NETBASE

Platform Development Question

Please develop a client/server applications based on below requirements to demonstrate your ability in implementing a reliable multi-threaded server.

I. Design a client (ClientA) which –
   1. Reads the attached txt file.
   2. Submits a network request containing the text in the attached file to server and waits for the bi-gram probability calculated and returned by the server. Please refer to next page for the bi-gram calculation.

II. Design another client (ClientB) which –
   1. Sends a message containing a string to the server once per minute, without waiting for reply.

III. Design a **multi-threaded** server which –
   1. Accepts requests from the two above clients.
   2. Those requests fall into two categories:
      A. Bi-gram requests sent from multiple ClientAs.
      B. String message sent from ClientB. The server should handle these requests ASAP by printing the string to standard error.

   - The two categories of messages are sent to the same channel (end-point), either same server port or same message queue.
   - Each bi-gram calculation request is dispatched to one server thread.
   - The maximum number of the threads the server can support is 12.

IV. Create a test case to calculate and validate the bi-gram probability of the attached txt file.

Bi-gram Probability Calculation

The Spam Filter

When crawling the internet, we often encounter meaningless spam pages. A typical spamming technique is keyword stuffing, which stuffs a page with popular keywords, like “mp3” or “ipod,” to increase its ranking in search engine results. We want to filter out such pages. Keyword-stuffed pages usually contain lots of machine-generated content, thus have less proper English sentences. Theoretically we may analyze the grammatical and semantic correctness for each sentence by natural language processing, but this would be computationally expensive. A lightweight alternative is to use a statistical technique, looking for probabilistic local consistency. We segment each document to *n*-grams of *n* consecutive words, where *n* is a small number such as 2, 3 … We define the probability of the *n*-gram $w_{i+1} \dots w_{i+n}$ starting at word i+1 to be:

$$P(w_{i+1} \dots w_{i+n}) = \text{number of occurrences of the } n\text{-gram}$$

Note that *n*-grams are overlapping. For example, the third word of a document is covered by the first, second and third tri-gram, if the document has at least five words. Although they are overlapping, to simplify the computation we assume each of them is chosen independently to each other. We than define the probability of a document with *k* *n*-grams (and hence k+n-1 words) to be the product of the individual probabilities, normalized by taking its *k*-th root:

$$\sqrt[k]{\prod_{i=0}^{k-1} P(w_{i+1} \cdots w_{i+n})}$$

Example

We are computing the bi-gram probability for a document consisting of a single sentence:

Don’t cry because it is over, smile because it happened.

By the definition above, *n* equals 2 and *k* equals 9. The bi-gram probability can be calculated by:

(P(dont cry) * P(cry because) * P(because it) * P(it is) * P(is over) * P(over smile) *
P(smile because) * P(because it) * P(it happened)) ^ (1/9)
= (1 * 1 * 2 * 1 * 1 * 1 * 1 * 2 * 1) ^ (1/9)
= 4 ^ (1/9)
= 1.1665290395761165

Requirements

As an initial step of the spam filter project, you will write some code to read sample data from the attached text file, and calculate its *bi-gram probability*. Let’s assume (1) words are case insensitive, and (2) words consist of letters and digits only. Please include the bi-gram probability, rounded to the nearest hundred thousandth (i.e., 5 decimal points), and the value of k in your solution.

Feel free to read reference manuals or search the internet.