FastRegexp

One use case we had was the need to perform very fast regular expression queries against the term dictionary.

The easiest improvement was to integrate Brics Automaton package around the RegexCapabilities interface. This gave a significant improvement in performance (because of the underlying DFA) but was not enough. The biggest problem was that most of our regular expressions do not have any constant prefix. We are searching Arabic which means a common regular expression would look like (|) at the beginning of a word because hamza is so frequently omitted. This meant it was doing regexp comparison on the entire term dictionary (although still significantly faster than the JDK implementation)

The final trick was to take advantage of the fact that Brics allows you to do a 'step' operation against a built regular expression and determine if it is in a reject state of the DFA. Because of this, if you organize terms into buckets by their first N characters (N=1 or 2 is typically enough) then you can simply use this step operation against each bucket "prefix" and only do the full regex comparison against the items in buckets that will not definitely end in a reject state.

For our implementation we simply loaded up all the terms into this bucket organized structure at startup with a TermEnum, but it might be very likely that the way terms are stored in Lucene is compatible with this trick and a more elegant integration method would work. Obviously this does not help for regular expressions where the beginning of the string can match *anything*, but it is more flexible than requiring just a constant prefix for reasonable performance.

On a 400M doc index with 2.1M unique terms we can do a regex expansion in 15ms on average with this method on my desktop computer.