Project Proposal for CSE549
Fall 2011

Reconstructing Books using Google Ngrams

Himanshu Jindal
Piyush Kansal

Contents:

1. **Objective:**
   The objective is to reconstruct books for a given year using Google Ngrams with following factors in mind:
   a) Reconstruction accuracy
   b) Extent to which one can reconstruct
   c) Running time

   To start with, we are suppose to pick a book (.txt file) from Project Gutenberg, create a 5grams file out of it just like Google does and then try reconstructing it. If it is able to achieve the desired accuracy, then apply the same on Google Ngrams and check how far the reconstruction goes.

2. **Implementation Concept**
   We have come up with following:
   a) After getting all the 5gram data sets from Google, filter them and extract 5grams only for a given year
   b) Then apply one of the following reconstruction strategy, whichever gives better results based on above three factors

   We have thought of following reconstruction strategy using 5grams:
   a) Implementation using Hash tables
   - Scan all the 5grams and store them in a list (using STLs in C++)
   - Read 5gram from the list one by one until just one is left
   - Match the last four grams of this 5gram with the first four grams of an another 5gram in the list
   - If the maximum match is found, form a super string out of it and store it in the same location from where first 5gram was read
   - This way, the length of this super string will increase to 6. So, we will now its take last 4 grams and again above steps till we found a mismatch. At that point we will move to a different 5gram and repeat the above steps. Finally, our list will have just have 1 super string left and that will represent the book
   - To avoid searching in the whole list (which will be really huge), we will match the very first character of the 4gram (last four grams of current 5gram) with all 5grams starting with the same character in the list. To make this search fast, we have thought of following:
     - Maintain a hash table for 128 ASCII characters. Each entry of this table consists of following:
       - an integer value of the ASCII character
       - iterator or starting address of the list from where all the 5grams whose first character matches with this character are stored
       - a count of all of such 5grams
   - The combination of above three will help in localizing the search area

- Although, in this list some of the characters will never show up like non-printable characters. But this is just to make the access to hash table in O(1) time.

b) Implementation using Tries

- We think that the above approach of searching and then matching can be optimized using Tries
- Instead of using hashing, we can store all the 5grams in a word trie
- This can give us very fast lookup time and can save the time which we are spending in matching with each of the 5grams in a given area in the list

3. **What has been tried**

a) To start with, we have taken a book from Project Gutenberg
b) We have written a program to shred this book in 5grams, sort it and then create a Google like data set with number of occurrences on each 5gram appended to it
c) We passed this data set to our reconstruction program but is not giving us the desired output. The program terminates after running for sometime with segmentation fault. We have localized a bug in our hash table implementation. We are working on it and expect to resolve it soon

4. **What is to be tried**

a) We have to evaluate the above program on the three factors listed above
b) In parallel, we will develop another program to implement it using Tries and will then compare the final results on the book from Project Gutenberg
c) We are expecting issues with the reconstruction accuracy and the extent of reconstruction and are yet to think of applying any heuristics, if possible