Programming Question

BBN (Beltway Bandits, NA) is building a query tool to help their grant writers find old proposals to reuse for new proposals. Your part is a "query improver" which should take a query and suggest better search terms to use. In particular, it should suggest more specific terms, e.g., "neural networks" and "expert systems" instead of "artificial intelligence," but only when such terms might actually discriminate better between documents, based on the TFIDF (term frequency/inverse document frequency) scores for search terms, **averaged over the current set of documents**. The TFIDF of a term *t* and a document *d* is calculated by:

$$TFIDF(t, d) = TF(t, d) * \log(N / DF(t))$$

where $TF(t, d_i) = \log(count(t, d_i)) + 1$, i.e., count occurrences of *t* in d_i and $IDF(t) = (N + 1) / N(t)$, where N = total number of documents (currently 372, but increasing), and $N(t)$ = number of documents containing *t*.

You have two data files available. First, an inverse index file for every search term, with lines like this:

term N(term) d_1 count(term, d_1) d_2, count(term, d_2) …

E.g.,

```
neural-networks 4 nsf0198 32 darpa1101 65 nsf0499 8 nih0797 10
```

Second, a file listing the specializations of terms, with lines like this:

```
artificial-intelligence expert-systems neural-networks …
expert-systems CBR OPS5 backtracker …
```

In Java or Lisp, write code to take a search term and return a list (any kind of collection) of more specific terms with higher average TFIDF's. Note that your code needs to do the TFIDF calculations, but you decide what to calculate in advance vs. at query time, and even what doesn't need calculating at all for this given problem. Write clean, top-down, well-structured code. Focus first on the core algorithms, and leave easy items, like accessor methods and file I/O, for last.