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: 7

1 PROJECT DOCUMENT SUBMISSION FOR APS360 COURSE

The format for the submissions is a variant of the ICLR 2022 format. Please read carefully the instructions below, and follow them faithfully. There is a **9 page** limit for the main text. References do not have any limitation. This is also ICLR's standard length for a paper submission. If your main text goes to page 10, a -20% penalty would be applied. If your main text goes to page 11, you will not receive any grade for your submission.

1.1 STYLE

Papers to be submitted to APS360 must be prepared according to the instructions presented here.

Authors are required to use the APS360 LaTeX style files obtainable at the APS360 website on Quercus. Tweaking the style is not permitted.

1.2 RETRIEVAL OF STYLE FILES

The file `APS360_Project.pdf` contains these instructions and illustrates the various formatting requirements your APS360 paper must satisfy. Submissions must be made using LaTeX and the style files `iclr2022_conference.sty` and `iclr2022_conference.bst` (to be used with LaTeX2e). The file `APS360_Project.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in sections 2, 3, and 4 below.

2 GENERAL FORMATTING INSTRUCTIONS

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing of 11 points. Times New Roman is the preferred typeface throughout. Paragraphs are separated by 1/2 line space, with no indentation.

Paper title is 17 point, in small caps and left-aligned. All pages should start at 1 inch (6 picas) from the top of the page.

Authors' names are set in boldface, and each name is placed above its corresponding address. The lead author's name is to be listed first, and the co-authors' names are set to follow. Authors sharing the same address can be on the same line.

Please pay special attention to the instructions in section 4 regarding figures, tables, acknowledgments, and references.

There will be a strict upper limit of 9 pages for the main text of the initial submission, with unlimited additional pages for citations.

3 Headings: first level

First level headings are in small caps, flush left and in point size 12. One line space before the first level heading and 1/2 line space after the first level heading.

3.1 Headings: second level

Second level headings are in small caps, flush left and in point size 10. One line space before the second level heading and 1/2 line space after the second level heading.

3.1.1 Headings: third level

Third level headings are in small caps, flush left and in point size 10. One line space before the third level heading and 1/2 line space after the third level heading.

4 Notable Contribution

These instructions apply to everyone, regardless of the formatter being used.

4.1 Data Processing

The main data used for this project consists of the haiku4u dataset in kaggle. This dataset is publicly available and comes from a web scraper run in October 2023. The data comes in the form of a csv file which has the following fields: Haiku, a field containing the entire haiku with lines separated by a "|" character; Domain, the name of the web page where it was originally found; Domains, the amount of times this specific haiku was encountered while scraping the web; URL, the specific url where the poem was found; URLs, the number of urls that were found in in the same domain. Siblings, the number of other Poems that were found in that same URL. Another data source we used is the CMU pronouncing dictionary. 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 row of the csv file, and removes all the rows that do not contain a haiku. Then, it splits the haiku into its three lines based on where the | delimiter can be found. Next it converts all the characters to lowercase. And finally, it appends the haiku to a text file in which the end of a haiku 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):

Yesterday it worked.|Today it is not working.|Windows is like that.,richardneill.org,27, https://richardneill.org/humour.php,27,19

Finalized (After cleaning):

yesterday it worked.

today it is not working.

windows is like that.

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, we testing new data simply means prompting the model differently. However, we are actively looking into ways of increasing the training data we have access to. Some examples of how we could do this are concatenating this data set with others like it and scraping the web ourselves in the hopes we find many more haikus that were not seen originally in 2023.

4.2 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:

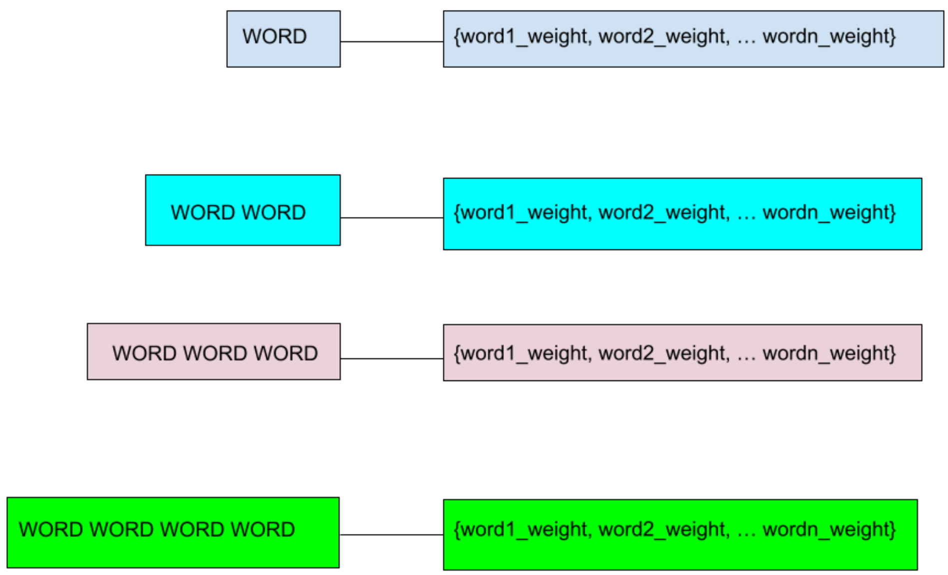

Figure 1: general weighting scheme for markov chain

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 $notebookRNN_Model_1$.

4.3 Primary Model

The overall model architecture is described in Figure 2 flowchart.

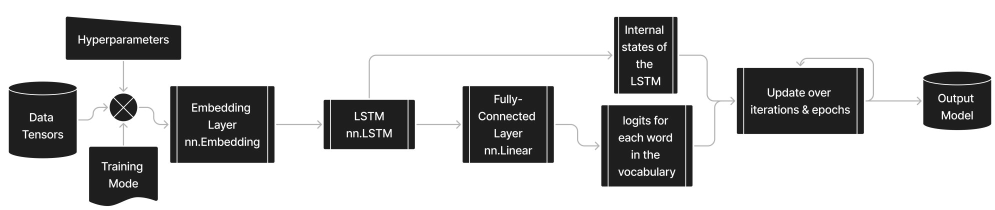

Figure 2: Model Architecture

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.

As we can see in Figure 3, 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 miscounting on the output. We are aiming to have our model only output 17 syllable haikus, so we have 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 flows wildly*
air patches on the surface
dissipates in life"
- *"rainfall through the roof*
dancers from serenity
as his life goes by"
- *"grass on the wayside*
looking for the light to come
warm clouds and weather"

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

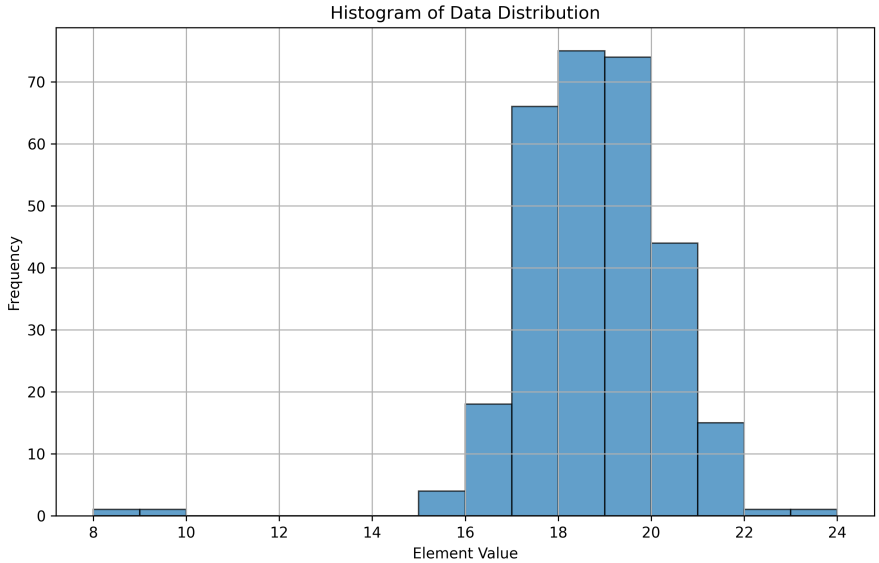

Figure 3: Syllables per haiku from 300 outputs

- *"bloom gives way to me
for i am not your hero
i am george carlin"*
- *"pond ripples to steal
winters for having their gaze
beg for hope bears now"*
- *"branch and wood filters
of night bright and sunny day
hot lemonade waits"*
- *"stream sits and pure fish
in between and i am glad
with what we i"*
- *"ancient tree freedom
anxiety creation
ignites and section"*

As we can see, the model is able to generate haikus that could pass for human-written, however, it still generates incoherent haikus. Moving towards generating more coherent outputs will be a focus of our as we improve to model.

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 are looking to integrate algorithms that can help resolve this, although this still only provides an approximation which can lead to slight issues during generation.

The slow training times were due to the large amount of data and limited access to compute resources. We tested differing batch sizes to reach a reasonable training time given our resources, however we are looking into ways to gain access to more powerful GPUs to speed up training.

5 DEFAULT NOTATION

In an attempt to encourage standardized notation, we have included the notation file from the textbook, *Deep Learning* Goodfellow et al. (2016) available at `https://github.com/goodfeli/dlbook_notation/`. Use of this style is not required and can be disabled by commenting out `math_commands.tex`.

6 FINAL INSTRUCTIONS

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the REFERENCES section; see below). Please note that pages should be numbered.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

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`.