CIS530-Project-MS2

Akriti Gupta, Bhumika Singhal, Namita Shukla, Smruti Chourasia

December 2022

1 Introduction

We are working on generating Haikus, the first task was to design an evaluation strategy that would evaluate the quality of the text generated. The standard metrics could not be used since they need a reference to compare, hence we are using an all round text evaluator - GRUEN. The challenging part of the project was to to ensure that the structure of the haiku generated is constrained within 5-7-5 syllable structure - so we check the mean syllable count across poems for respective lines. In this report, we will first discuss our comprehensive evaluation strategy as it is critical in any Generative model setting to evaluate the generated outputs correctly. We will then discuss the two baseline approaches we have tried and their results using our evaluation script.

2 Evaluation Script

As discussed in the previous report, we conduct some preprocessing to ensure the structure of text is correct using FastPunct module to add uppercase letters and punctuation. It helps with the text quality analysis. Our proposed metric has 2 components, namely quality and structure.

2.1 Quality Metric

Although BLEU works well for assessing text quality based on the expected result, it doesn't help determine the quality of creative writing. So we used an automated system called GRUEN to assess the quality of the prose used in the Haiku dataset.

To date, these metrics have focused almost exclusively on the content selection aspect of the system output, ignoring the linguistic quality aspect altogether. We bridge this gap by proposing GRUEN for evaluating Grammaticality, non-Redundancy, focus, structure and coherence of generated text. [1] The GRUEN score is based on four criteria for text evaluation:

- **Grammar**: A high grammatical score means the system output is readable, fluent, and grammatically correct.
- **Non-redundancy**: Non-redundancy refers to having no unnecessary repetition (e.g., the repeated use of name of a person when a pronoun would suffice).
- **Focus**: A focused output should have related semantics between neighboring sentences
- **Structure and coherence**: A well-structured and coherent output should contain well-organized sentences, where the sentence order is natural and easy to follow.

The final linguistic quality score is a linear combination of the above four scores:

GRUEN-Score = grammatically + non-redundancy + focus + structure and is on a scale of 0 to 1.

2.2 Structure Metric

The haiku's generated are 3 lines of poem where each line should have 5,7,5 syllables respectively - this is the essence of a haiku. Hence for the generated poem we check the number of syllables in each line of the poem and plot the statistics (like mean, median etc.) across line 1 of each poem, then line 2 of each poem and same with line 3. This gives us the sense of structure.

2.3 Complete Evaluation

Evaluation score = GRUEN score + Syllable structure

Finally a good model will have a combination of high GRUEN score and a syllable counter with average close to 5,7,5 for respective lines.

3 Simple Baseline (for instance, a majority class baseline)

3.1 Methodology - SpaCy Matcher

SpaCy is a NLP open-source library. It is used in designing systems for information extraction or natural language understanding systems.

1. Our approach involved learning the English words and their corresponding POS(Part-Of-Speech) tags using the Matcher module of SpaCy.
2. Then we defined the syllable(Haiku) structure i.e. 5-7-5 and the POS tag that we expect for each word in the Haiku.
3. Following this, we randomly generated words from our dataset that corresponded to the pattern specified.
4. This model is a simple guessing technique. It can be seen that there is no meaning of the Haikus generated and is made up of the most frequent words in the dataset. For example - randomly mixing sentences to generate a haiku with the required syllable structure.
5. Their quality is low, though their syllable structure is correct which is expected as we specified the pattern.

3.2 Results

We generated about 1500 poems -

1. **GRUEN score statistics** The average GRUEN score was 0.29 - with a median of 0.26 showing an almost consistent distribution. The max GRUEN score achieved was 0.79 for the poems generated. Hence the quality of the poems generated are not great.
2. **Syllable structure statistics** The average number of syllables across line 1: 4.9, line 2: 6.8 and line 3: 4.9 although the median of the distribution is line 1: 5, line 2: 7 and line 3: 5 - showing the generated data has learnt the structure of the data. This is simply because the Matcher module is able to generate based on the given POS tags and poems.

4 Strong Baseline you may have seen in literature

4.1 Methodology - Character RNN

In a gist, our character level RNN reads haikus as a series of characters - outputting a prediction and "hidden state" at each step, feeding its previous hidden state into each next step.
Step by step approach:

1. First step was data preprocessing - only the haikus (character by character) are feeded into the model without any additional metadata (such as syllable structure, topic of haiku,etc.).

2. Our neural network does not understand text so we have to convert our text data to integer. For this purpose we create a token dictionary and map character to integer and vice versa. In addition to this, we one hot encode the data to represent characters.

3. The parameters that determine the batch size are -

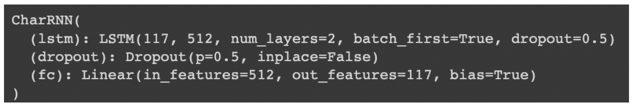
   - Batch Size, which is the number of samples we send to the model at a time.
   - Sequence Length is the length of the sequence of input data.

4. Now finally, we create a class called CharRNN. Here we use LSTM as our RNN model. Hyperparameters for the model and training are as follows:

   - Number of LSTM layers = 2
   - Number of epochs = 30.
   - Dropout = 0.5
   - Learning rate = 0.001
   - Batch size = 128
   - Sequence Length as 100
   - Gradient clipping = 5
   - Adam Optimizer
   - Cross Entropy Loss as criterion

5. After training, we define a sampler (top-k samples with k =2) which takes in a character as a seed and generates next characters.

6. Model Architecture:



```
CharRNN(
  (lstm): LSTM(117, 512, num_layers=2, batch_first=True, dropout=0.5)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=512, out_features=117, bias=True)
)
```

Figure 1: CharRNN Architecture

7. The generated output is stored in 3 columns in a csv where each column corresponds to each sentence in the Haiku. For evaluation, we use the GRUEN script.

8. The quality of generated poems is better than the simple baseline although since this learns poems character to character basis the syllable structure is on the weaker side.

4.2 Results

We generated about 1458 poems -

1. **GRUEN score statistics** The average GRUEN score was 0.34 - with a median of 0.28 showing an almost consistent distribution. The max GRUEN score achieved was 0.88 for the poems generated.

2. **Syllable structure statistics** The average number of syllables across line 1: 7.5, line 2: 8.7 and line 3: 8.0 although the median of the distribution is line 1: 5, line 2: 7 and line 3: 6 - showing the generated data has some anomalies although given the input haiku the model was able to learn some structure. Though 25%+ poems are unable to follow the structure. The 75th percentile is 10-10-10 syllable structure which states that the architecture has less than required information about the syllable structure.

5 Conclusion

We implemented a simple baseline - SpaCy matcher model and a stronger baseline - CharRNN LSTM based model for generating haikus. Considering the evaluation metrics the stronger baseline performs better in the quality of text basis. There is work needed around the generated poems having the structure required.

6 Bibliography

1. GRUEN for Evaluating Linguistic Quality of Generated Text, Wanzheng Zhu and Suma Bhat `https://aclanthology.org/2020.findings-emnlp.9.pdf`

2. Generating Poetry with PoetRNN `http://sballas8.github.io/2015/08/11/Poet-RNN.html`