ElasticSearch Query Expansion

Relevance Models

It should be possible to create an RM1/RM3 by manually fetching the term vectors for an initial retrieval. This is untested, but based on reading documentation the process for the web app or other code interfacing with ElasticSearch would be:

1. Issue original query (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-body.html)
    a. Specify number of feedback documents (default is 10)
    b. `{ "query": { ... }, "size": fbDocs }`

2. Receive list of hits and record ID and score:
    a. "hits" field contains list of "hits" (see link in 1a above)
    b. For each hit in "hits," save "_id" and "_score" (assuming index has been set to LMDirichlet similarity so that "_score" is QL)
3. Request term vectors for each hit (https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-termvectors.html, https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-multi-termvectors.html)
    a. Fetch all term vectors at once:
        i. `POST /<index>/_mtermvectors { "docs": [ { "_id": "1" }, { "_id": "2" } ] }`

4. Receive term vectors
    a. Each term in the response will have a "term_freq" field with its frequency in the document – store this.
5. Construct RM1/RM3 from term vectors
    a. We need to sum the "term_freq" values to get document length, otherwise all the data we need is available (term frequencies and document QL scores)
6. Issue expanded query
    a. This is the tricky part. We can either use boosting or write a custom score script:
        i. Boosting implies relative importance of query clauses, but whether this equates to proper term weighting is unclear/doubtful.
        ii. Custom scoring allows us to write a negative KL-divergence scorer to score each document. We should be able to access term statistics in the script to enable this by passing in the specific terms and their weights as parameters. It'll be rough, though.
7. Receive final search results