

Tasty Search!

Build a search engine to search gourmet food reviews data and return the top K reviews that have the highest overlap with the input query. Requirements are as follows:

- Use the dataset available at <http://snap.stanford.edu/data/webFineFoods.html>. Use this [alternate download link](#) in case of issues.
- Randomly sample 100K reviews from the above set for the index. In case of memory issues, downsample to 50K.
- Build an api that returns the top K highest scoring documents for any given query, where the score is defined by the following:

$$\text{Score}(D, Q) = Q \cap D$$

(i.e. # tokens matching between Query(Q) & Document(D) normalized by query length the number of tokens in the given query).

e.g. given $Q = \{ \text{cat, processed, bad, good} \}$ and a document described by the following:

product/productId: B001E4KFG0

review/userId: A3SGXH7AUHU8GW

review/profileName: delmartian

review/helpfulness: 1/1

review/score: 5.0

review/time: 1303862400

review/summary: Good Quality Dog Food

review/text: I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.

$$\text{Score} = 2 / 4 = 0.5$$

- Queries are input as a set of tokens (words) and multiple occurrences of a given token in the text are counted as a single hit (i.e. you can ignore term frequency). You can assume queries are between 1 - 10 tokens. Ties between scores can be resolved using review/score for the document.

- Please note that it isn't necessary for all query tokens to exist in a given document for it to have a nonzero score. The score is computed based on the number of matching query tokens in a given document
- Document summary and text are to be used for computing score
- Set K to 20
- Expose a simple REST interface for search. A simple web form to perform search on the server itself would be great.

Your server can be inmemory only so don't worry about persistence etc. Also, please build your own rather than using part of frameworks like Lucene. Typically, your server should read in the review data and build its internal data structures to allow it to answer queries quickly thereafter.

Deliverables

- Link to Github repository with the code along with instructions on running it. Please also include the generated queryset.
- Bonus points for including a running instance on AWS!
- Please feel free to use any language you are comfortable with.