Programming for Computational Linguistics 2018/2019

Final Project Checkpoint 5 — `generate`

These checkpoint documents provide a detailed recommendation for how to proceed with the project. They will break the requirements into small, well-defined steps. It is not necessary to follow this plan — you can choose to implement the requirements however makes the most sense to you. However, if you are unsure of how to proceed, this should provide you a good starting point.

Here is the fun part! At this point, we can take any sequence of tokens, and predict the probabilites for what word comes next. Generating text is not too difficult from this point.

As before, we will write one helper function before we write `generate`: `sample(distribution)`. This function (or method) should take as input a probability distribution — that is, a dict whose values are numbers which all add up to 1. It should return a key from the input dict, chosen according to its probability. For example, `sample({'heads': .5, 'tails': .5})` should return `'heads'` 50% of the time, and `'tails'` 50% of the time. This function is actually kind of tricky, but is solvable using just `random.random()`.

Once you've written `sample`, `generate` is actually really straightforward. Start with an empty list of tokens. Use `p_next` to get the probabilities for the next word, use `sample` to pick a word according to those probabilites, append that word to your list of tokens, and keep going until you predict `None`, at which point you return your list of tokens. Implement this method.