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 project proposal for our APS360 final project. We propose building a Recurrent Neural Network that generates Haikus. We will train our model on haiku data from Kaggle, and we will base the architecture off pre-existing deep learning models that have proven successful in generating poetry in the past
—-Total Pages: 7

1 INTRODUCTION

"Poetry heals the wounds inflicted by reason" —Novalis

In this project we propose to build a deep learning model to generate short poems of various topics. 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.

This is a similar technical problem to the one addressed by large language models, though it is considerably smaller-scale given our limited time and resources. Rather than generating arbitrary text, we propose to train on and generate only poetry. Focusing on poetry offers technical advantages beyond limiting the scope of the project: the rules of grammar and punctuation are generally considered more flexible in poetry than in prose writing, which may reduce the required complexity of the mode and allow for a wider range of acceptable output.

Besides our interest in the philosophical implications of a machine engaging in what many consider a characteristically human activity, there will also be practical applications for our proposed model. Generating poetry of different styles may yield new insights on the defining characteristics of each style. Additionally, the successes and failures of a neural network trying to generate poetry may uncover novel insights into the structure of different poetic forms and what constitutes 'good' poetry for different genres. As such, our project will help enable human writers to be better poets, while raising interesting questions on the nature of poetic expression and its connections to emotion and consciousness.

Our code can be found at this GitHub repository: https://github.com/Evan-Banerjee/aps360

2 ILLUSTRATION

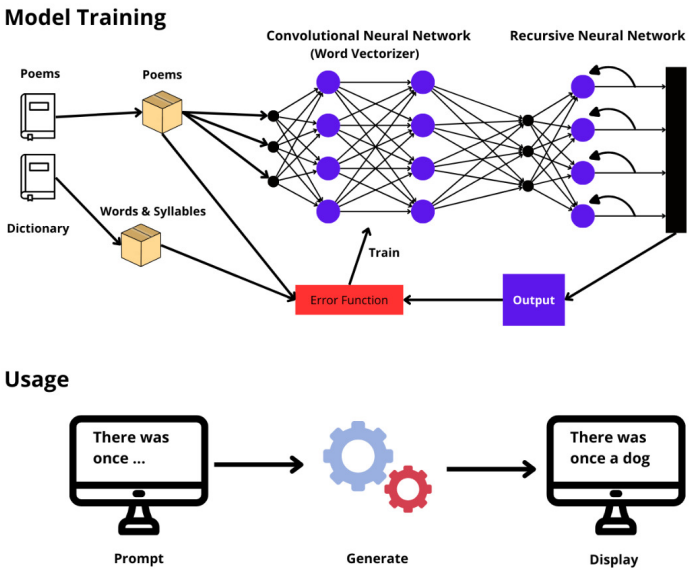

Figure 1: Figure 1: Proposed Model (Training/Deployment)

3 BACKGROUND & RELATED WORK

Previous poetry generation models utilized Recurrent Neural Networks (RNN) with some form of memory limiting to generate poems off a large dataset.

One model, created by researchers at the University of Saarland, utilized encoder decoder systems, as well as long short-term memory (LSTM) and Gated Recurrent Units (GRU) to address vanishing gradients, (Novikova et al., 2022).

Another model, developed at the University of Edinburgh utilized multiple neural networks chained together, with a convolutional neural network (CNN) for encoding, one RNN for creating hidden context parameters, and another RNN with a separate weight vector for modifying the final output (Zhang & Lapata, 2014). The output of this model was particularly compelling, with human ratings for the generated poems being over 75% of the ratings given to human written poems.

Another model, built at City, University of London, used syllables as the inputs and outputs of their RNN instead of words (Lewis et al., 2021). This model's poems were incredibly convincing, with 56% of polled individuals believing the AI generated poems were written by humans.

When focusing specifically on haikus, even a relatively basic model built by students at Stanford University was able to score well (Aguiar & Liao, 2017). For this model human evaluators gave the AI generated haikus an average quality score that was over 80% of the average score given to human poems. This model only used a single RNN with LSTM. Granted, the evaluations were only done by 8 individuals, but the strength of RNNs for this kind of task is evident.

Finally, a model presented at the 13th conference on language resources and evaluation found a large degree of success in constraining the output of neural networks to generate poems (Popescu-Belis et al., 2022). This model forced the AI to repeatedly output lines of a poem until certain syntax requirements were met. It also enabled the user to specify emotional tone by encoding category preferences (happy, sad, etc.) into a weighted vector that was then used after generating a line to swap out words that didn't fit the categories well with words that did.

This was most effectively done when words were chosen based on word frequency from other poems that fit the specified category. This model also utilized a softmax function that truncates the least likely words from the sample space during generation, increasing the probability of likely words and drastically improving output.

Our model will utilize many of these techniques, though the exact extent is still to be determined.

4 DATA PROCESSING

The data we have gathered consists of two datasets collected from Kaggle. One contains over 115,000 haikus in English and their meter (Jhalani, 2020). The second one is composed of approximately 130.000 words in the English dictionary (Schwartz, 2022) and will serve as a starting point for the language model to gather words from it. From the haiku database, we will only take the ones with a 5-7-5 structure so that we have a more rigid definition of a haiku our model can hopefully learn. In terms of data processing, we will parse the CSV files and extract the poem's text and meter. By the end, we will be left with only the haikus who follow the 5-7-5 structure and a list of all the words in the English language along with their syllables.

The data processing of the poems will then follow these steps:

1. Remove the unused columns of data.
    a) Remove the columns that are not relevant for our purposes. The CSV file for haikus will only have the poem and meter field.
    b) In the dictionary dataset, this means removing all the columns except for "word" and "syllable count". This dataset is now processed.
2. Filter the poems.
    a) Remove the rows that contain poems whose tags aren't related to one of the "genres" we will work on. The related tags are to be manually selected. For the haikus, this removes all the haikus that do not have the 5-7-5 structure.

5 ARCHITECTURE

For this project, we have chosen the Long Short-Term Memory (LSTM) model, which is based on a Recurrent Neural Network (RNN).

An LSTM model is a modified version of an RNN that is especially good when it comes to remembering past data that was entered. One of the issues present in many RNN models is called the vanishing gradient, which occurs when the gradients become extremely small as they are back-propagated during training. This is can occur due to the activation functions having derivatives less than 1 (or equal to zero in the case of dead neurons using ReLu activations), but it can also happen simply because the weights used as coefficients in many of the derivative terms in backpropagation are relatively small, so the number of distinct terms in the sum for any one gradient (equal to the number of distinct paths from one node to the output) ends up being exponentially smaller than the size of each individual term. This is unavoidable if the weights are kept small enough to prevent the opposite problem, where gradients explode due to each individual term not shrinking fast enough to account for the massive increase in terms. This is especially problematic in RNNs because each new term processed during training essentially creates a new layer, so as time goes on, each chain of partial derivatives decreases exponentially faster than the number of chains increases, and the gradient tends to zero. LSTM will address this issue by changing the RNN so it doesn't create new layers. Instead, a new vector called the cell state will be made, likely with the same dimensionality as the hidden state vector, and it will be updated at each time step using forget and input "gates". These gates will be separate neural network layers (likely with low depth) that use the new inputs and previous hidden states to increase or decrease certain values in the cell state

to represent "forgetting" or "learning" new information. Then an "output gate", which is another neural network (of relatively low depth) takes in the modified cell state to calculate the new hidden state, which can be used with the next input to begin the modification process all over again. These gates have to be trained with backpropagation, and they are limited in the quantity of information they can encode, but they fix the error of vanishing gradients because now the number of layers is constant during training since the "memory" of the training is encoded in a finite vector.

Beyond using LSTM, our model will incorporate a vectorization algorithm that will originally be a pre-trained Convolutional Neural Network (CNN), but that we may replace with an untrained one that we include in the backpropagation training. The output of the RNN will be a logit, which we will transform with a softmax function to create a probability distribution to choose the next word in the haiku. Newline characters will be part of the corpus of words so that the outputted poems span multiple lines. The softmax function will incorporate truncation into our softmax model to improve word selection. The network will be forced to output lines multiple times if it fails to meet syntax requirements, and we will algorithmically choose the output that matches our format most closely. Finally, if time permits, we will incorporate category labels into our training dataset and utilize word frequency lists between categories to dynamically swap out words post-line generation to push the topics in the direction the user specifies. If possible, we'll also incorporate this word frequency weighting into the final softmax function to prioritize words that more closely match the theme.

6 Baseline Model

We will be comparing our model against a Markov chain based text generation algorithm. The specific instance we will derive this from is (Aguiar & Liao, 2017).

Markov-chain based algorithms for text generation were one of the more common methods before neural network based models became a viable option. By starting with a body of text, the algorithm creates a probability map of which words are most likely to follow one another, and through a starting word or phrase, the algorithm recursively finds the most likely word to appear next in a sentence based on the previous word until a desired length is achieved.

These algorithms are usually simple and highly performant, however, because they are only able to capture local patterns, Markov based algorithms are limited in scope and have a lack of coherence over longer passages. Comparing our neural network model against a Markov chain algorithm will demonstrate the difference in sophistication we are able to achieve using a more modern approach, and how it changes over various generated passages.

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. After all, if a human can learn to imitate another human, what stops a machine from imitating the copyrighted work of an artist?

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 which is why we are taking additional caution. We will take several steps to ensure that we are not infringing upon an author's intellectual property. First, we will be taking our poems from datasets found in Kaggle. This way we can use their terms of services as a moral guideline for how to utilize these datasets (**?**). This means we will not redistribute them as that would constitute plagiarism. Additionally, the poems we will be using list their source of origin. With this, we can be certain that the poems look at come from a publicly accessible source and not copyright-protected material. 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, 2024).

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.

8 Project Plan

Task	Primary team member responsible	Deadline	Status	Deliverable
Draft introduction and project plan	Ji	2024/10/01	Ongoing	Project Proposal
Draft data processing and ethics sections	Diego	2024/10/01	Ongoing	Project Proposal
Draft background and related work sections	Evan	2024/10/01	Complete	Project Proposal
Draft risk register and baseline model sections	Noah	2024/10/01	Complete	Project Proposal
Architecture section and model illustration	Evan		Not started	Model Construction
Research and gather online dataset sources	Diego	2024/10/10	Not started	Model Construction
Consolidate dataset for our project	Ji	2024/10/20	Not started	Progress Report
Working prototype network architecture	Unassigned	2024/10/20	Not started	Progress Report
Overseeing training	Evan	2024/10/26	Not started	Progress Report
Validation and troubleshooting	Noah	2024/10/29	Not started	Progress Report
Finalize model architecture	Unassigned	2024/11/05	Not started	Model Construction
Final model training	Noah	2024/11/17	Not started	Model Construction
Final testing and quantification	Ji	2024/11/20	Not started	Model Construction

Figure 2: Figure 2: Project Plan

8.1 Communication

Team communication will take place via WhatsApp group, which all team members will check daily. We will also hold weekly in-person meetings to discuss task progress after our shared lab session, and asynchronous Zoom meetings as needed for specific tasks and deliverables that require full-team input.

8.2 Team Norms

we plan to work in a GitHub repository, which will facilitate version control and ensure no work is lost during editing.

Note: the project plan remains somewhat open, to allow for pivoting as we learn more about the requirements of the project. Also note that the individual task assignments are not absolute, and all team members will collaborate on difficult tasks as needed.

9 Risk Register

9.1 Model does not generate coherent output

This is possible due to the complexity of generating text, and the potentially large variations in the training data. If our created model is not generating poems that make sense, does not output enough text to be acceptable, or takes too long to generate text, we would first scale back the model to only focus on creating one sentence at a time, and increase the amount of training data it has to work with. Once we have refined our model and see consistent results, we would then increase its size and continuously evaluate its performance as it scales.

9.2 Team member drops the course

If a team member decides to drop the course, a meeting will take place where the departing member will go over in detail all of their assigned sections such as: what has been completed, what is in progress, what gave them challenges, etc. Once the team has figured out which parts of the project are affected, the remaining members will meet and see which parts align best with their currently assigned portions and/or expertise. In the event the departing member had a portion that is unrealistic to finish in time, the scope of the project will be re-evaluated.

9.3 Model is taking too many resources to train

If the model requires much more time or compute resources to properly train than anticipated, and it is not an issue of improperly constructed code, the team will analyze the individual components of the model to figure out which ones are requiring the largest amount of resources. For example, if the training sets are too large or take too long on our computers, we will first see if we have access to a more powerful computer the training can be performed on. If this is unrealistic or inaccessible, the scope will be re-evaluated in order to have it properly fit within our given resources, which could be reducing complexity of the model or using smaller datasets.

9.4 Original Scope is too large

If the realization comes up that we were too ambitious in our original goal, and constraints such as available working time or complexity become large enough issues where in the proposed design is unlikely to be met, each member will share their concerns which could be regarding other commitments such as classes, lack of understanding when it comes to their assigned section, or any other concern that impacts the progress of the project. We will then make any necessary changes to the project design and scope to ensure a working model is delivered.

9.5 Internal Disagreement about project direction

If one or more team members brings up any issues they see with the project that could impact progress, a team meeting will take place at which the concerns are communicated to the other members. Each member will have the opportunity to share their opinions on the matter, after which we will deliberate over potential solutions until the entire team feels satisfied with the choices made going forward.

REFERENCES

Rui Aguiar and Kevin Liao. Haiku generation. GitHub, 2017. URL https://github.com/raguiar2/haiku_generation/tree/master.

Legislative Services Branch. Consolidated federal laws of canada, copyright act. Copyright Act, Oct 2024. URL https://laws-lois.justice.gc.ca/eng/acts/c-42/page-6.html#docCont.

Harshit Jhalani. Haiku dataset. Kaggle, Aug 2020. URL https://www.kaggle.com/datasets/hjhalani30/haiku-dataset.

Danielle Lewis, Andrea Zugarini, and Eduardo Alonso. Syllable neural language models for english poem generation. In *12th International Conference on Computational Creativity (ICCC'21)*, pp. 350–356. Association for Computational Creativity, 2021.

Svetlana Novikova, Sangeet Sagar, Peilu Lin, Meng Li, and Pavle Markovic. English and chinese poetry generation software project: Deep learning for the processing and interpretation of literary texts. 2022.

Andrei Popescu-Belis, Alex R Atrio, Valentin Minder, Aris Xanthos, Gabriel Luthier, Simon Mattei, and Antonio Rodriguez. Constrained language models for interactive poem generation. In *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*. 20-25 June 2022, 2022.

Jon Schwartz. English phonetic and syllable count dictionary. Kaggle, Aug 2022. URL https://www.kaggle.com/datasets/schwartstack/english-phonetic-and-syllable-count-dictionary.

Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 670–680, 2014.