INF144- Obligatorisk oppgave

Generate random text

Zeroth order approximation:
The letters are chosen completely independently of each other. This makes it look mostly like junk, and the character <space>, appears most often. You can discern one or two real words, like “og”, or sometimes larger words like “tre”.

To run the zeroth approximation Markov chain you need the files:

- Folktale.txt
- MarkovZeroth.java

Run these commands:

```
javac MarkovZeroth.java
java MarkovZeroth
```

First, second, and third order approximation:
When the letters are chosen dependently of each other, more meaningful text appears. For example, during my testing, I’ve often used the starting state as “a”, “as”, or “ask”, this is the start of the name Askeladden, which is a rather common word in the source file. This makes it so that it’s not uncommon to see the first set of characters be close to the name.

The text is generated from MarkovOblig, when run with the parameters 1, 2, or 3 it will generate text with the corresponding approximation of the Markov model. You will then be asked to provide a starting state: the length of this will need to match the appropriate approximation.

To run the 1st, 2nd, and 3rd approximation Markov chain, you need the files;

- Folktale.txt
- MarkovModel.java
- MarkovOblig.java
- Bag.java

These commands would work for a 3rd approximation:

```
javac MarkovOblig.java
java MarkovOblig 3
ask
```

1. Do you recognize some real words in the random text?
Yes! The higher the approximation, the more words are recognizable.

2. Are the Markovian information sources unifilar, and if so explain why they are unifilar.
Yes! The sources are unifilar because they are reachable, in one step, from a common prior state.

Lempel-Ziv-Welch and Huffman Compression

The Lempel-Ziv-Welch and Huffman compression are implemented in the files LZWHuffman.java, and Huffman.java. Splitting the compression into two files made it much more clear to read. After all files are compiled. Simply running the command: “java LZWHuffman” will execute the compression of the file Folktale.txt. The program will create two new files, one compressed using only LZW compression, and another using both LZW and Huffman compression on top.

The average compression rate without Huffman compression, sadly, was no larger than 1.034. After an extensive search, I have no idea why the compression isn’t better.

The average compression rate with Huffman compression, is a disaster. It takes more bits than the original to store… I believe that I’ve located that problem with all the <space> characters in the LZW compressed file. I have no idea how to fix that, so sadly that leaves us with a “compression rate” of 0.94.

I will continue to research this, and hopefully can have some successful compression later. For now, this assignment must be turned in.