Language Generation Exercise

Natalie Steepleton
Westmont
nsteepleton@westmont.edu

INTRODUCTION

For this project I attempted to create a stochastic language generation program that uses an n-gram model to predict the next word in a randomly generated text. To accomplish this I was given a corpus of texts to manipulate. From there I processed the text by removing certain characters and punctuation, then assigned words in the text to probabilities based on what words are likely to come after what words based on an n gram model. To run my program one would instantiate a language generator (LanguageGenerator) object and pass that it an integer n for how long the user wants the n gram model to be based off of, a String (null) and the length of how long the user wants the randomly generated text to be. The program will take these arguments, randomly generate the first word and then generate the text up to length l using an n-gram model of length n.

PROCESS THE CORPUS

My program reads in the corpus of texts by reading in a text file using a string representing the directory to where the document belongs. As the file is being read in from the specified directory line by line commas, and slashes are removed. Further improvements on this project could create a more sophisticated way to process the text and handle punctuation. Furthermore, the capitalization on letters could be unified to lowercase so that after the random language is generated there could be another method to predict where punctuation should be added and letters should be capitalized.

N-GRAM MODEL

This project uses an n-gram model to predict words for the randomly generated sentence. An n-gram model is a type of probabilistic language model for predicting the next item in a sequence. In this example the sequence was the corpus of texts and the items were words in the corpus. Therefore the n gram model was use to predict what word would likely follow some given string of words.

Attempt 1 at n-gram

To implement an n gram model in this project I initially tried to use Bayes theorem to calculate A given B where B was the text that was already seen up to n words and A were the words in corpus that were likely to follow B. using this model I attempted to assign every word in the corpus to a value, so that every word would have a probability, but those that were seen in the corpus as following "B" would be assigned a higher probability then the other words in the corpus. I had a very hard time making the n-gram model work using this approach and eventually ended up abandoning this attempt and trying something different.

Attempt 2 at n-gram

The n-gram model that is working in my project now goes through the entire corpus every time before a new word is added to the randomly generated text. As it loops through the corpus it looks for words following the last n words that were generated that appear in the text. It creates a list of all of those words (even when there are repeats) and then randomly chooses the next word to be added to the randomly generated text using the list of words. This seems to work but in my opinion it is a sloppy approach.

OTHER IDEAS

To improve this project I would start by expanding on the way that the texts are processed. This would potentially make the randomly generated text is more believable because the text would not have random punctuation like it does now. Second I would try to implement my first approach at the n gram model and use Bayes rule or a different algorithm so that all words would be assigned a regardless if the corpus contains the preceding text. I think that this would make the randomly generated text more interesting.

REFERENCES

[1] Russell, Stuart J., and Peter Norvig. Artificial Intelligence: A Modern Approach. Englewood Cliffs, NJ: Prentice Hall, 1995. Print.