GRAMPS: Generating Really Awesome Metaphorical Poetry (Sometimes)

Description

We implement a word-level Recurrent Neural Net (RNN) to generate topical, rhyming poetry with embedded metaphors. We present our unique approach and provide some illustrative poems and commentaries. We argue our system has a greater degree of creative autonomy than most rival systems while producing poems that are novel, (sometimes) valuable, and intentional. Though the system may not consistently create valuable poetry independently, we believe it can be used by human poets for inspiration.

Introduction

“Why poetry you ask? Because of life, I answer.” -Dejan Stojanovic

Poetry is joy and sorrow, nature and self, life and death. It exists in every culture, and is the highest, most elevated textual art. Its aim is to lexically capture some fragment of the human experience, to convey emotion from the soul of the poet to the soul of the reader. Our work is then cut out for us: to give the machine a soul.

Machine-generated poetry is not a new topic (Lutz 1959, Manurung 1999, Gervas 2000, Levy 2001), and has been achieved at varying levels of success. In many ways, poetry generation is an easier task than other text generation problems because meaning is often intentionally veiled, cryptic, and open to interpretation, and random text may be confused with deep insights. Because the form of poetry can be fairly easily mimicked by a machine, it can be difficult for a human reader without contextual knowledge of a poem to discern whether the text was generated by a machine or a human, particularly when only provided with small samples. Previous machine poetry generation efforts have demonstrated this.

At the same time, insightful, intentional poetry that blends the noumenal with the phenomenal, the spiritual with the physical, remains elusive. For a machine to understand the human condition well enough to connect with the reader, to understand ideas and language well enough to mean what it says and say what it means while

subjecting these expressions to lyrical constraints remains as challenging as any endeavor in the field.

One critical feature of poetry previous researchers have largely neglected is intentionally embedding metaphor. If hifalutin verbiage is a vehicle for poetic expression, surely metaphor is the gasoline that fuels it. By viewing a subject through the lense of a metaphor, deep and complex aspects of meaning and of human experience are revealed. In our work, we seek to embed metaphor into our poetry in a natural, non-templated way.

To accomplish this, we leverage the generalization capacity of deep RNNs to capture the semantics of language, and creatively pair this with tools developed by Datamuse and Dr. Tony Veale to create topical poetry with intentional metaphors.

We argue that our poetry has some subjective value, and that our approach is, in some ways, creative. After placing our work in context of current work in poetry generation and after describing our model and its results, we detail some of the contributions our work gives to the community. Finally, we highlight areas with potential for future work.

Model

The soul of our system is a word-level recurrent neural network (RNN) based on Hunkim's Word-RNN implementation. To train the RNN, we downloaded over 2,000 poetry-related texts and anthologies from Project Gutenberg, with each file having the complete text of the printed version of the book. Consequently, we attempted to remove forewords, footnotes, printing information, Project Gutenberg specific header and footer information, tables of contents, indices, and other non-poetic text.

The collection of poems also spanned over thirty languages. We used langdetect, a Python port of Google’s language library, to restrict our corpus to English poetry. We further omitted texts where over 5% of words were not found in a standard English dictionary—this was done to exclude foreign language poetry that includes both the original text and the English translation, as well as to exclude Middle English poetry with inconsistent spelling conventions (e.g. The Canterbury Tales). We ignored files that were errantly encoded. After this initial cleaning, the size of the vocabulary for the dataset was still over 300,000 tokens—for comparison, a contemporary Oxford dictionary has only 220,000 words. The reasons for our burgeoning vocabulary are manifold, as our expanded vocabulary included mythical people and place names, unusual onomatopoeia, non-standard spellings, and the ever-present vowel-apostrophe substitution. While these features are an important component of classical poetry, we

restricted the vocabulary to the 50,000 most common words in order to simplify our model. The final vocabulary includes basic punctuation marks as well as the end-of-line character. After cleaning, we ended up with about 130 MB of poetry for training.

Architecture

Our cleaned poetry corpus is then fed into our word-level RNN. In keeping with our design goals, we opted for an RNN because it has considerable freedom in how it generates new text, while also being fairly successful in generalizing from the training data. In the following discussion of the model architecture, refer to Figure 1 for a visual depiction.

Typically with an RNN, a sequence of inputs is fed into the network with the hope it can predict the next token. In our case, each word in the vocabulary is assigned an index number, which is fed into a 256-dimension embedding layer. This embedding is then passed through two RNN layers, resulting in a single embedding output. Because each embedding is a convex combination, the output embedding can be directly multiplied by the universe of embeddings, where the norm of this product will be higher for more likely word candidates. A softmax is computed from these word scores to find the probability of each word being the next word in the sequence. The most probable candidate word is compared to the target word in the loss function. We experimented with different sequence lengths and batch sizes, though we found a 50-word sequence and batch size of 256 worked for our hardware configuration.

We experimented with using both GRU and LSTM architectures, but did not discern any significant difference in their outputs. While empirical text generation results using a basic LSTM demonstrate how well the model can learn to mimic the style and form of the input text, the output text is typically devoid of any real meaning. Since we wanted our model to intentionally write poetry about a particular topic, rhyme, and potentially embed metaphors, we added the end word of each line as an additional input feature. In this way, the model would learn to end a line with a provided word, allowing us to achieve some control over rhyme, topicality, and metaphor injection. We also provided the number of syllables as an input feature. In both cases, we randomly provided some null value to the model so that it could learn to generate meaningful text when these parameters were absent.

When text is generated, a target word for the end of each line can be chosen from a list. The contents of this list might depend on the desired output. For example, when rhyme is required, we can query Datamuse for words with the desired rhyme. On the other hand, when generating a free-verse line, we query Datamuse for words related to the

topic of the poem. For embedding metaphors in the poem, we query Veale's Metaphor Magnet to find words that view the poem's topic through the lense of the given metaphor.

In all cases, the target end word was ultimately chosen randomly from top candidates on the list, based on scores provided by Datamuse and the Metaphor Magnet. The chosen word is handed to the model as a target for the end of the line, but since our model is probabilistic and never fully constrained, GRAMPS is free to end with another word. This artistic freedom allows our agent to be more autonomous, while also giving it a larger breadth of creative possibilities when compared to other approaches.

For each line of the poem, GRAMPS generates 5 to 10 candidate lines and selects the "best" one to add to the poem. We used a blend of statistical likelihood provided by the LSTM and a hand-coded fitness function to select the best among generated lines. By default, the system favors short lines with common words. To combat this, we variously omitted both the most likely and least likely words from this statistical calculation, in an attempt to prune poorly constructed lines without excluding well-formed lines with a surprising word choice or two.

However, GRAMPS struggled to produce coherent lines when uncommon words were chosen as the end word target. These lines often exhibited excessive punctuation, missed the target word, had a highly irregular length, or simply tacked on the target word to a line otherwise unrelated to the target.

To alleviate this, we employed a simple fitness function to filter out the worst of the generated samples. This function had three components: syllable count, end word check, and punctuation count. Syllable counting simply penalized lines for every syllable more or less than the target count. End word check gave a penalty for not ending with the given target word; this ensured that most of the lines stayed on topic and rhymed as appropriate. Punctuation count penalized every punctuation mark appearing in the line. This was beneficial because GRAMPS occasionally inserted punctuation marks between most or all of the words in a line. We think this is because the model learned that separating words with punctuation reduced their dependence on each other, making it easier to meet the other demands on syllable count and end word. By penalizing punctuation, and by balancing the coefficients on the linear combination of these three criteria, we were able to get a simple function that filtered out the worst of the candidate lines.

The best candidate line is then appended to the poem, which is used to prime the RNN in preparation for the next line. For all of our experiments, we specified the number of lines to generate, though we consider having GRAMPS choose when to terminate a poem an important future refinement.

Results

Different versions of our model varied the amount the RNN was influenced by external sources. The various versions exhibited different strengths and weaknesses, illustrated in the examples below.

We first show how the model was able to learn syllable count, and guide each line toward a rhyming or thematically relevant end word in a fairly coherent way. The following example, primed with the subject “dirt,” shows success in accomplishing this while maintaining the theme:

```
That of the heart of mortal stain,
And I shall be to thee again.
I have been cleansed from the gravel.
My riddle I will unravel.
```

Here, the target end word “gravel” was chosen randomly from a list of words related to “dirt” as supplied by Datamuse. The word “unravel” was then chosen randomly from a list of words that rhyme with “gravel,” again supplied by Datamuse. Given the word “gravel” as target, GRAMPS suggests that one can be cleansed from such, demonstrating some measure of awareness regarding the meaning of the word and the theme of the poem. GRAMPS then recognizes that a riddle is something that can be unraveled, and hits this target word in the final line in precisely the eight syllables specified.

Another example shows GRAMPS choosing not to end a line with the suggested rhyming word, possibly because the model found a better fit given the fitness function. The poem below was given the topic of “friendship”:

```
Their memories and comradeship
Of the soul of human kindness.
The half of half of flirtation
In the same of awe and trepidation.
```

This shows that simply suggesting end words is usually enough to keep the entirety of the poem focused on one theme. Guiding a line to end with “comradeship” results in the use of the word “memories,” another word strongly associated with friendship. Similarly, suggesting “kindness” brings a mention of the human soul.

We can also prime the model on the first part of an existing poem, and have GRAMPS complete the poem. When constrained to follow the existing syllable count and rhyme

scheme, reasonable output was generated for the last stanza of Robert Frost's "The Road Not Taken":

```
I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I had so much work to satisfy
The road that had been in sad suspense.
```

GRAMPS completed this challenge with varying levels of success:

```
(early revision, without rhyming)
Two roads diverged in a yellow wood;
And sorry I could not take them both,
And be one traveller, long I stood,
I am a little man-forester.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
From the neighbouring lanes—oh! Why
Do hear the frog-beaver! Try me thence.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
He, who gives them no response so sly,
Across a long path, unseen, with joy intense.
```

We also attempted intentionally embedding metaphors in the poem. This was more difficult than previous experiments because we now required the poem to stay on topic, embed a metaphor, rhyme, and meet syllabalic constraints. We often found that subjecting GRAMPS to too many simultaneous constraints degraded the output. Consequently, when embedding metaphor, we often dropped the rhyming constraint. Even forsaking the rhyming constraint however, was often not enough to embed both a topic and a metaphor in four lines. To further encourage the incorporation of the metaphor, we variously changed end target words in the middle of a line as we sampled from the model. This simple trick was surprisingly effective in more densely packing disparate ideas.

The following poems were generated given the metaphoric conceit of a "child as a tempest."

```
Shrill and the roaring of the infant
```

```
That we crossed in the gale.
A whisper of the guilty son
Capped with the fiery storm.

The rush to the winds sprang threatening
For our enemy, the infant;
Echoes of war and the howling
Waves which yet took on the baby.
```

In incorporating words related to the metaphor and the topic, we find GRAMPS sometimes personifies the children as storms (as demonstrated by the “roaring” infant), and in other cases, he creates a scene with children in a tempest (e.g. when waves take on the baby). We consider both of these approaches valid embeddings.

We observe that the trick of changing target end words may have influenced the model to frame the infant as an enemy, as the first target word for this line was “threatening,” before it switched to infant. This infant-as-enemy conceit, though surprising, demonstrates surprising humanity for people familiar with infants.

While our attempts to metaphor and rhyme were generally less successful, the couplets below demonstrate that they are simultaneously achievable:

```
Demeter, my love, poor scholar,
Thou shalt ever dance for thy dollar.

You’re a bond like a pudding-sick pauper,
While I—I am a professor!
```

Finally, as one last humorous example, we present “marriage as death.”

```
I did suffer a bitter nightmare,
A drone of a voluptuous husband.
```

We believe these examples show that GRAMPS is successful in expressing ideas in creative metaphoric ways. This gives our output some value. Different people will value a particular artifact to different extents, but we feel that these examples demonstrate some understanding of some common human experiences.

To verify that our model was not plagiarizing the training set, we measured the proportion of 4- and 5-grams in our generated artifacts that also appeared in the training set. 4-gram plagiarism was 3.6%; while 5-gram plagiarism was 0.7%. We feel that these numbers are low enough to claim our poetry is genuinely novel.

Contributions

The primary contribution of our work lies in our unique method of generating poetry from an RNN that is metaphorical, novel, valuable, yet follows no templates and few hard constraints.

Because of this autonomy, GRAMPS has the freedom to come up with truly novel ways of expressing complex relationships between ideas. Poetry generated by GRAMPS portrays common human experiences in ways that readers can relate to, through use of apt metaphors, resulting in poetry with real meaning and value, even if the semantic structure is not always strictly correct.

This autonomy, however, may partially come at the cost of consistent coherence. The more constrained a poem is, the more samples were needed to find a coherent, grammatically correct one.

Consequently, in its current state, perhaps rather than masquerading as a stand-alone poet, GRAMPS might function as a smart poetry tool. Human poets could use GRAMPS as a source of inspiration, or perhaps as a tool to draft poetry, which can then be revised by the poet.

Future Work

There are many ways GRAMPS could continue to be improved. Many of these features we might have addressed had we been less constrained; others constitute projects unto themselves.

The most important refinement would be to improve the evaluation of the generated poetry. While our simple line evaluator was sufficient to filter out some of the worst lines, we had no way to measure the extent a line was coherent, let alone conveying something interesting, topical, or metaphorical, or whether flowed with the preceding lines.

While we attempted to filter both poems and lines of poems based on grammatical coherence using the the Python package “grammar_check”, we found it was not equal to the task. Though it seemed to be able to handle the most basic issues, such as subject-verb agreement, it struggled when the subject or verb was ambiguous (or omitted entirely as may be the case in poetry), and often allowed even the most senseless strings to pass.

One possible approach to remedy this is an LSTM grammar checker — sometimes generated lines would be really good if one word were replaced, or one more word were inserted. We might then train an LSTM by variously omitting words and replacing words with incorrect words, train it to recognize these situations and recommend the right word.

There are a couple ways the training data could be further refined. Occasionally, GRAMPS would fall into patterns of spewing out chains of single letters, or roman numerals, suggesting that our data was not sufficiently cleaned.

Another improvement to the training data would be to include a larger body of English text, including non-poetry, to widen the knowledge base of the model and expand the breadth of subjects it can cover. For example, G. K. Chesterton wrote that “Poets have been mysteriously silent on the subject of cheese” (Chesterton 1924). GRAMPS also had difficulty producing poems on that subject, as shown in the following example, presumably because there were no examples in the training set. Our best example of a poem on the subject of cheese follows. While it is not completely unrelated, we felt that poem lagged well behind some of the others in relevance and sense. Training on a more general corpus of text could overcome this limitation.

```
For very glad that I would give it up
Thy skin, nor milk, and whose cup
For the artifice of taste,
That many a cake takes in the waste.
```

As noted in the description of the model architecture, we also feel the model could benefit from a smarter selection of end-line words. Rather than choosing these randomly from a list at the time they are needed, it may be an improvement to compute a set of most likely topic words beforehand; and for each of these topic words, obtain a list of likely rhymes. Then during generation, an end word can be chosen according to its relevance to the current line, as well as its potential for future rhyme or its rhyming with a previous line.