Generative AI Part 2

Eckel, TJHSST AI2, Spring 2024

Background & Explanation

At this point, you should have a network running to train a text predicting RNN on your text file of choice. Now it's time to do the actual *generative* part of generative AI.

Generating Text

Write a script that retrieves your letter-to-one-hot-index dictionaries and your most recently trained network.

Decide on a seed string. I just used the letter "t", but you can use longer text or you can just send in a vector of all zeroes to let the network sort of freely guess the first letter. Feed your seed string through your network.

Your network should, at each step, output a probability distribution. Use random.choices with the optional weights argument to select a letter according to this probability distribution. Append it to your seed string. Then, turn it into a one-hot vector, and feed it back into your network, performing another step. Get a new probability distribution. Repeat!

We could end the assignment here – and in fact in my first draft I did – but there's a cool variation on this I want to teach you about.

Softmax "Temperature"

Just to avoid confusion let me clearly state at the beginning: **this applies to generating text only, and does not affect training in any way.** Let your training continue, unmodified, while you read & play with this.

Once your network is trained and you're generating text, you can modify the softmax activation function to do something cool. The modification is simply this – instead of softmax working like this:

```
temp = np.exp(dot_N_F)
a_N_F = temp / np.sum(temp)
```

…you add in another parameter, T, called temperature, and add it to your code so it does this instead:

```
temp = np.exp(dot_N_F / T)
a_N_F = temp / np.sum(temp)
```

(I am realizing now that my choice of "temp" as a temporary variable in Generative AI 1 might be confusing now that we're in this context; "temp" does not mean "temperature". "T" means "temperature". I will fix this the next time I give this assignment but it's too late now. Onward!)

This is a tiny change in your code, so you could just do it and move on, but developing some intuition here is a worthwhile exercise. Think about what this T parameter accomplishes – as T gets bigger than 1, the values of dot_N_F get smaller, which means that when those values are used as exponents, the results are *closer to each other* than they

would've been if T was just 1 (as in a standard Softmax). As a result, the probability distribution you get is *closer to evenly distributed*, or in other words, closer to random chance. Higher entropy. More chaos. In the other direction, as T gets *smaller* than 1, the differences between the probabilities in the distribution matter *more*, so there is *less* chaos.

In practical terms, what this means for this assignment is that if your network thinks the most likely next character is "e", then it is most likely to choose "e", but there is some probability it picks a different character. If the temperature goes up, the odds of your generator *not* picking "e" go up. If the temperature goes down, the odds of it *not* picking "e" go down. The lower the temperature, the more predictable the output.

This can have an interesting effect on your generated text! I trained mine up to about 2.5 million data points, and at that point, with temperature = 1, it still produced a lot of English-esque words that were not actual words. But when I set the temperature to 0.7, suddenly almost all the words were actual English words! That knowledge was encoded in the network, I just needed to accentuate it a bit by reducing the overall amount of chaos/entropy at each decision point.

And it's not just this assignment - this idea of "temperature" shows up a lot in actual, professional generative AI. If you look at the developer side of ChatGPT, the tools to build your own GPT that runs off of ChatGPT, the temperature is a parameter in ChatGPT's API that you can set! It does exactly what the temperature does here – makes the output more or less chaotic/random vs predictable.

Tweak Your Results Using Softmax Temperature

Before you submit this assignment, I'd like you to add this Temperature parameter to your generating code. It literally only affects the single line of code I showed above (and again, has no effect on the training process at all). But what a difference it makes!

Using your most-trained network, try a few temperature values, a little below 1 and a little above, maybe 0.7, 1, and 1.3. Maybe a few others if you want to. **Send me some results on Mattermost** – send me some text generated with the default T of 1, then send me some results generated with whatever T you think is the best. What kind of difference does this make to your output? Decide on your favorite value of T and hardcode it, then move on to the spec on the next page and get ready to turn in your assignment.

The spec is on the next page.

Specification

Make a .zip file. Put into it all of these files, then submit to the link on the course website:

- The .txt file you trained on.
- Your training code. This can either read directly from the .txt file and create test / training sets or read a pickled test and training set; either is fine. If you go for the latter, include those pickled files too. This should save a pickled network every 10,000 data points, each with a new filename.
- These **specific** three (3) pickled networks (don't include more):
    - Your network after 10,000 data points
    - Your network after 100,000 data points
    - Ideally, your network after 1,000,000 data points, but if you can't make it that far, I'll accept anything from 300,000 on up.
- Optionally: if you train a lot longer, include your best network also.
- Your testing code configured to do all this, automatically, no command line args, when I run it:
    - Read in the 10k data point network. Generate 400 characters of text.
    - Then, read in the 100k data point network. Generate 400 characters of text.
    - Then, read in the 1m data point network (or as close as you got). Generate 400 characters of text.
    - Then, optionally: read in your best network. Generate 400 characters of text.
    - Make sure each of these four generations is distinctly clearly labelled in your output. Any seed string is fine.
- A txt, doc, or pdf (no gdocs) where you answer these questions:
    - How difficult was this lab? How interesting was this lab?
    - Does 2 RED credits for writing all this seem about right?
    - Should I keep this unit in future versions of this course?