Version 1.2 ***due:*** *Mon, Feb 18 @ 11:59pm*

1 Introduction

Text omitted for brevity.

2 Files

Text omitted for brevity.

3 SEQUENCE_UTIL

3.1 String Tokens

Task 3.1 (10%). *See code solutions.*

3.2 Histograms

Task 3.2 (10%). *See code solutions.*

Task 3.3 (10%). *See code solutions.*

Task 3.4 (5%). We want to generate a data structure representing the cumulative distribution function. Thus we can first generate a sequence of prefix sums where each `'a` is associated with the sum of frequencies of tokens that come before it. We can then divide each prefix sum by the total sum to get a `real` between 0 and 1, representing a CDF.

```
exception EmptyHist
exception RealRange

type 'a cdf = ('a * real) seq

fun preprocess (histogram : 'a hist) : 'a cdf =
  if length histogram = 0 then raise EmptyHist else
  let
    val prefixSums = scanI op+ 0 (map #2 histogram)
    val total = Real.fromInt (nth prefixSums (length prefixSums - 1))
    val normalized = map (fn i => (Real.fromInt i) / total) prefixSums
  in
    map2 (fn x => x) (map #1 histogram) normalized
  end
```

We can then implement `choose` in $O(\log n)$ work and span by filtering out all `'a`s with CDF value less than the input r and then taking the first value of those greater than or equal to r:

```
fun choose (cumulative : 'a cdf) (r : real) : 'a =
  if   (r < 0) orelse (r > 1)
  then raise RealRange
  else nth (filter (fn x => x >= r) cumulative) 0
```

3.3 Testing

Task 3.5 (5%). *No official solution provided.*

4 K-Gram Stats

Task 4.1 (2%). *See code solutions.*

Task 4.2 (23%). *See code solutions.*

Task 4.3 (5%). *See code solutions.*

4.1 Testing

Task 4.4 (5%). *No official solution provided.*

5 Babble

5.1 Implementation

Task 5.1 (10%). *See code solutions.*

Task 5.2 (5%). *See code solutions*

5.2 Babbling

Text omitted for brevity.