Random Text

Jon Sporring

October 17, 2019

1 Lærervejledningn

Emne Functional programming, histograms, random values

Sværhedsgrad Middel

2 Introduktion

H.C. Andersen (1805-1875) is a Danish author who wrote plays, travelogues, novels, poems, but perhaps is best known for his fairy tales. An example is Little Claus and Big Claus (Danish: Lille Claus og store Claus), which is a tale about a poor farmer, who outsmarts a rich farmer. A translation can be found here: `http://andersen.sdu.dk/vaerk/hersholt/LittleClausAndBigClaus_e.html`. It starts like this:

> Hans Christian Andersen's "Lille Claus og Store Claus" translated by Jean Hersholt.
>
> In a village there lived two men who had the self-same name. Both were named Claus. But one of them owned four horses, and the other owned only one horse; so to distinguish between them people called the man who had four horses Big Claus, and the man who had only one horse Little Claus. Now I'll tell you what happened to these two, for this is a true story."

In this assigment, you are to work with simple text processing, analyse the statistics of the text, and use this to generate a new text with similar statistics.

3 Opgave(r)

1. The script `readFile.fsx` reads the content of the text file `readFile.fsx`. Convert this script into a function which can read the content of any text file and has the following type:

```
readText : filename:string -> string
```

2. Write a function that converts a string, such that all letters are converted to lower case, and removes all characters except a...z and space. It should have the following type:

```
convertText : src:string -> string
```

3. Write a function,

```
histogram : src:string -> int list
```

which counts occurrences of each lower-case letter of the English alphabet in a string and returns a list. The first element of the list should be the count of 'a's, second the count of 'b's etc.

4. The script `sampleAssignment.fsx` contains the function

```
randomString : hist:int list -> len:int -> string
```

which generates a string of a given length, and contains random characters distributed according to a given histogram. Modify the code to use your histogram function from Exercise 3.

Further, write a program, which reads the text `littleClausAndBigClaus.txt` using `readText`, converts it using `convertText`, and calculates its histogram and generates a new random string using `histogram` and `randomString`. Test the quality of your code by comparing the histograms of the two texts.

5. Write a function

```
cooccurrence : src:string -> int list list
```

which counts occurrences of each pairs of lower-case letter of the English alphabet including space in a string and returns a list of lists (a table). In the return list, the first element should be a list of the counts of 'a' being the initial character, i.e., how many times "aa", "ab", "ac",...,"az", "a " was observed. The second list should containt the counts of combinations starting with 'b', i.e., how many times "ba", "bb", "bc",... was observed and so on. The function should include overlapping pairs, for example, the input string "abcd" has the pairs "ab", "bc", and "cd".

6. Write a function

```
fstOrderMarkovModel : cooc:int list list -> len:int -> string
```

which generates a random string of length `len`, whose character pairs are distributed according to a user specified cooccurrence histogram `cooc`.

Use the function developed in Exercise 2 and 5, and test your function by generating a random string, whose character pairs are distributed as the converted characters in H.C. Andersen's fairy tale, "Little Claus and Big Claus". Calculate the cooccurrence histogram for the random string, and compare this with the original cooccurrence histogram.

7. Write a program that counts occurrences of each triple of lower-case letter of the English alphabet in a string and returns a list of lists of lists. The program must include the function

```
triOccurrence : src:string -> int list list list
```

to calculate the number of occurrences of triples.

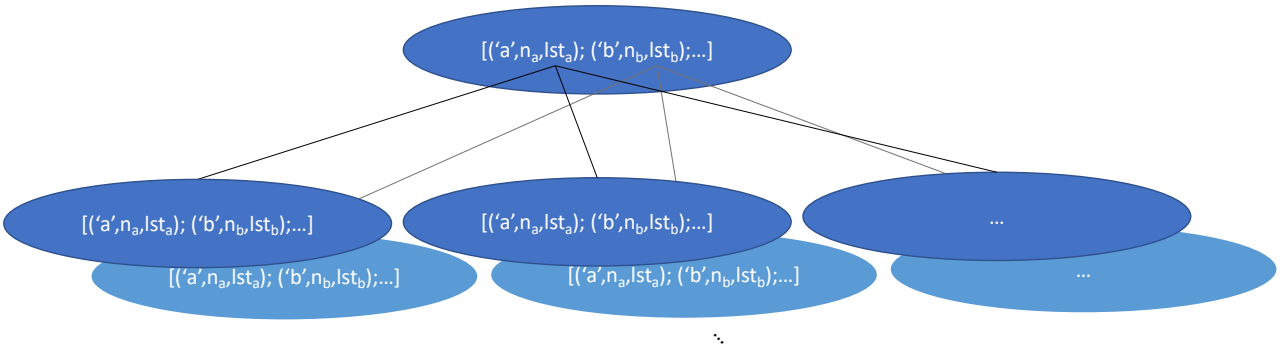

Figure 1: An illustration of a list of values of the type `Tree`.

8. Write a program that generates a random string of length `len`, whose character triples are distributed according to a user specified trioccurrence histogram `trioc`. The function must have the type:

   ```
   sndOrderMarkovModel : trioc:int list list list -> len:int -> string
   ```

   Use the function developed in Exercise 7, and test your function by generating a random string, whose character triples are distributed as the converted characters in H.C. Andersen's fairy tale, "Little Claus and Big Claus". Calculate the trioccurrence histogram for the random string, and compare this with the original trioccurrence histogram.

9. Write a function that counts occurrences of each word in a string and returns a list. The counts must be organized as a list of trees using the following `Tree` type:

   ```
   type Tree = Node of char * int * Tree list
   ```

   An illustration of a value of this type is shown in Figure 1 Words are to be represented as the sequence of characters from the root til a node. The associated integer to each node counts the occurrence of a word ending in that node. Thus, if the count is 0, then no word with that endpoint has occurred. For example, a string with the words "a abc ba" should result in the following tree,

   ```
   [Node ('a', 1, [Node ('b', 0, [Node ('c', 1, [])])]);
    Node ('b', 0, [Node ('a', 1, [])])]
   ```

   Notice, the counts are zero for the combinations "ab" and "b", which are words not observed in the string. The function must have the type:

   ```
   wordHistogram : src:string -> Tree list
   ```

   Write a program which reads the text `littleClausAndBigClaus.txt`, discard all characters that are not in `['a'..'z', 'A'..'Z',' ']`, convert all the remaining characters to lower case and calculate the occurrence of the remaining words as a `Tree list` type.

10. For a given value of a `Tree` type, see Exercise 9, write a function

    ```
    randomWords : wHist:Tree list -> nWords:int -> string
    ```

    which generates a string with `nWords` number of words randomly selected to match the word distribution in `wHist`.

Use the function developed in Exercise 2 and 9, and test your function by generating a random string, whose words are distributed as the converted characters in H.C. Andersen's fairy tale, "Little Claus and Big Claus". Calculate the word histogram for the random text, and compare this with `wHist`.

11. Write a short report, which

    - is no larger than 5 pages;
    - contains a brief discussion on how your implementation works, and if there are any possible alternative implementations, and in that case, why you chose the one, you did;
    - includes output that demonstrates that your solutions work as intented.