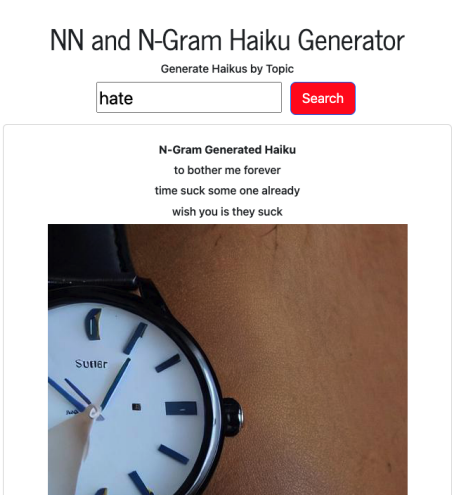
Glen Damian Lim, Ben Tunney
CS4120
Felix Muzny
04/19/2023

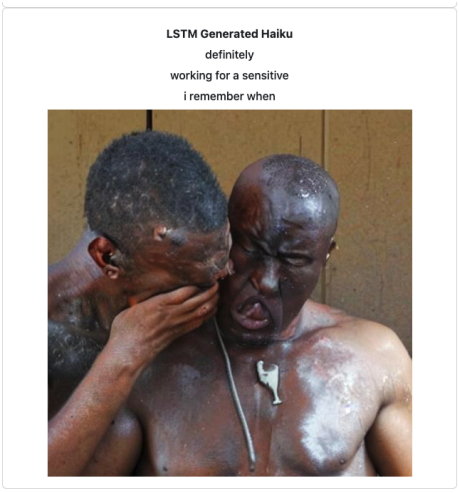
Generating and Visualizing Haikus With NLP

What did you do and why is it interesting?

We created a Plotly dashboard in Python that allows a user to query to generate and visualize haikus based on a requested topic, using an N-Gram language model and an LSTM language model. These haikus were then visualized with AI text-to-image generator API calls. These models were trained on unlabelled haikus. Haikus are poems that are 3 lines following the syllable structure of 5-7-5. It was an interesting challenge to attempt this task using NLP techniques because of the artistic nature and the syllable/line structure.

The Plotly dashboard can be run locally via the "nlp_haiku_generator" Jupyter Notebook. Users can input a search term and a haiku will be generated from both language models. For each haiku there is an output AI text-to-image photograph generated. This is an implementation of Hugging Face's Stable Diffusion API from Runway ML (1). This model was trained on 512x512 images and uses a text encoder to condition the model to text prompts. For the text prompts to query the API, we implemented NER from the spacy package (2). If there existed named entities within a given haiku, then a string of these NERs would make up the text prompt. In the case that there were no NERs, we used NLTK to remove stopwords from the haiku and the resulting text was used as the image generator text prompt. This dashboard can be run many times for many different queries. Below are example outputs of the Dashboard.





The main challenge and goal of this project was to be able to create meaningful, interesting, and artistic haikus. When written by humans, there is a lot of time and effort that goes into crafting these haikus. Haikus often follow nontraditional language structures, disregarding common grammatical rules in order to follow syllable structure and optimize artistic

effect. Still, these haikus are understandable and effective, but using haiku data means that we will frequently train on text data that might result in linguistically confusing haikus. It was an interesting task to approach multiple language models to try to maintain readability, but also capture some sense of beauty and artistry in the haikus.

The other challenge and goal was to maintain the structure of these haikus. While generating text, the original language models produced words that might go against the syllable count of that line. From this challenge, we had the task of editing our language models to generate text while simultaneously counting syllables. We also had the task of considering how the context of lines preceding lines will affect generation of following lines (haikus make continuous contextual sense over all 3 lines).

How do the models work?(Point out any important details regarding your approaches to unknown words, training, decoding, etc.)- will remove this

For our first model we used a modified version of our N-Gram language model (bigrams). First, we trained a Word2Vec word embedding. When a user queries for a specific haiku topic, we acquire the top 5 most similar words to this query using the word embedding. The data we used was 144,123 unlabeled haikus. We used the top 5 similar query words to scan all of the haikus in the haiku dataset and use the subset of the dataset with haikus that contained any of the 5 words. Using this method, the N-Gram model was only trained on data that was similar-to or related-to the topic requested. We did not have to perform much preprocessing of the text; the dataset had no punctuation or no frequent abnormalities. We added begin and end tokens to the endpoints of each haiku to help with text generation. To acquire tokens we simply split the data on all of the spaces. All tokens with one occurrence were saved as <UNK> tokens.

We used this data to train the model, acquiring the counts of all of the bigrams and individual tokens, and continuously generating bigrams using probabilities from these counts. As we generated these tokens, we used the "syllables" library from PyPi, to keep track of the syllable counts of each line. Once a line had met or exceeded the syllable count, we moved on to generate the tokens for the next line. These rough attempts often produced haikus that did not follow the 5-7-5 structure, so we continuously iterated until the haiku generated followed this structure. For the Dashboard implementation in order to produce a readable and interesting haiku, we continuously iterated to generate haikus until no <UNK> values were in the output.

For our second model we used a long short-term memory (LSTM) network. This is a version of a recurrent neural network that takes sequences of data into context and can learn long-term dependencies. For this model, we used the same trained word embeddings that we used for our previous model in order to increase computation efficiency. We pre-processed our data into fixed sequence length by using a pre-padding library from scikit-learn, then we implemented the sliding window technique for every sequence to cover every possibility. Therefore, when we are trying to generate our haikus, the model would be able to predict

correctly what is the next given word by looking at the current sequence. Before constructing the Neural Network and inputting our data into a generator, we also converted the sequence into 3D vectors (batch size, sequence length, embedding size). We used multiple LSTM layers with multiple hidden units, multiple dropout layers to prevent overfitting, then a dense layer which is our output layer that will produce a probability distribution over all of our vocabulary. When predicting the sentences, depending on the current sequence integers, we need to convert it into its corresponding word embeddings before calling the model to predict the next word. We also use the greedy algorithm approach that we used for our previous model, enforcing the syllable limit in each line.

Results and Conclusions

For our N-Gram model, we found that the haikus don't make sense in a lot of cases. Lines were often cut-off or interrupted and sometimes did not follow grammatical rules. These results were expected because it is a statistical language model without sequential learning that could produce comprehendible sentences. The haikus generated were often related to the topic and fairly artistic. While the haikus were usually unreadable, the words in the haikus usually had some artistic merit and the words were grouped artistically and interestingly.

first blossom silver	i had breathed it	the way we get back
spring rain just wanted to change	my hope in the soft whisper	is the power is my mother
blowing rain where	rain outside in	a garden buddha

Here are examples of the N-Gram model with the query: “wind.” As previously mentioned, they do not necessarily make grammatical sense in some cases, but some subsections of the text are linguistically interesting and artistic. For example “spring rain just wanted to change,” could be interpreted as related to wind with its referencing of weather, seasons, and change. In the second haiku, “the power is my mother,” creates an interesting image of power and familiar relation; nature elements like wind are often referred to familially in art. In the last haiku, “my hope in the soft whisper,” could be interpreted as a metaphorical representation of wind as a “soft whisper,” and this line also creates an interesting linguistic connection to the abstract word, “hope.” While they could be improved, these results were certainly interesting and artistic.

For our LSTM model, unfortunately we need to use 50% of our training data due to limited memory as we are using a GPU from an external source. It can be seen from our results that our LSTM network did not fully capture the underlying pattern behind the data and we suspected that this might be caused by limited training data. When we try to run it on our local machine, it produces a better accuracy and a lower loss but it is training for an extremely long period of time. Therefore, we are expecting that our LSTM model will produce a much better result when we run it using the full dataset.

Future Experiments and Future Work

One change I would like to make in the future is to look into trying different datasets. The haiku dataset was good because it contained examples of very small pieces of texts that likely had a cohesive topic throughout each individual haiku. This was good training data, because we could collect a subset of the data that was very likely related to the topic requested. Additionally, it was a good dataset because all of the haikus are poetry, and thus might have some "artistic" quality that could be captured by the language model. However, while it was a large dataset, these subsets of the full dataset were likely too small to produce extremely meaningful or grammatically-sound haikus with the N-Gram model. We would be interested in looking into short form text data of which we could acquire much larger amounts like tweets, social media post captions, or text messages.

For our final product we would be interested in creating visualizations that capture the mood of these haikus. We could use some text dataset with labeled emotions associated and train a ML model off of this dataset. Using this model we could predict the top couple of emotions associated with generated haikus. These emotions could then be used alongside an AI text-to-image generation to create emotive images and visualize these haikus.

We would also like to add capability to query the haiku generator with multiple words or a sentence. We could implement some part of speech tagging to capture values like nouns, verbs, adverbs, and adjectives. After training the models on some subset of the data from these words and related words, we could generate haikus that might result in more specific or interesting outputs.

Finally we would be interested in exploring other models that might be more specific or effective for this task. For example, Transformers that focus more on the important parts of the input sequence due to attention mechanism, might be able to generate Haikus more efficiently due to parallelization (LSTMs are sequential and have limited memory). Another thing that we are interested in exploring is building our GANs (Generative Adversarial Network) to generate AI art.

Works Cited

1. NER implementation
2. Hugging Face Text-to-Image