A Simple Version of BM25 in Postgres

Di Qi
Lantern

What is BM25? The TLDR version

- BM25 is the most popular algorithm for text search - used in ElasticSearch
- It considers term frequency within a document, and term frequency across the all documents.
- Uncommon terms are ranked higher

The essentials of BM25

- N - Total number of documents
- df - Document frequency per term
- tf - Term frequency in documents
- avgdl - Average document length

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\text{avgdl}}\right)},$$

Try 1: Naive implementation

- Auxiliary tables with doc_id, term, tf, doc_len
- Too slow! Poor performance on large datasets
- Excessive index lookups for common terms

Try 2: Inspired by columnar storage

- Auxillary table containing a single row per term.
- Each row contains the term, doc_ids[], fqs[], doc_lens[]
- Reduces index lookups to one per term in the query

Try 3: Custom aggregate function

- Calculate BM25 scores from arrays with custom aggregate function
- Calculates multiple metrics simultaneously - unlike SUM or AVG

Performance metrics

- Evaluated on 500k Quora dataset
- Naive method: Unusable due to long execution time
- JSON aggregation: ~3 seconds
- Custom aggregate function: ~1 second

Key takeaways

- You can implement a basic version of BM25 in Postgres with just SQL
- You can implement an acceptable version of BM25 with just PLRust, a trusted PL
- You can make this even faster with privileged PLs to build custom functions