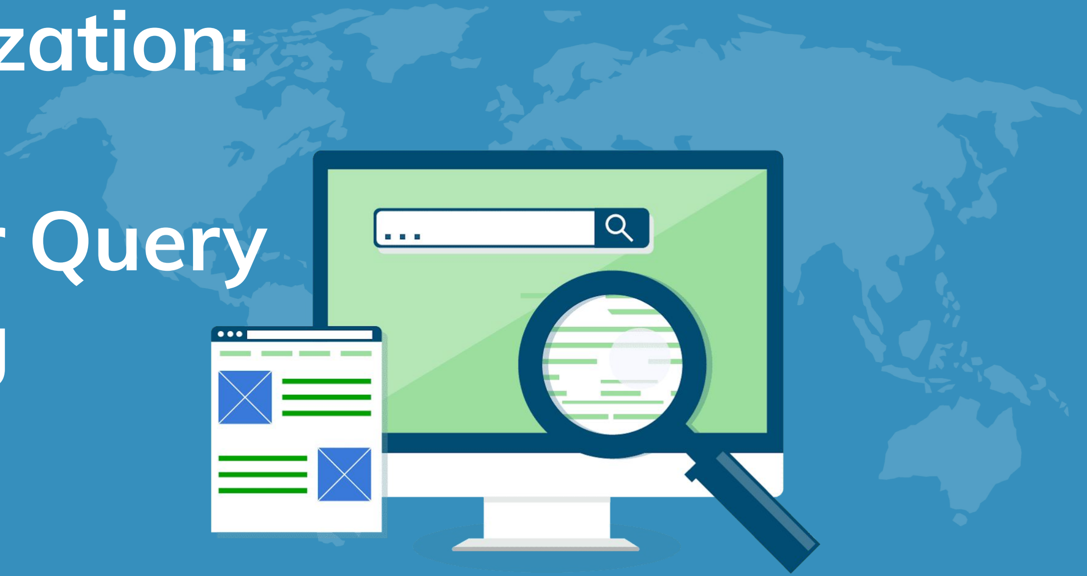
OpenSource Connections

Dynamic Hybrid Search Optimization: A Practical Framework for Query Understanding



BASED Meetup
March 20, 2025
Daniel Wrigley

Lexical Search

🔍 fruit

id: 1
Text: … a **fruit**
basket that
contained apples
and oranges …

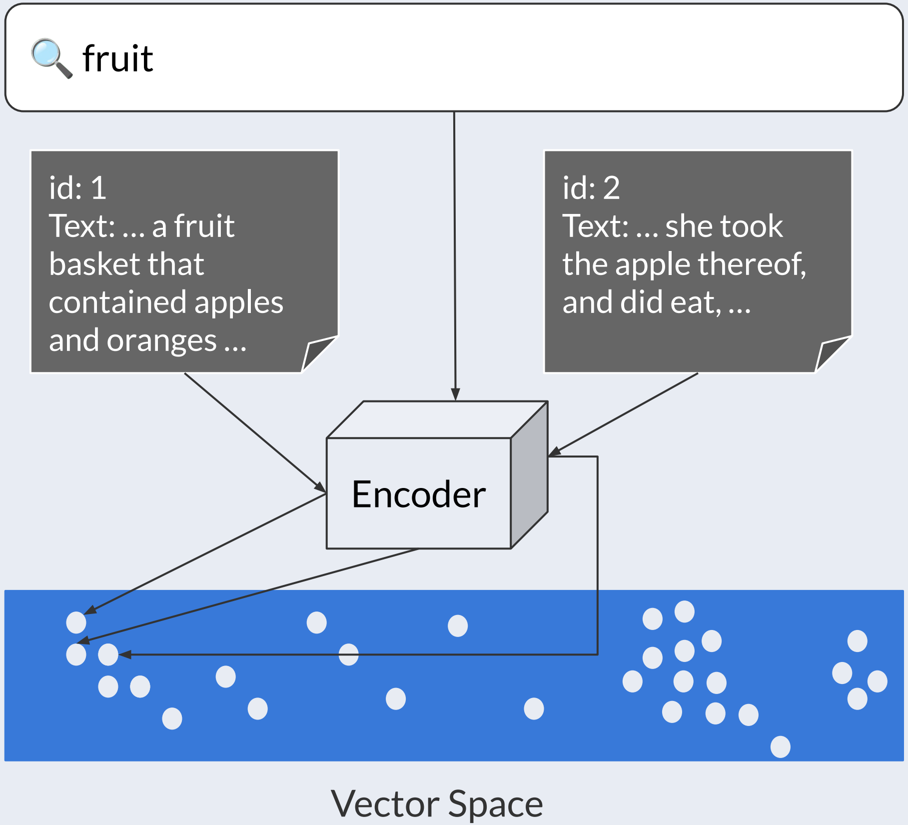
id: 2
Text: … she took
the apple thereof,
and did eat, …

✅
It's a
match

🚫
It's not a
match

- 👍 Precise keyword-based matching
- 👍 Fast and efficient retrieval with inverted index structure
- 👍 Works well for structured & high-precision queries
- 👎 Needs careful configuration of text analysis (tokenizers, stemmers, synonyms etc.) to handle advanced matching
- 👎 Struggles with long-tail or ambiguous queries

Vector Search



- 👍 Captures semantic meaning, handling synonyms & related concepts
- 👍 More effective for long, natural language queries

- 👎 Computationally expensive
- 👎 Can retrieve non-relevant documents: **always** shows the *k nearest neighbours* - no matter how far away

Why Hybrid Search?

Lexical Search + Vector Search = 💙

Hybrid search **combines lexical and vector search** to improve relevance:

✅ Lexical search ensures precision for term-based matches.

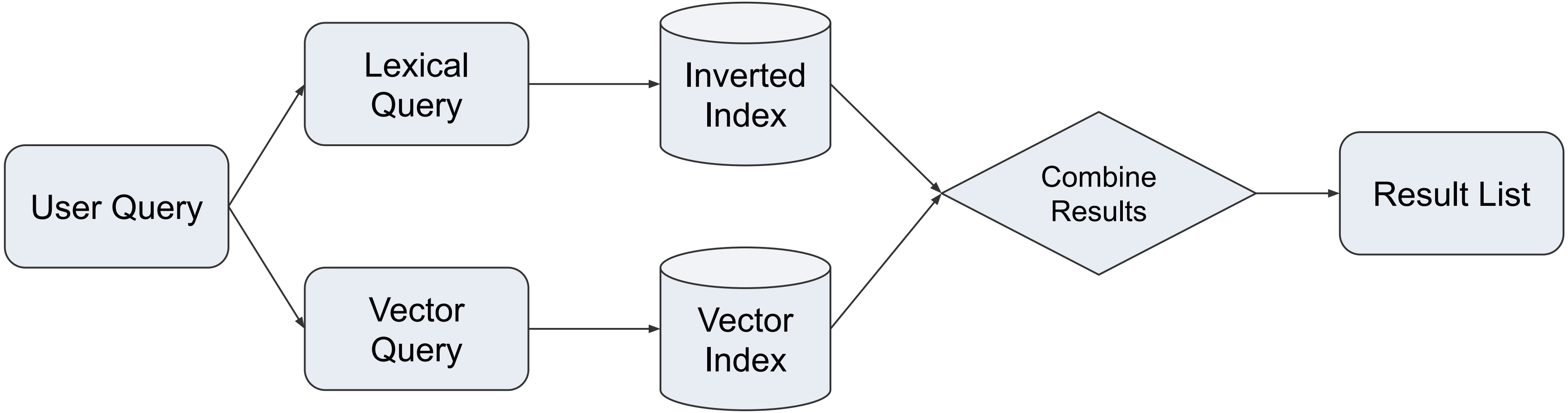
✅ Vector search enhances recall by capturing semantic meaning.

Example Query: *"How to improve neural networks?"*

- Lexical search finds exact keyword matches, but might miss semantically related terms like *"deep learning optimization"*.
- Vector search captures related terms but might lack specificity.

Hybrid search balances both.

Hybrid Search - An Illustration


User Query

Lexical Query

Vector Query

Inverted Index

Vector Index

Combine Results

Result List

Inverted Index and Vector Index can be part of the same search platform

Basic Hybrid Search Techniques

How to best combine the results of the sub-queries? 🤔

1️⃣**Linear Combination**

- (w_1 × normalized BM25 score) + (w_2 × normalized dense vector similarity)
- **Pros:** Simple, tunable weight parameters
- **Cons:** Needs score normalization & hyperparameter tuning (w_1 & w_2)

Our experimentation focus

2️⃣**Reciprocal Rank Fusion (RRF)**

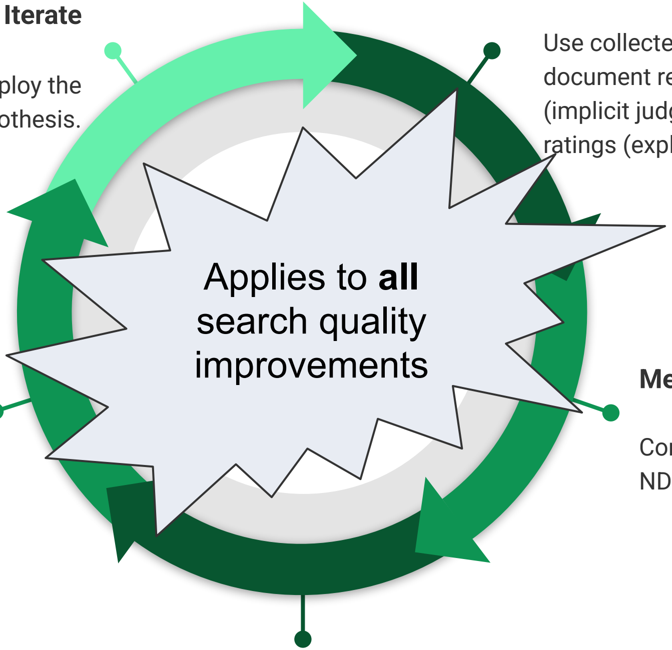
- Ranks from BM25 & vector search are merged: $RRFscore(d \epsilon D) = \sum_{r \epsilon R} \frac{1}{k + r(d)}$
- **Pros:** No score normalization required
- **Cons:** Less flexibility compared to linear combination

Reciprocal Rank Fusion paper

Why Tune Hybrid Search?

How do you **combine different search techniques** (ingredients) effectively for **improved findability** (tastier recipe)? How do you know **which parameters are the best** parameters for hybrid search?

Search Result Quality Improvement Cycle



Applies to **all** search quality improvements

Gather Relevance Judgments

Use collected user signals to infer document relevance for each query (implicit judgments) or gather human ratings (explicit judgements).

Measure Search Quality

Compute search quality metrics (e.g., NDCG, MRR, Recall, Precision).

Generate Hypotheses for Improvement

Identify weaknesses in ranking, retrieval, or query understanding.

Experiment & Validate Improvements

Measure search quality again using the same metrics to determine improvement.

Evaluate & Iterate

If search quality improves, deploy the change; otherwise, refine the hypothesis.

Experiment Hypotheses

1) By identifying the best parameter set for hybrid search we can outperform the baseline search quality metrics → global hybrid search optimization
2) By dynamically predicting the best parameter set per query we can outperform the search quality metrics for identified best global hybrid search parameter set → dynamic hybrid search optimization

Experiment setting:

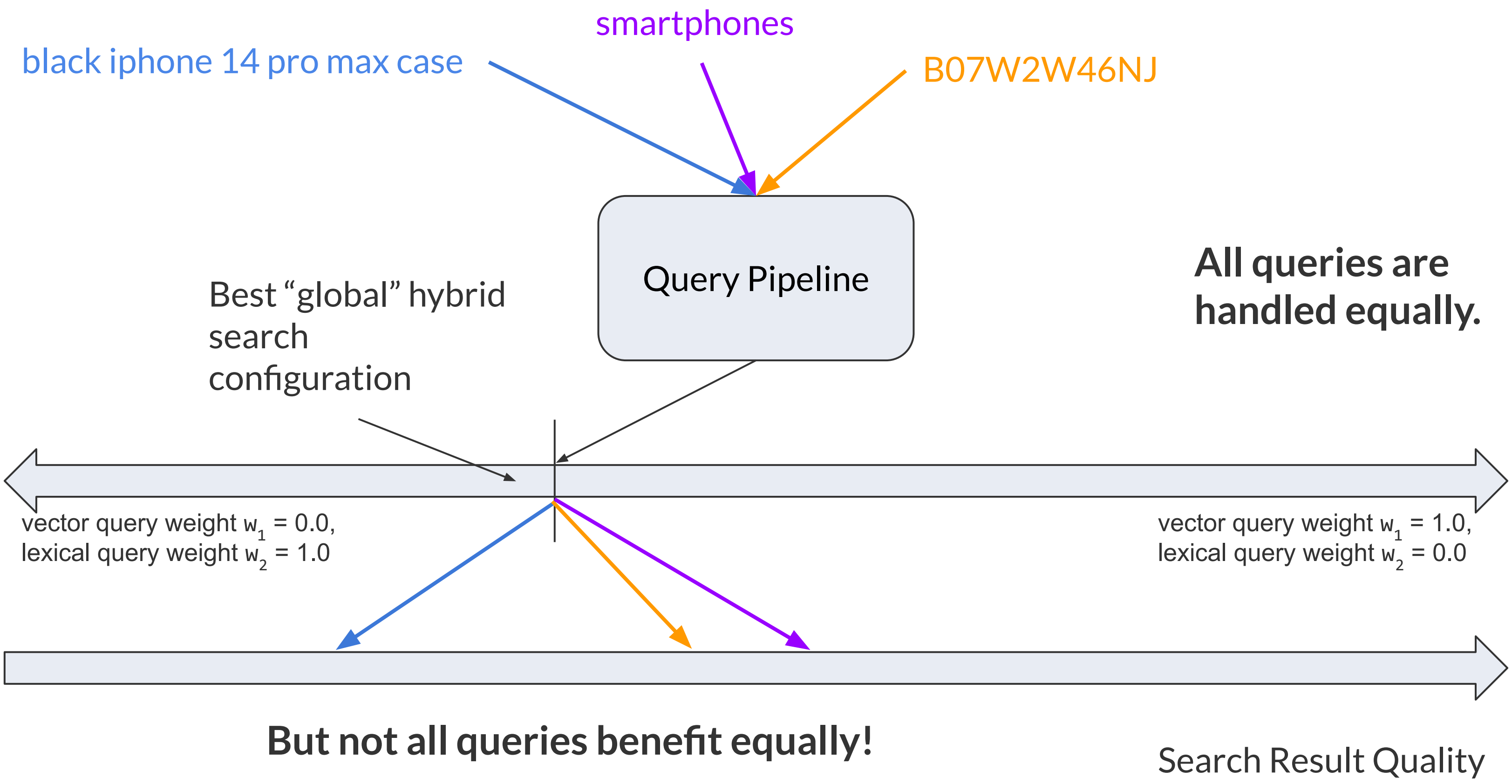
- ESCI dataset
- 5,000 randomly sampled queries with judgments
- Lexical search baseline
- OpenSearch hybrid search query: arithmetic combination of lexical and neural search

Global Optimization Strategy – Grid Search

Systematically **test different parameter values**

🎯 **Best Setting: ?**

Parameters	**DCG@10**	**NDCG@10**	**Precision@10**
vector query weight w_1 = 0.0, lexical query weight w_2 = 1.0	?	?	?
vector query weight w_1 = 0.1, lexical query weight w_2 = 0.9	?	?	?
vector query weight w_1 = 0.2, lexical query weight w_2 = 0.8	?	?	?
…	?	?	?
vector query weight w_1 = 1.0, lexical query weight w_2 = 0.0	?	?	?



smartphones

black iphone 14 pro max case

B07W2W46NJ

Query Pipeline

Best “global” hybrid search configuration

All queries are handled equally.

vector query weight w_1 = 0.0,
lexical query weight w_2 = 1.0

vector query weight w_1 = 1.0,
lexical query weight w_2 = 0.0

But not all queries benefit equally!

Search Result Quality

Feature Engineering for ML-Based Search Optimization

Feature groups and features

We divide the features into **three groups**: query features, lexical search result features, and neural search result features:

- **Query features**: These features describe the user query string.
- **Lexical search result features**: These features describe the results that the user query retrieves when executed as a lexical search.
- **Neural search result features**: These features describe the results that the user query retrieves when executed as a neural search.
- **Additional feature**: the weight of the vector search query (w_1) in our hybrid search setup

ML Model Training Data

Per query: the vector search weight w_1 that maximizes NDCG together with its features

NDCG	Vector search weight w_1	Number of query terms	Query length	Contains numbers	Contains special chars	Number of keyword search results	Max title score	Sum of title scores	Max vector search score	Avg vector search score
0.54	0.0	4	22	1	1	14	0.19	1.42	0.48	0.47
0.23	1.0	5	26	1	1	3	0.22	0.41	0.60	0.59
…										

Target

What we really want to predict

Query Features

Keyword Search Result Features

Vector Search Result Features

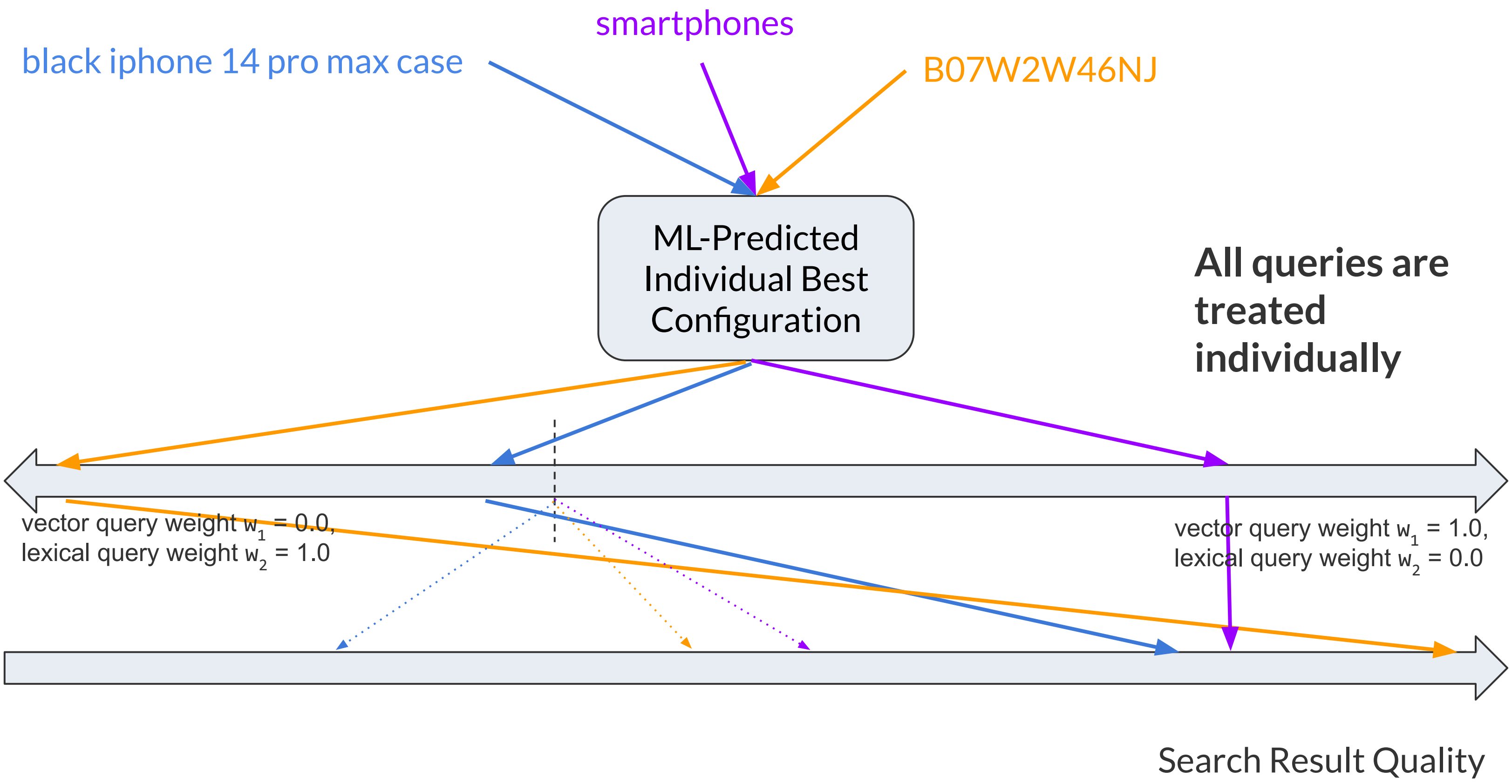
ML Model Evaluation

Evaluation Approach

- Linear Regression & Random Forest Regression Model
- Cross-Validation (5 splits, 80/20 train/test size)
- All Feature Combinations
- Regularization

Evaluation Results

- Best RMSE Linear Regression: 0.23
- Best RMSE Random Forest: 0.18
- Best feature combinations were always features from all groups (query, keyword search result & vector search result)



black iphone 14 pro max case

smartphones

B07W2W46NJ

ML-Predicted Individual Best Configuration

All queries are treated individually

vector query weight w_1 = 0.0,
lexical query weight w_2 = 1.0

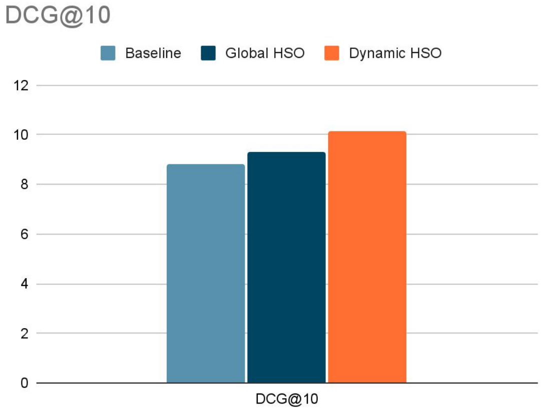
vector query weight w_1 = 1.0,
lexical query weight w_2 = 0.0

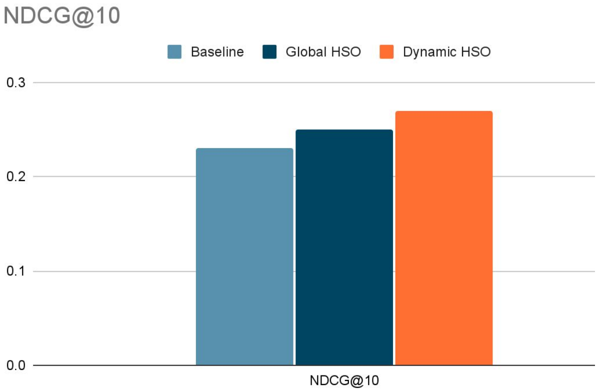
Search Result Quality

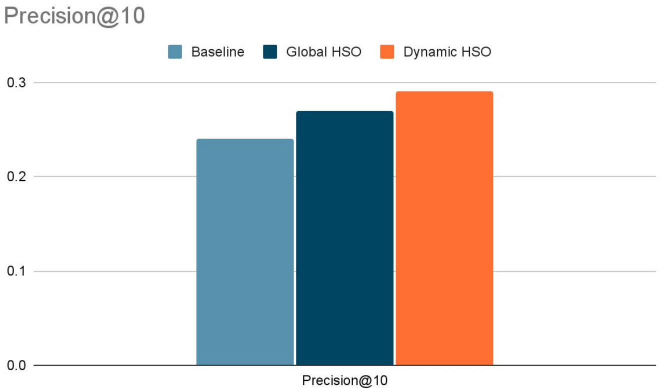
Experiment Results

Metrics improved when applying the static approach of the global hybrid search optimizer and yet again moving to the dynamic approach:

- DCG improved by 8.9% (from 9.3 at the global HSO to 10.13 at the dynamic HSO).
- NDCG improved by 8.0% (from 0.25 at the global HSO to 0.27 at the dynamic HSO).
- Precision improved by 7.4% (from 0.27 to 0.29 at the dynamic HSO)







Production Considerations

- Do thorough offline testing to identify the best candidates for online experimentation
- Run online experiments (A/B tests)
- Explore different feature options for your scenario
  - Presented features may not be suitable for your search platform
  - Engineering search result features may not be feasible in low-latency search platforms
- Identify low-hanging fruit opportunities first
  - Come up with heuristics that let you confidently bypass any complex queries. Examples:
    - Queries for IDs should always be keyword queries
    - Queries like *return policy* in an online-shop should be redirects to customer support pages
    - Your head queries most likely benefit from manually curated rules rather than ML-driven processes
  - Experiment on baseline optimization: there are a lot of parameters to tune even without hybrid search!

Resources

- OpenSearch Blog “Optimizing Hybrid Search”
- OpenSearch Issues Enabling Native Usage Within OpenSearch:
    - https://github.com/opensearch-project/neural-search/issues/1172
    - https://github.com/opensearch-project/neural-search/issues/1005
- Hybrid Search Optimizer Repository
- Optimizing Hybrid Search OpenSearch Blog Post
- Search Quality Evaluation App Repository

Thank you!
Questions?

www.opensourceconnections.com