CS 4120
Project Proposal
Nick Bagley & Ali Moeinzad

Writing Poetry With AI

The aim of this project is to train a model that can generate poems. This model will train on datasets containing examples of poems. The trained model will then be able to generate new poems based on the learnings of the dataset. Optionally, the generated poems can be fed into another model that reads through the generated poem and replaces some of the words with synonyms to make multiple versions of a generated poem. The end user can then look at these different versions and choose the words that work best for them.

The Data

There is a dataset on Kaggle, linked here, that has hundreds of different poems separated by type of poem. This dataset is very convenient to use at is already formatted for training. However, we are worried that having many different types of poems (haikus, quatrains, etc.) will cause the model to output a jumbled combination of poem structures. Another option for a dataset is through text files downloaded from Project Gutenberg. This site has works from many poets such as Frost, Longfellow, and Thoreau. Linked here is an example of a publicly available file containing Emily Dickinson poems from Project Gutenberg. We will most likely use a combination of the Kaggle dataset with various Project Gutenberg files to create a large corpus of poems to train on.

The Tools

This project will best be done using a notebook environment for development. We will most likely be using the Keras library to train deep learning models. Based on our research it looks like recurrent neural networks, especially LSTM, are good models when working on poetry generation. We could test the readability of a deep learning model's output against other simpler models such as n-grams and Markov chains. For training these models we will try and

use word embeddings using either the Word2Vec or Glove embeddings. What our final model uses will depend on the results from our evaluations of different inputs into the model.

Another tool that we will need to use is some sort of thesaurus available for import into Python. NLTK has a library called wordnet that contains information on word synonyms. This library could be used in a poem generation pipeline by replacing various words in the generated poem with synonyms found in the wordnet. This could be done multiple times, picking random words and synonyms to replace, which would create several different versions of the final poem. The end user could then decide which of these poems sounds best to the human ear.

Results

The main goal of this project is to produce text that is readable and able to be recognized as a poem opposed to something like a short story. It will be interesting to perform some analysis on the generated poems to see the trends of the model. Looking at things such as the average length of a poem, the distribution of rhyme schemes (if it rhymes at all), and the number of syllables in a line would all give some insight into how the model was trained. Comparing these results with the same analysis for the input data would see how much the structure of the input data influences the model performance. These are all things that can be shown through visualizations in a final report to quickly see the discrepancies in the input and output data.

Conclusion

Our primary goal is to evaluate different methods of generating poems and seeing how to achieve the most readable output. We do not expect to produce any masterpiece poems, but we believe we can explain a good approach to take when trying to create a model for poem generation. We will test different models for doing this, but our main focus will be using neural networks from the Keras library in Python. We will be following the methods from homework 4 and other examples of deep learning projects online that focus on text generation. Hopefully an end user that is unfamiliar with the model will be able to read the output and identify it specifically as a poem.