Pawsitive Retrieval

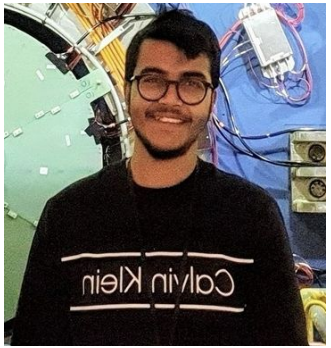
Retrieval of Relevant Responses from Reddit

Kristina Knowles

Marcos Ortiz

Sayantan Roy

Karthik Prabhu



Diptanil Roy

Objectives and Dataset

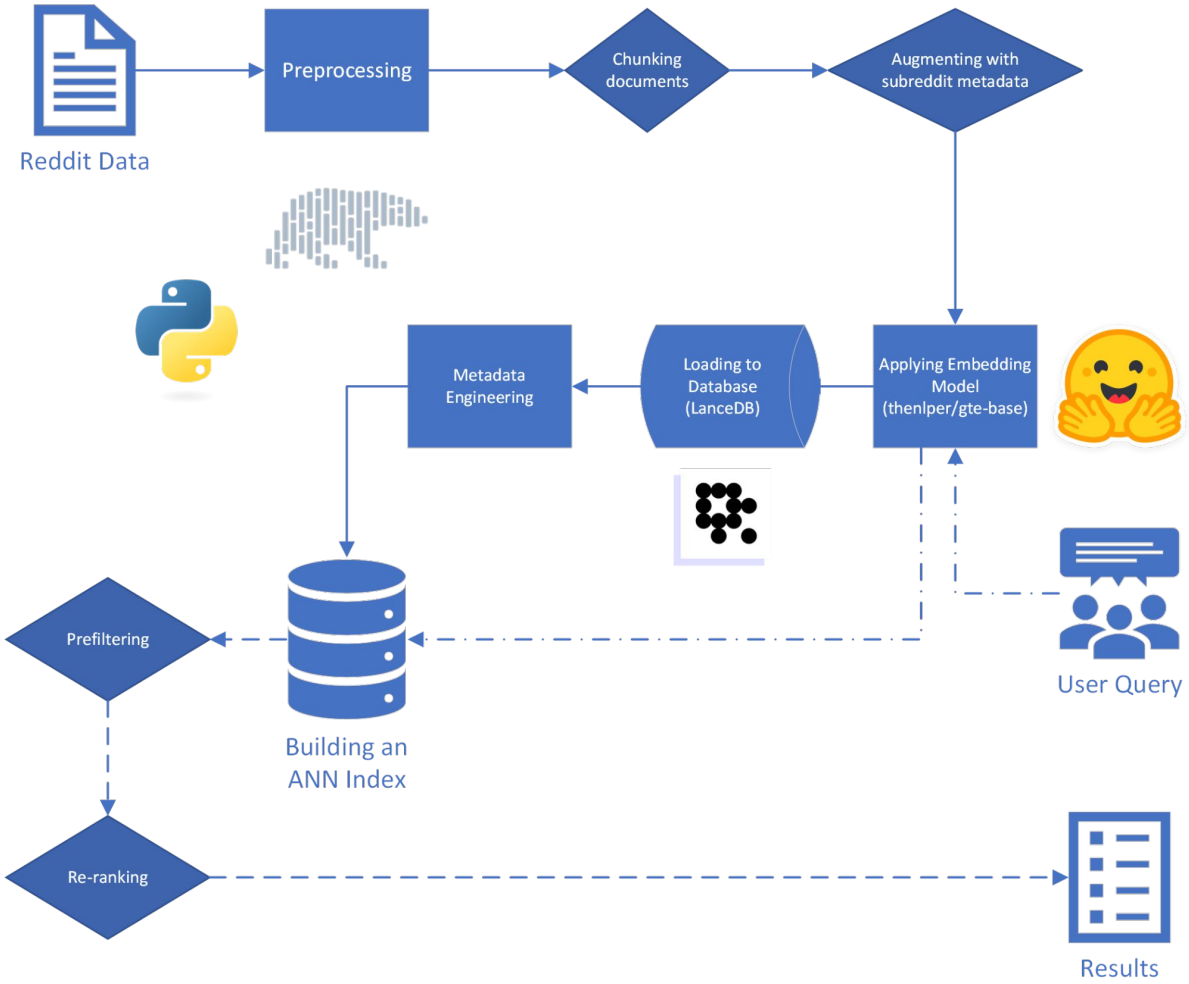
Key Objectives:

- Retrieve relevant responses from Reddit dataset, given arbitrary user query
- Develop a means of judging relevance to quantify improvements in relevancy of responses across model iterations
- Keep query times below 1 second

Dataset

- ~ 5.5 million posts from 34 different subreddits
- Training subset: 13 subreddits (~30% of dataset)
    - `[ 'KrakenSupport', 'Chase', 'TalesFromYourBank', 'GeneralMotors', 'wholefoods', 'fidelityinvestments', 'starbucksbaristas', 'GameStop', 'Fedexers', 'CVS', 'Lowes', 'UPSers', 'starbucks']`

Pipeline Overview



Processing

- Pre-Embedding:
  - Partition Data into Subreddits
  - Drop rows that don’t appear to have useful content
  - Replace `reddit_text` with `reddit_title` in submissions with empty `reddit_text`
  - Flag rows that appear to contain short questions (<100 characters, ends with “?”)
  - Add `reply_ids` to each row (a list of the comments that are replies to this one)
  - Add `subreddit_name` as metadata to each row that have useful content
  - Chunk `reddit_text` with model specific chunk size

- Post-Embedding:
  - Add “sentiment of replies” metadata
  - Add “agree (and disagree) distance of replies” metadata

Embedding

General Text Embedding Model (thenlper/gte-base) chosen because:

- Manageable size: 0.22GB, 109M parameters
- Suitability to our task:
  - trained on similar data
  - allows embedding of 512 token sequences

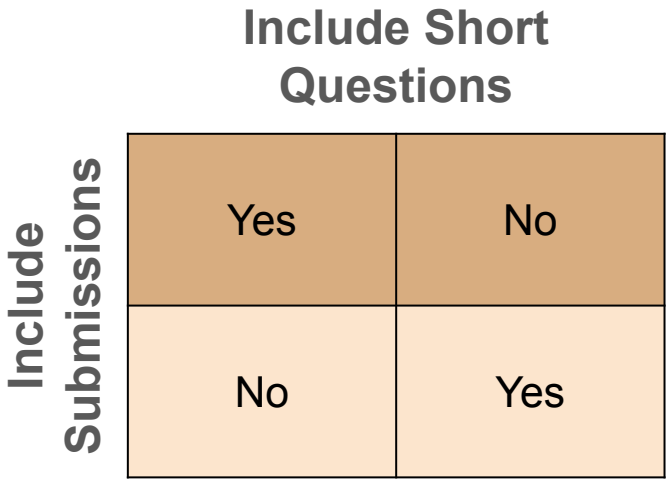
Parameters we varied:

- Chunk sizes: 512, 256, 128*
- Attaching metadata before embedding

Meta-Data	Chunk Size: 512	256	—
Yes	512	256	—
No	512	256	128

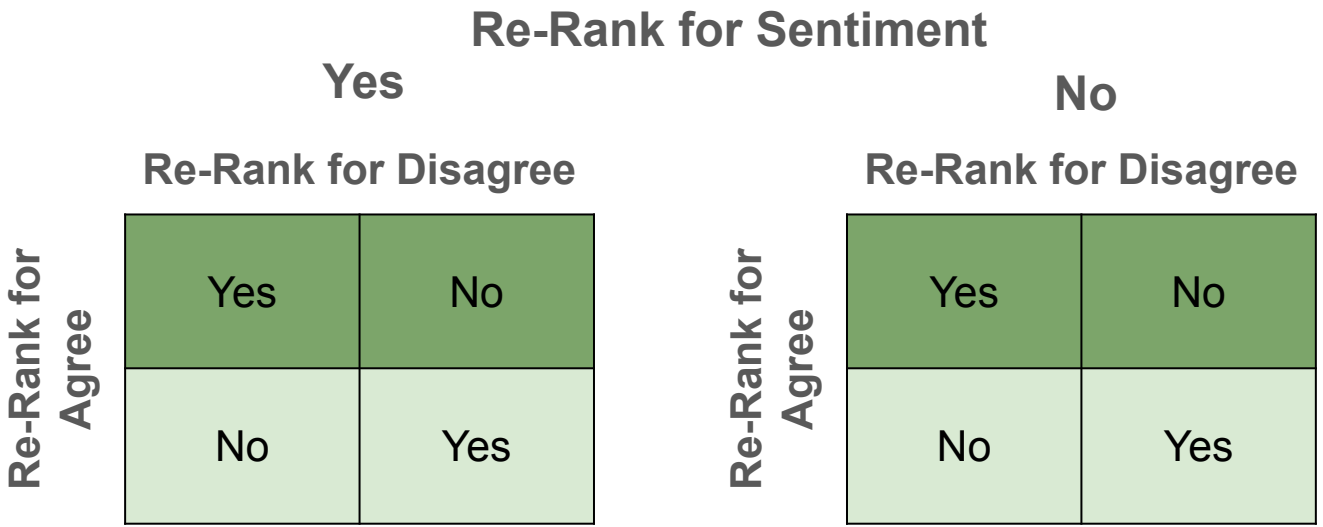
Vector Database, Indexing and Prefiltering

- LanceDB:
  - Open source vector database
  - Integrations with Python and Polars
  - No server needed for our application
- Almost Nearest Neighbors (ANN) Index:
  - Composite inverted file index (IVF) with product quantization (PQ)
- Prefiltering:
  - SQL queries narrow DB focus before retrieval:
    - Filter by post type (`submissions`)
    - Filter by engineered metadata (`is_short_question`)



Re-Ranking

- Reply sentiment
  - The sentiment of each text was calculated
  - Sentiments of replies were calculated and summed (1:positive, 0:neutral, -1:negative)
  - Results with positive reply sentiments sums were moved up
  - Results with negative sums were moved down
- Replies agree (disagree) distance
  - Each text was given an “agree distance” by computing its average distance from a set of “agree statements”
  - The average of these distances was calculated across replies
  - Results with lower agree distances were moved up
  - Results with lower disagree distances (computed similarly) were moved down.

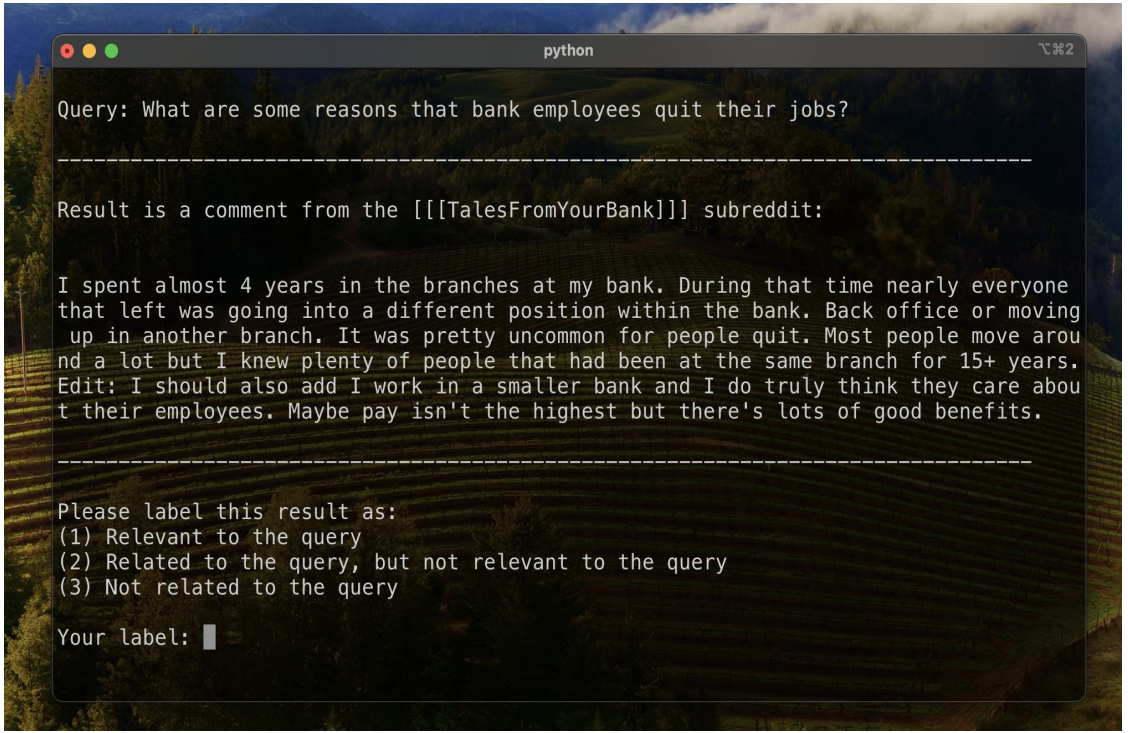


Evaluating Model Performance: Labeling Data

We set the ``reference set`` of replies to the standard queries by retrieving results from training set (in baseline configuration)

We manually labeled a total of ~1000 query-result pairs, across 26 queries, as “relevant”, “related but not relevant”, and “not related”.

The final label for each pair was determined by the label that receive the most “votes” across 3-5 human labelers.



Evaluating Model Performance: Scoring Results

- Metrics for tracking performance:
  - Mean Reciprocal Rank
    - How high up in the results does the ***first relevant result*** appear?
  - Extended Reciprocal Rank
    - Given that we have several known relevant results, how close do we come to having ***all known relevant results*** appear in at the top?
  - Normalized Discounted Cumulative Gains
    - Without regard for the total number of known relevant results, are ***relevant results*** appearing near the top?

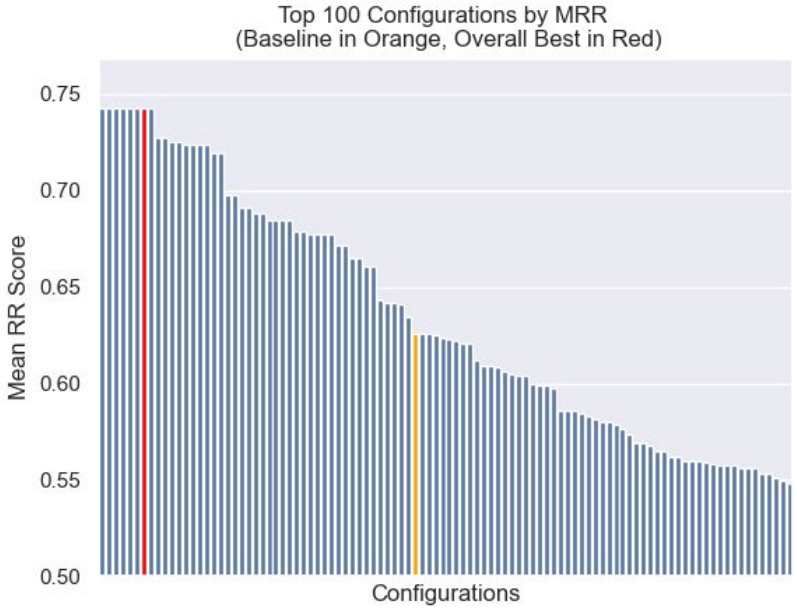
Results: Best Overall Configuration

We tested all of our configuration on the entire dataset.

The configuration that achieved the highest average rank across all three of our metrics was:

- Chunk size of 512
- Metadata attached before embedding
- Pre-filtering out “short questions” before querying the data
- Re-ranking the results by reply sentiment, then
- Re-ranking by reply "agree distance"

Results for queries using this configuration can be retrieved in **~300-450ms**

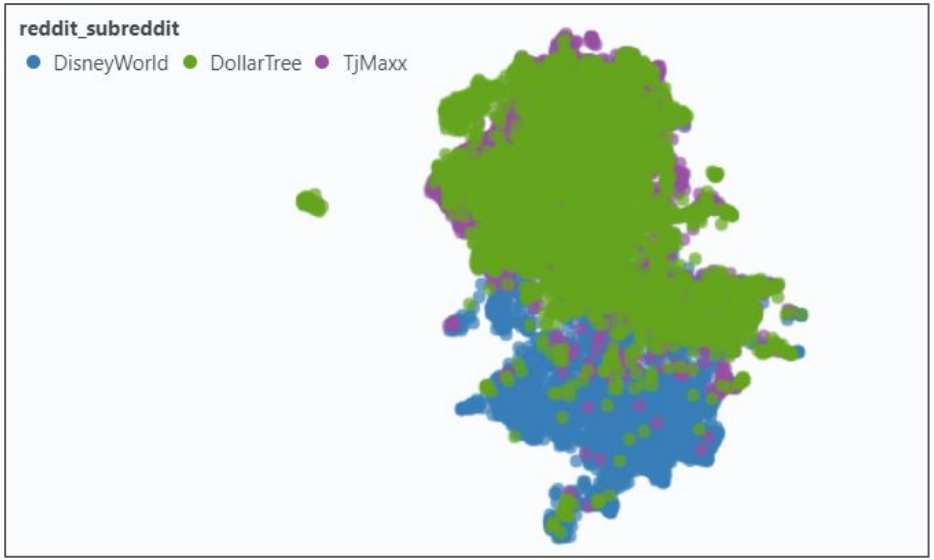


Results: Parameters with the Highest Impact

Chunk Size

Results: Parameters with the Highest Impact

Metadata



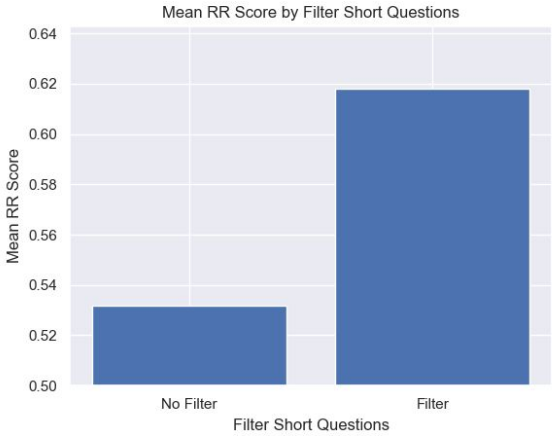
Without Metadata

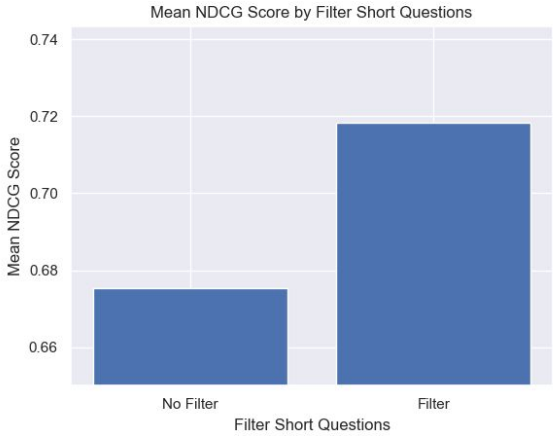
With Metadata

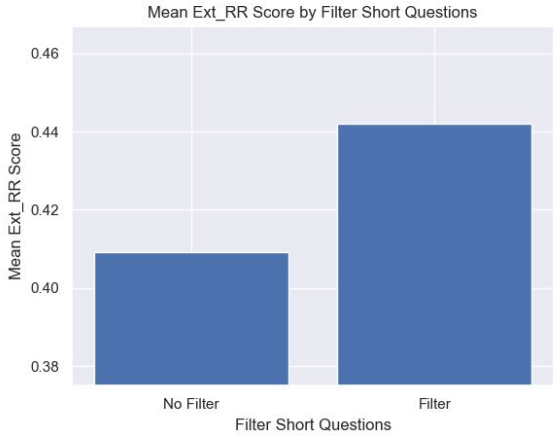
Created using [renumics-spotlight]

Results: Parameters with the Highest Impact

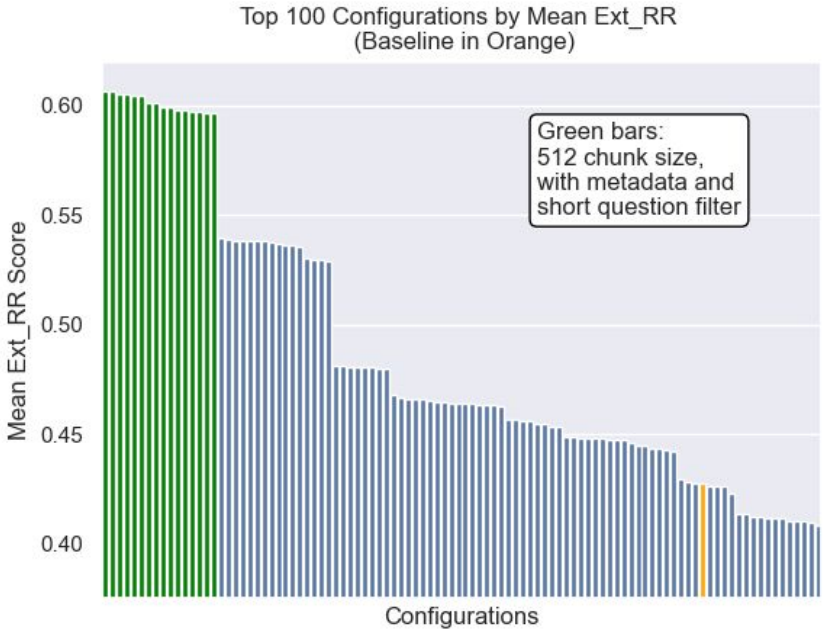
Filtering Short Questions

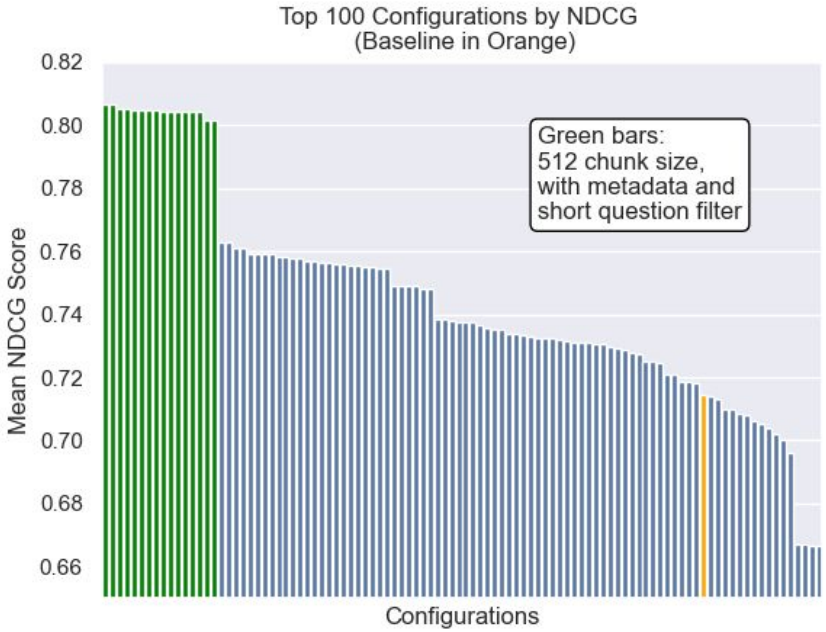






Results: Chunk Size and Filtering Combined





Future Work

- Preprocessing+Labeling:
  - Handle emojis, images and common abbreviations
  - Improved manual and automated labeling

- Optimizing embedding models:
  - Tune LanceDB parameters to see if retrieval times can be improved further
  - More aggressive filtering
  - Experiment with distance metrics
  - Other embedding and transformer models

- Reranking strategies:
  - Apply new criteria for re-ranking that reflect user-intent
  - Experiment with re-ranking methodologies

Thank you!

Dall-E