Artist’s Statement.

In this day and age, technology has undergone a rapid evolution. In the span of 10 years, humans have gone from making the first smartphone to creating AI that can write code. In this piece I try to represent this progress by teaching an AI model to simulate one of the best poets in history: William Shakespeare. I trained a very minimal model with little data to do this and the results come surprisingly close to interpretable. One of the central ideas in this work was to incorporate topic modelling, and it is surprising how well Shakespeare’s poems can be classified into topics, and how effective that can make the final model in generating cohesive, themed poems.

The data I used for this project are the Sonnets of William Shakespeare. The sonnets total to 154, with 2155 sentences. The dimensions and the layers of my models are:12.

For this project, I use two models. A RNN generative model and a LDA topic model. The RNN model outputs a probability distribution for all the words and then I attempt to augment that distribution with my topic model, by boosting words that are more likely to appear in a given topic.

During Preprocessing, I split my data up into different poems separated by lines i.e Dataset = List of Poem. Poem = List of lines. I do this, because separating by poems allows for context to be preserved while training the Topic model. While training the generative model, I flatten the dataset into a list of lines. Each line is a separate training instance for the model, to make sure the input data is of consistent length, I pad my inputs according to the max length of a sentence. I use the tokenizer from keras to tokenize my data for both the RNN and LDA model. I also imported a stop words list to remove any stop words in the dataset before training the LDA. While training, I manually extended the list to remove any words that I felt weren't definitive of a topic, while this improved the performance, topics only had a vague sense of coherence. I found code online, to train a coherence model to determine the number of topics the LDA should produce, but for some reason that did not work on my station (Something about writing multithreading code in windows wasn't working, so I decided to scrap that idea rather than spend a lot of time on it), I just trained a HDP model on my dataset. The HDP model attempts to learn Beta, Alpha and the number of topics in the dataset, and I used those values as guidelines. I was also looking into using GloVe vectors to augment the final probability distribution so that words related to the topic would be more likely to be chosen while sampling (determined by the Cosine Similarity). This would then be compared to sentences generated by a trained LDA model. I got pretty close to completely implementing it, but ran out of time, so I decided to stick with LDA augmented generation instead. I trained many models with various architectures, and in my presentation I have included some of the data I collected on the many variations.

There were many avenues that I pursued that didn't end up in the final project. My main idea was to produce images. To do this, I made some prototypes for style transferring using cycle-gans, I was planning on using a sentiment analysis NLP model that would output a vector which would be used to introduce transformations in an original image supplied by the user. Deciding that that would be out of the scope of this project given the timeframe and the fact that I was planning on doing it alone, I scraped that idea. After a bit of soul searching, I realized I really liked the idea of producing a poem when prompted by an image. Completely arbitrarily I decided to generate poems trained on

Shakespearean sonnets. I would use a CNN to prompt the generative RNN, but a big chunk of my time was spent trying to get good quality poems from the RNN, so I decided to scrap the CNN idea as well. I looked into generating Word Clouds with my Topic models, but didn't think that would really add much value to my project so I decided to not go ahead with that.

For the future, I was mainly worried about the way I incorporate the two models. Right now it is a simple addition of probabilities of related words. I suspect this may not be the best way to do things, as it completely ignores the sensitivity of the distribution and how it might change by adding a value to one of its words. I would like to have the topic models only slightly nudge the 'topic words' in the final distribution output by the RNN. This nudge would possibly have a scaling factor, that would determine how aggressive the final distribution is changed. If given enough time, I would also like to add a CNN and implement the image prompt idea alongside using GloVe vectors for topic modeling.

I believe if given another day I would mainly focus on various preprocessing techniques on my dataset, as I believe I can increase the quality of the model by just improving the dataset quality. I was mainly thinking about incorporating POS tagging into my work, as I believe that would greatly benefit both my LDA and my RNN and it is something that is evidently missing in the outputs (they have very bad sentence structure). Another potential idea that could be implemented would be using a simple n-gram model to augment the RNN generative model. The n-gram model would determine the start and stop of sentences and attempt to enforce grammar on the RNN, which might speed up the training process. I read a paper where they used 3 models, one to enforce Rhyme, another to enforce meter and a third model that generated sentences. The Rhyme and Meter models scored sentences produced by the language model and thus ended up with good poetry. Another experiment I wanted to test out was, if given the same resources and training time, how would a character based RNN model fare on the same dataset as the word based RNN. I touched on this briefly while trying to understand why my current model wouldn't produce good quality sentences, and think it would be interesting to see the results. I also read of a multiplicative RNN, and I'm not sure if that would improve my results, but I'd be curious to try. A realistic next step would be trying to use my current model on other datasets of poetry, I suspect that if given a dataset of equal size with smaller form poetry such as haikus, my model would be much more effective.

Bibliography

Datasets
Beyond Narrative Description: Generating Poetry from Images by Multi-Adversarial Training
https://github.com/researchmm/img2poem#download-trained-model

Emotions dataset:
https://www.kaggle.com/praveengovi/emotions-dataset-for-nlp

Shakespeare Sonnet Dataset:
http://shakespeare.mit.edu/Poetry/sonnets.html

Papers

https://icml.cc/Conferences/2011/papers/524_icmlpaper.pdf

Automatically Generating Rhythmic Verse with Neural Networks
https://research.fb.com/wp-content/uploads/2017/06/automatically-generating-rhythmic-6-2.pdf#1d

Word2vec tutorial
https://www.tensorflow.org/tutorials/text/word2vec

Deep Learning-based Poetry Generation Given Visual Input
http://computationalcreativity.net/iccc2018/sites/default/files/papers/ICCC_2018_paper_59.pdf

Building Emotional Machines: Recognizing Image Emotions through Deep Neural Networks
https://arxiv.org/pdf/1705.07543.pdf

Deep-speare: A joint neural model of poetic language, meter and rhyme
https://www.aclweb.org/anthology/P18-1181.pdf

Beyond Narrative Description: Generating Poetry from Images by Multi-Adversarial Training
https://arxiv.org/pdf/1804.08473.pdf

Generating Poetry using Neural Networks
https://neuro.cs.ut.ee/wp-content/uploads/2018/02/poetry.pdf