**Document Search Engine**

3 steps process:

1. Query filtering – Token removal, stemming and converting to lower case.
2. Query augmentation – Adding synonyms and acronyms contraction/ expansion.
3. Document scoring – Score for documents based on query.

**Detailed explanation for each step:**

**Query filtering:**

Token removal: Avoid storing common terms (stop words).

Tokenizers: Components that break a steam of test into words, phrases, symbols, or other units called tokens.

**Query augmentation:**

Generating synonyms-

* To improve the number of relevant results for a query is to use synonyms.
* Synonyms allow you to expand the number of potential ways a query or piece of indexed document id expressed.
* Allow people to express the same concept in different ways- and still retrieve the same search results.

Methods to generate synonyms:

1. Using python NLTK module “wordnet”
2. Generating Word Embedding and then calculating similar words using cosine similarity.

**Document scoring:**

Two approaches:

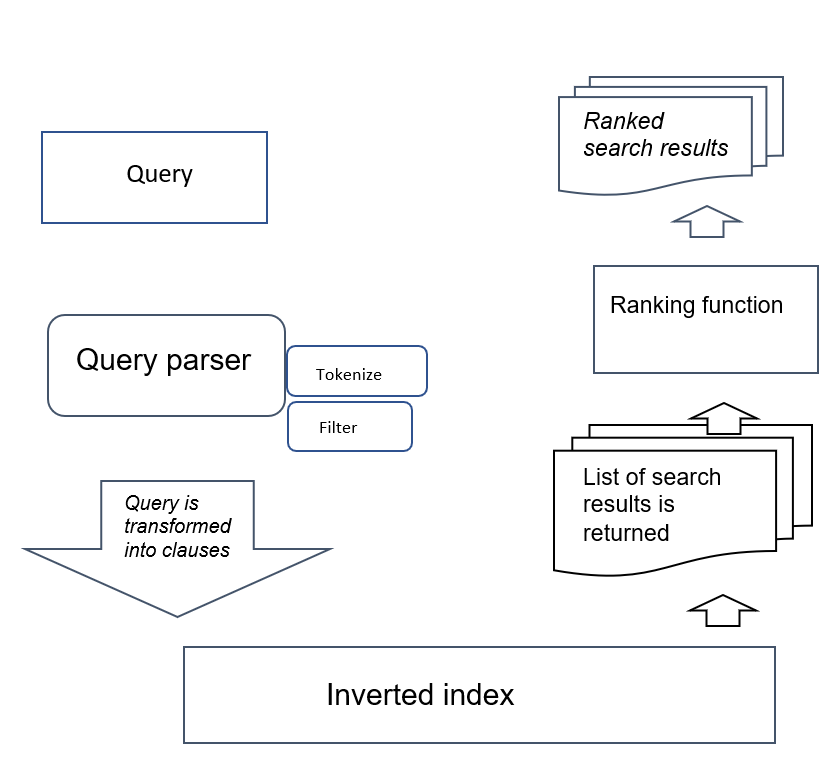
1. Using probabilistic retrieval framework “Okapi BM25” it is a ranking function used by search engine to rank matching document according to their relevance to given search query.

It is implemented in python using “rank-bm25” module.

1. Creating word embedding using CBOW or skip gram model and then converting these word embedding to document embedding by normal average or weighted sum average and then calculating the cosine similarity between document and query vector, and uses that as a document as document score.

**All steps:**

1. A query written by the user is parsed, analysed, and broken down into a set of of terms clauses.
2. The encoded query is exceuted against the search engine data structures (for each term, a lookup in the inverted index table is performed).
3. The matching document are collected and passed to the ranking function.
4. Each document is scored by the ranking function.
5. Typically, the list of search results is composed of such documents, sorted according to their score in the descending order.



**Overall work-flow**

More additional thing we can add in this search engine to make it more advance :

1. **Query autocompletion**: Suggest query based on first characters typed.

Elasticsearch: The Suggest API is one of the most developed APIs in Elasticsearch. It is extensively used in search solutions which tremendously improve user experience. Ranging from normal autocomplete to context based suggestions

1. **Using users history to rank document**: Record the past activity of a user and take it into consideration when ranking. For example, recurring terms in past queries may indicate a user’s interest in a certain topic, so search results on that same topics should have higher ranking.