Generating text with reinforcement learning

Maximilian Balthasar Mansky

October 17, 2020

Goal of this project

Machine learning approaches to text generation try to replicate a given text, for example by predicting the next word for a sentence. While the results can be interesting, this approach has obvious limitations.[1] Text can also be generated from structured information, such as information about somebody's life.[2] These approaches have the drawback that the machines only learns to imitate, in the sense that it reduces the distance between the generated text and the truth. This well-put in an article by Karpathy[3] and its response[4]. In this project, I want to explore whether a reinforcement learning approach provides different results.

Instead of attempting to generate text by imitation, we can check a generated solution for its correctness. This is done by implementing grammar, vocabulary and further checks in the environments that the machine interacts with. As an example, part of the environment might be responsible for evaluating word order, another for measuring word variations. Text is useful because it is easily verified by humans, compared to game solving strategies. It also allows for a wide variety of possible solutions. "The quick brown fox jumps over the lazy dog" is equally grammatically valid as "the lazy dog is jumped over by the quick brown fox".

Implementation

There are two parts to reinforcement learning, the agent and an environment. The agent interacts with the environment and receives feedback from it. Conversely, the environment receives input from the agent to change and evaluates the new state. The result of the evaluation is passed back to the agent and forms the basis for its learning.

A simple example is a state environment. Consider a five-state environment, consisting of a negative reward state, a positive reward state, two intermediary and a starting state.

-1		start		$+1$

Figure 1: A simplified example of an environment. The agent starts in the center cell and can move left or right. The leftmost cell contains a negative reward (denoted by -1), the rightmost one a positive reward ($+1$).

The agent then chooses its actions based on its current state. Possible actions are left or right (It is left to the environment to decide whether a state change is valid. Going left on the left-most cell is often interpreted as not changing state.) and the agent is given information about the result of that change (nil for changing to one of the center states, ± 1 for the outermost ones). How the agent learns is dependant on its implementation (i.e. whether it keeps a history, how the initial interaction is treated, how learning is implemented...). The environment is only responsible for evaluating the actions, the agent is responsible from learning from the feedback.

[1]

[2]

[3]

[4]

Environment

In this project, the Environment needs to check whether the current state (a text or sentence to be built by the agent) is correct and assign some form of feedback based on it. Each item of correctness will be implemented separately in grammar model. Possible models are:

- Sentence length
- Presence of nouns, adjectives, verbs
- Position of words
- Word order/grammatical correctness
- Sentence variation

As current agent learning frameworks can only work with a single reward dimension, the model output must be coerced into a single output by the Environment. It may be worthwhile to explore whether gradually returning the reward from more complex grammar models[5] is useful in guiding the agent. For evaluating the states, the Environment must at least keep track of the current sentence. In this context, a state means a particular sentence, not the individual words.

Within a state, words itself need to be encoded somehow. Either by using a one-hot encoding (limited to the most common words) or a small word2vec model based on a text. The latter has the advantage that the size of the vocabulary can be smoothly increased, by adding more words near to similar ones. In the latter case of a word2vec encoding, it probably also makes sense to include some sort of word accuracy reward, indicating how close the chosen word vector is to the vector output of the agent.

The actions changing states can be implemented in several ways. Either by generating a new state from scratch (that is, the agent generates a sentence for evaluation) or by manipulating the existing state, for example by changing the current word or the ones next to it. An action would consist of two parts: Deciding which to manipulate (current word, previous or next one or quit/finish sentence[6]) and choosing which word to use.

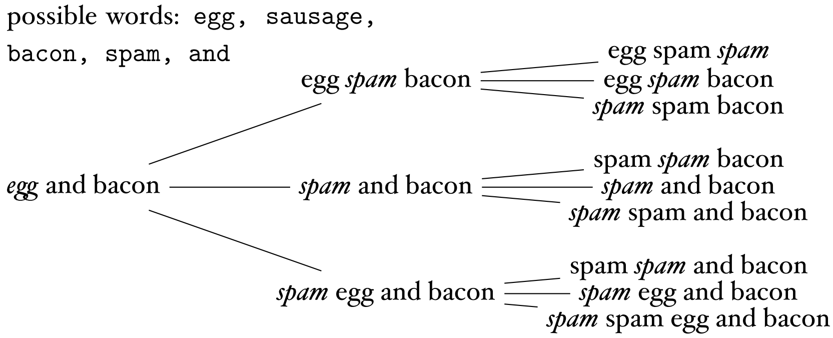

Figure 2: Example of state transfer from an initial sentence, with the possible words replacements indicated by *emphasized* text.

Moving between states can give different paths to the same state (for example ”spam spam bacon”), but the reward derived from it should be the same.[7]

[5] Implementation idea: Use some form of weighted average from the outputs. In the beginning the model may focus on producing sentences of the right length and with a bit of variation (80% sentence length, 20% word presence) and as learning progresses, focus more on the higher end of complexity (20% word position, 30% word order, 50% sentence variation).

[6] Maybe it is worthwhile to also include insertion of words? Or just move, do not change word?

[7] The Environment is ignorant of its previous states and keeps no memory.

References

[1] Lebret, Grangier, Auli: *Neural Text Generation from Structured Data with Application to the Biography Domain* `https://research.fb.com/wp-content/uploads/2017/02/neural-text-generation-emnlp-camera-ready.pdf`

[2] Lau, Cohn, Baldwin, Hammond *This AI Poet Mastered Rhythm, Rhyme, and Natural Language to Write Like Shakespeare* `https://spectrum.ieee.org/artificial-intelligence/machine-learning/this-ai-poet-mastered-rhythm-rhyme-and-natural-language-to-write-like-shakespeare`

[3] Karpathy: *The Unreasonable Effectiveness of Recurrent Neural Networks* `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

[4] Goldberg, *The unreasonable effectiveness of Character-level Language Models*`https://nbviewer.jupyter.org/gist/yoavg/d76121dfde2618422139`