The Recurrent Neural Network in the Hat: Generating a Dr. Seuss Picture Book

Michael Karr, *mkarr{at}stanford{dot}edu,* and Allison Tielking, *atielkin{at}stanford{dot}edu*

I. Task Definition

Dr. Seuss is one of the most popular children's book authors of all time, with over 222 million copies of his books sold worldwide. His books have been translated into more than 15 languages, and they continue to spark young people's creativity today. Recent trends in artificial intelligence have attempted to generate works of art, scripts, and books given a particular art or writing sytle. Given the unique structure of a Dr. Seuss book, which is full of rhymes and made-up words, we believe it will be an interesting challenge to generate a Dr. Seuss picture book, including words and potentially pictures. We will implement a basic text generator with N-grams, then advance to using a Recurrent Neural Network (RNN) to create more realistic fake books. We will also look into how to hold a rhyme structure and fake word generation, along with fake image generation if there is time.

II. Infrastructure

We are beginning our infrastructure by gathering all the text from Dr. Seuss books. We found several sources of text on Github that include *Dr. Seuss*, *Cat in the Hat*, *Green Eggs and Ham*, and *Fox in Socks*. If we need more books later on, we can hand-label by reading and typing out the books ourselves. We will make use of several Python libraries in this project. An initial web search gives us a few potential options:

- pyrhyme: a python library to find rhyming words for a provided word.
- nltk: a natural language toolkit with many tools to powerfully analyze text.
- numpy: provides highly optimized data structures, will be used alongside nltk
- A neural network library that works for recurrent neural networks. This will be determined upon further research.

III. Approach

To generate the text of a Dr. Seuss book, there are several elements to consider before we choose the best model. First, Dr. Seuss is known for his rhyming style; most of his works are written with an AABB or ABCB rhyme scheme in anapestic tetrameter. An anapestic tetrameter is a poetic meter that has four anapestic metrical feet per line. An anapest has three syllables. Each foot has two unstressed syllables followed by a stressed syllable. This means that each line has a rhythm like this: *da da DUM / da da DUM / da da DUM /da da DUM*. This kind of verse is the most common meter used for nursery rhymes and a majority of English poetry, so Dr. Seuss may have used this verse to improve the comprehension of his young readers. Here is an example of this rhyming style in Seuss's *Horton Hears a Who*:

On the FIFteenth of MAY, in the JUNgle of NOOL, In the HEAT of the DAY, in the COOL of the POOL, He was SPLASHing enJOYing the JUNgle's great JOYS When HORton the ELephant HEARD a small NOISE.

In his works, Dr. Seuss used several strategies to keep his rhymes from getting too regular. We can see one of his strategies above: in the fourth line, note that there is only one unstressed syllable before HORton, when there should be two. Seuss often subtracted an unaccented syllable from the beginning or end of a line. This small change is still in tetrameter, but it keeps readers from falling asleep. Seuss also chose different line breaks, sometimes breaking one verse into two printed lines. He also can break up lines differently to avoid hyphenating words and confusing his readers. Here's an example from his book *McElligot's Pool*:

From the WORLD'S highest RIVer
In FAR-off TiBET

It is important to note that Seuss would not break patterns in his verse in a specific pattern; rather, he made them by ear. He made each change with the intention to improve the experience of his readers.

The made-up names and words Seuss invented will be another factor to consider in choosing a model. Scientists have theories about why Seuss's words, such as "snunkoople," "yuzz-a matuzz," and "oobleck" are so funny: one theory suggests that they are more disordered, with unusual or improbable letter combinations. So, our text generator will have to deal with three things: a rhyme scheme that is not always consistent, making up words, and following Seuss's writing style.

One last consideration would be Seuss's tendency to impose creative constraints on his work. For example, *Green Eggs and Ham* used only 50 unique words total. Seuss did this for two reasons. First, he enjoyed the creative challenge. He also did this because the repetitve use of sight words–220 common words frequently found in printed material–help children learn to read. Dr. Edward William Dolch developed the sight words list in the 1930s-40s by studying the most frequently occurring words in children's books of that era. The list contains 220 service words (pronouns, adjectives, adverbs, prepositions, conjunctions, and verbs) plus 95 high-frequency nouns. Sight words represent over 50% of all English print media and $75-90\%$ in Dr. Seuss and other "learn to read" books. The book we end up generating should fulfill this constraint. It would also be interesting if we could limit the

words available to use.
Our primary approach could be a recurrent neural network because we believe it can best address the aforementioned considerations. RNNs make use of sequential information; we can think of them as having a "memory" about what has been captured so far. They have had great success in many NLP tasks, and RNNs have been trained to spell words, copy general syntactic structures, and generate text trained on a specific author. We could therefore work on hidden features that address the above considerations.
Now, we will discuss baselines and oracles for this project, which will set our lower and upper bounds on performance.

A. Baseline

As our simple baseline algorithm, we will generate Dr. Seuss books using an n-grams approach. This algorithm will create text that vaguely follows the Seuss style by nature of reproducing patterns of his words, but it is not able to follow a rhyme scheme or make up words as is.
Scanning through Dr. Seuss's seven most popular works, the n-grams approach reproduces words in the order in which they were used, meaning that the content of the generated sentences would be strictly limited to the order of the words in the books. Given these seven works, an n-value of 3, and an iteration length of 20, the n-grams approached produced the following pseudorandom output:

> [...] you did not know what to say. our mother like this? we don't know. and you may. try them and you may [...]

While the English in the output is technically correct and sounds somewhat "Seussian", there is no sense of context in the generated text whatsoever. Furthermore, there are many strict limitations to the n-grams generated text. Due to the nature of Dr. Seuss's works, in particular the short lines and many chained-together rhymes, n-grams has a tendency to produce rhymes by a factor of the simplicity of the words used, rather than intentionally. In addition, there is no infrastructure in place to support any kind of syllabic calculation, nor can it split text into lines intelligently. Despite these shortcomings, we believe that the n-grams approach is an appropriate baseline because it generates text with a distinctly "Seussian" quality, but without any specific features on top of that characteristic.

B. Oracle

While there is no pure objectivity to how "Seussian" certain text may sound, we believe that it is possible to evaluate this metric with reasonable accuracy.
First, we can consider Dr. Seuss himself in the loop, approving or rejecting the choices provided by our RNN. In this case, any glaring syllable structure and rhyme pattern digressions would be rejected implicitly, optimizing for the most "Seussian" text possible from the given generated text. Furthermore, Seuss would add randomly generated words for the purposes of both entertainment and fulfillment of the rhyme scheme, so keeping this in mind would create a strict upper bound for "Seussiness" of the generated text with respect to rhyming, syllable count, and random word generation.
We can also consider a child's perspective in reading the generated book, as there needs to be some kind of loose connection between individual thoughts and phrases at the minimum. A child would likely react better to a story than to a pseudorandom collection of sentences, so keeping this in mind as an upper bound for coherency is a useful exercise.

C. Evaluation Metrics

In order to evaluate the success of our modified RNN, we will identify and classify all test output in the following fields:

- appropriate rhyming schemes consistent with Seuss's work
- accurate syllable splitting in a specific poetic meter
- reasonable random word generation to fit rhyme scheme and meter and for creative purposes
- proper syllable stress with respect to the meter
- intelligent line breaks, as well as meter and rhyming scheme breaks at appropriate locations

One of the greatest challenges in working on the project will be creating a constrained RNN that will be able to adequately meet all of these criteria. In short, all of these criteria will likely be treated at first individually to develop a broad sense of how they will interact with the RNN, and will then be integrated one by one. Ideally, the input would consist of a corpus of Dr. Seuss's work as well as an approximate story length, with the output being a Seuss-style story of that length with all of the "Seussian" characteristics.

IV. Literature Review

In terms of understanding "Seussian" characteristics, current research discusses how although Dr. Seuss's books contain very liberal use of grammatical forms in creating imaginative language and vocabulary, Seuss was actually quite conscious of the use of highly frequent words from childrens literature. Seuss tackled the challenge to write a book for children in language they could understand. This and other analyses go deeper into Seuss's methodology and impact. Computer science research papers detail how to implement a recurrent neural network, the possible applications of these networks, and examples of text generated by these networks.
We were unable to find literature that covers all of the evaluation criteria we want to evaluate in this project, so we look forward to exploring the combination of various AI challenges throughout the quarter.

References

[1] J. Foster and C. Mackie, *Lexical Analysis of the Dr. Seuss Corpus*, *Concordia Working Papers in Applied Linguistics*, 2013.

[2] A. Graves, *Generating Sequences With Recurrent Neural Networks*, University of Toronto, 2014.

[3] J. Brownlee, *Text Generation With LSTM Recurrent Neural Networks in Python with Keras*, Machine Learning Mastery, 2016.

[4] I. Sutskever, J. Martens, G. Hinton, *Generating Text with Recurrent Neural Networks*, University of Toronto, 2011.

[5] A. Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*, karpathy.github.io, 2015.

[6] M. Michaels, *The Basics of Seussian Verse*, mickmichaels.com, 2012.

[7] J. Beck, *The Secret to Dr. Seusss Made-Up Words*, The Atlantic, 2015.