CSE 368 - AI: Markov Models for Text

Here we will use n-gram transition probabilities to both generate and analyze text. The class `Ngrams` implements many useful helper functions and comes with some processed n-grams that you can use. There is a function called `slurpFile()` that will take text files (i.e. from gutenberg.org) and generate the n-grams. To save processing time, the handout files have a stored set of n-grams and you don't need to make new ones.

1) **(50) Text Generation** Write the function `makeSentence(n=5)` that takes an integer and generates sentences by conditioning the probability of the next characters on the previous n characters. For example, an argument of zero means that each character is drawn from the character frequencies. The sentences up to `n=4-5` should be mostly gibberish and then suddenly produce text once the `n` increases.

There are a few glitches that can happen. For example, you could run into an n-gram that has not next letter! This happens if it was at the end of some training text. One effective strategy is to 'back off' and try to generate a character using only the $n-1$ previous characters, etc. This should always succeed, since in the worst case you just use single character statistics.

Write a function `grow(text, gramLen=4)` that adds a character according to the `gramLen` statistics to the end of the text. It should use the gram (at the end) to sample the next character. The sentence generation function will use grow repeatedly to make text until it has made reasonable sentence.

1) **(30) (Enter this text box) Deciphering Text** Use the n-gram statistics to decrypt the text stored in `scrambledText`. It starts with: `xvkndui?d).xc,) nwwnxqv)x ?c,)wxvi)xfjnw)nd)xd)?jwx ....` Each character has been scrambled by substituting each occurrence of a character with a different one from the available set in `chars`. Find the two strings, `orig`, `subs`, so that the function `substitute(orig,subs,scrambledText)` returns the unscrambled text. In order to receive credit you will need to be supply these two strings. You can do this by hand, using the text statistics tools to analyze it.

3) **(20) Automatically Deciphering Text. (Extra Credit)** Write a function that analyzes a new sample of text where characters from the map have been randomly permuted and try to un-scramble it. The function `decode` should take in scrambled text and return two substitution strings. (This is somewhat redundant since we could always assume that the `orig` string is always the list, `chars`).

Have fun, and happy hacking!