the speed at which we were able to train the model.

7 Ethical Considerations

Despite AI being around for several decades, it was only recently that computational power necessary to generate human-level works of art became widely available. As impressive as they are, Increasingly improving models that generate images, written stories, and poems also raise questions about how these models get and utilize their training data. While this is a very trending topic, legislation protecting individuals from having their written works used to train AI models is still in early stages. Because of this we have decided to only use works in the public domain. The works we use in this model can all be found in the Gutemberg Project digital library meaning none of them are held under exclusive rights. Finally, this model is a purely academic experiment, and we do not seek to benefit from the material it generates. Therefore, the literary works we will deal with in this project and our use of them fall under fair dealings as per Canadian law (Branch, 1985) .

We have discussed the steps that we will take to prevent misuse of our training data, now let's dig deeper into how someone with harmful intentions might use our project. The biggest cause of concern comes from poem-writing competitions. Instead of having to naturally come up with poems, someone might take advantage of a deep learning model to easily generate several entries, increasing their chances of winning or simply overloading the contest's submission portal. While many competitions may not explicitly ban AI-generated poems, it is not hard to see how a generated poem winning over many human authors may be seen as controversial. Our system somewhat counteracts this by having the user be vital in the poem creation as they must prompt the model in order for it to generate anything. Additionally, our limited scope of haikus would make it harder for someone to take advantage of the model on a massive scale as haikus are notoriously short. With these ethical considerations, we can confidently say that misuse of this model his highly unlikey.

REFERENCES

biglam. gutenberg-poetry-corpus, 2022. URL https://huggingface.co/datasets/biglam/gutenberg-poetry-corpus.

Legislative Services Branch. Consolidated federal laws of canada, copyright act, 1985. URL https://laws-lois.justice.gc.ca/eng/acts/c-42/page-6.html#docCont.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

TensorFlow Team. Text generation with an rnn, 2023. URL https://www.tensorflow.org/text/tutorials/text_generation.

Carnegie Mellon University. The carnegie mellon university pronouncing dictionary, 2014. URL http://www.speech.cs.cmu.edu/cgi-bin/cmudict.