Java Backend Software Engineer

*** Please upload your solution in a single archive (e.g. solution.zip), which contains the results of **1) The Spam Filter** and **2) The Spam Filter v2**. Please use English to complete the test.***

1) 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 frequency of the *n*-gram $w_{i+1} \ldots w_{i+n}$ starting at word i+1 to be:

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

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 then define the probability of a document with *k* *n*-grams (and hence k+n-1 words) to be the product of the individual n-gram frequency, 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. Feel free to read reference manuals or search the internet.

Result contains (1) a text file with your answer and (2) runnable Java source code. The answer should include the bi-gram probability, rounded to the nearest hundred thousandth (i.e., 5 decimal places), and the value of k in your solution.

2) The Spam Filter v2

Our system needs to process lots of documents, and we want to speed up processing through concurrency.

- Image that the program will be given a directory, which contains multiple text files, up to thousands. The directory's content is guaranteed to be static during processing.
- Calculate each text file's bi-gram probability independently.
- Maximum bi-gram processing threads is 12.
- Print each bi-gram result to STDOUT, one result per line, in format of - `k, bi-gram probability, occurrence of (k, bi-gram probability) pair so far`

  Example output:

```
9, 1.16653, 1
20, 1.23456, 1
30, 1.34567, 1
9, 1.16653, 2
10, 1.12345, 1
```

- The program can handle interrupt in the middle of processing and shut down gracefully.

Requirements

Result contains runnable Java source code.