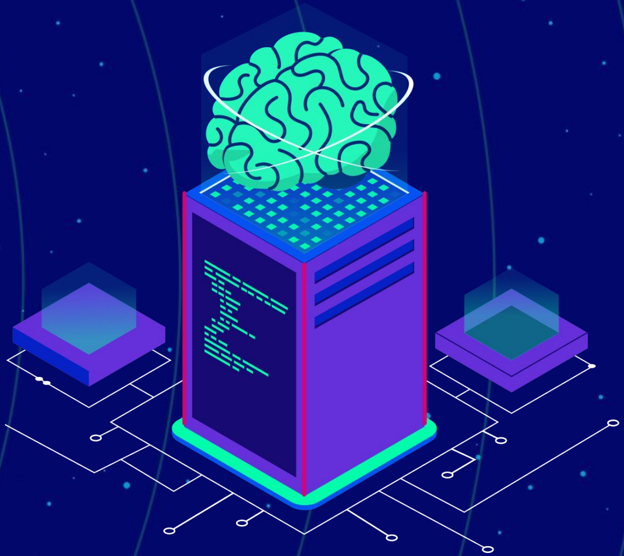
CIKM MMSR 2024

marqo

Designing Multimodal Search Interfaces

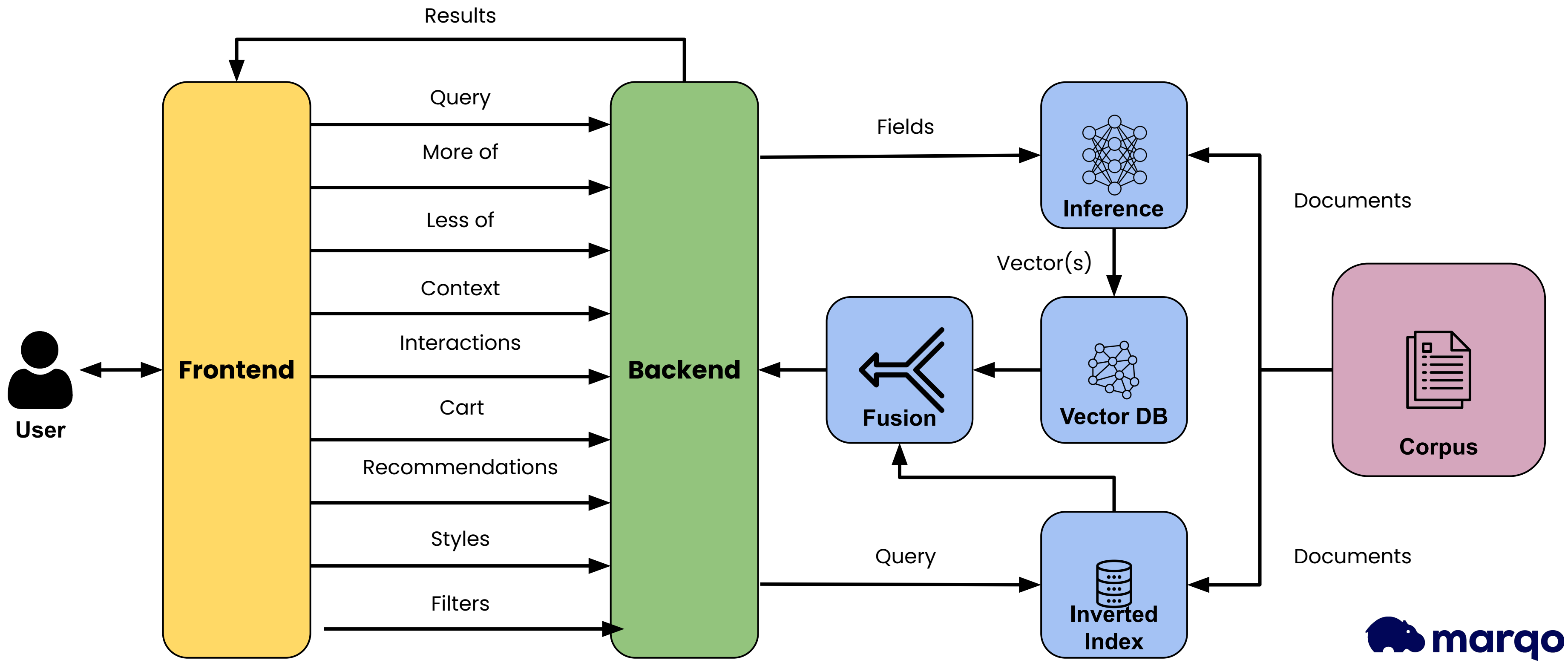
Tom Hamer (CEO, Marqo)
Owen Elliott (Solutions Architect, Marqo)
Jesse Clark (CTO, Marqo)

https://marqo.ai/



Overview

In this work we present implementations and example user interfaces for multimodal search applications, leveraging unique properties of dense retrieval models to build more effective ways for users to express their information needs

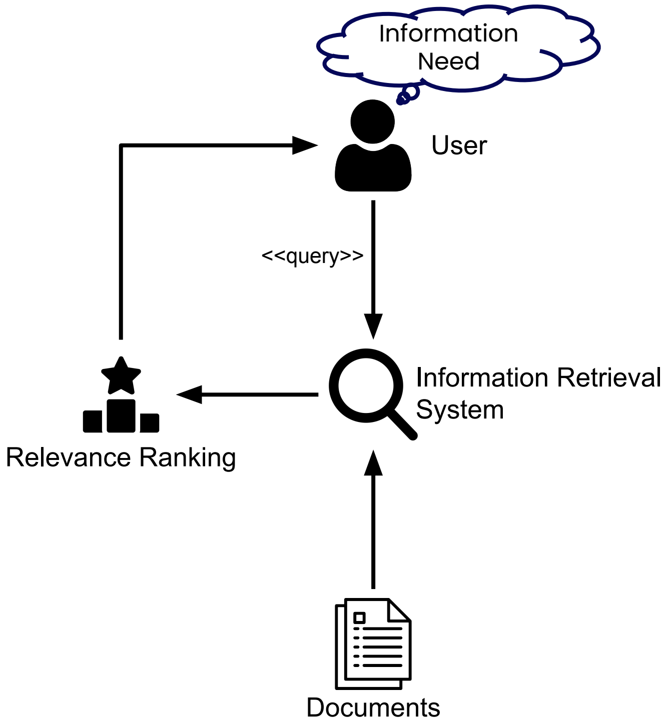


marqo

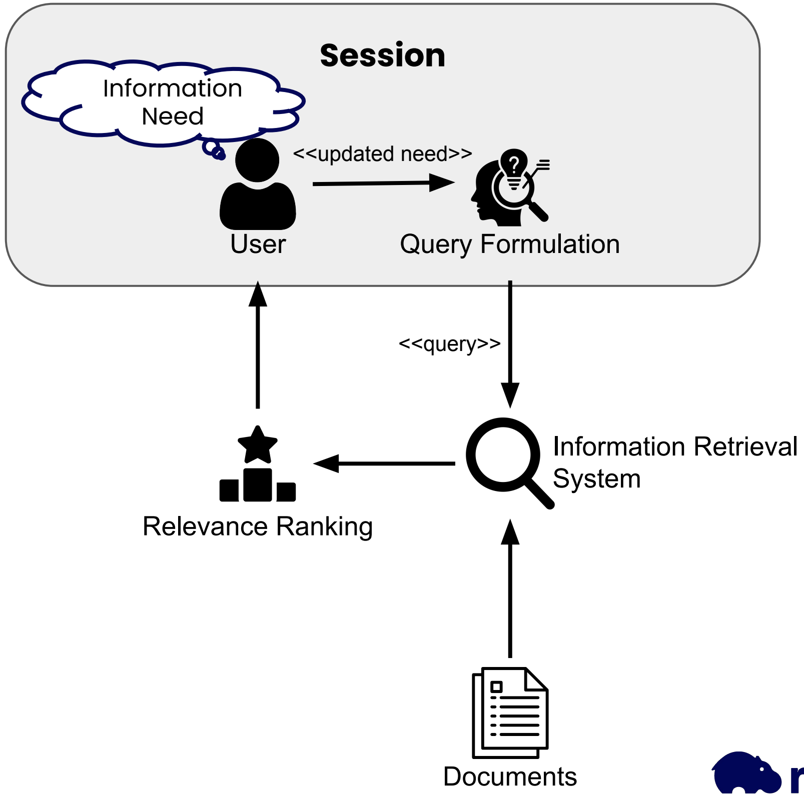
Information Retrieval

Simple models don't capture our session based reality

Simple model of Information Retrieval



Iterative query refinement in a search session



marqo

Dense multimodal retrieval vs lexical retrieval

We specifically look at dense multimodal retrieval systems, which differ from traditional lexical retrieval systems

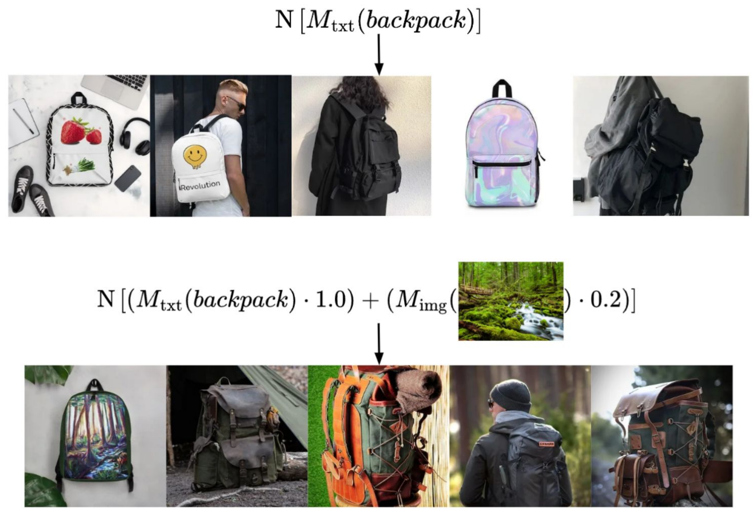
Dense retrieval enables flexible expression of information needs

Dense Multimodal Retrieval:

- Dense retrieval systems encode images and text into continuous vector spaces.
- Multiple vectors can be interpolated to make new vectors.

Lexical Retrieval:

- Lexical retrieval systems (e.g., TF-IDF, BM25) rely on sparse token-based representations.
- You can't interpolate or combine sparse lexical vectors meaningfully - each token is discrete and context-independent.


$\mathrm{N}\left[M_{\mathrm{txt}}(backpack)\right]$
$\mathrm{N}\left[(M_{\mathrm{txt}}(backpack)\cdot 1.0) + (M_{\mathrm{img}}(\;)\cdot 0.2)\right]$
Revolution

Refining queries with additional pieces of multimodal information

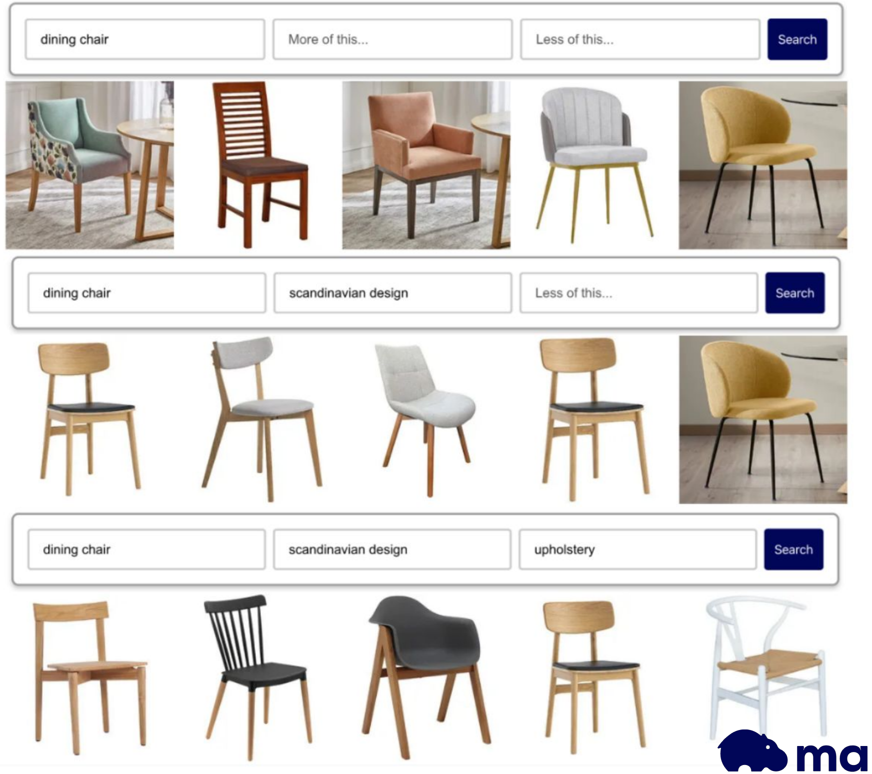
Query refinement via composite queries

Query refinement plays into the session based nature of most search applications where users can iterate on their search or evolve their information needs

Enabling users to express their information need

- Classical information retrieval depends on the users ability to express their information need.
- Dense retrieval offers new mechanisms for query design.
- Interpolating positive and negatively weighted query terms can be presented in a user friendly way.

(dining chair * 1.0) + (scandinavian design * 0.6) + (upholstery * -1.1)

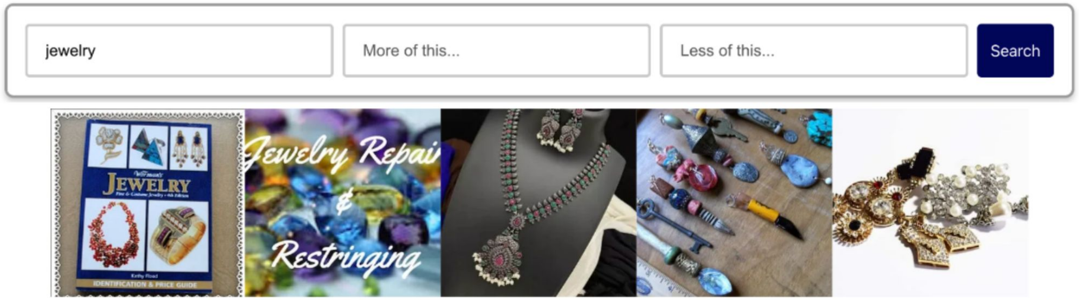


Influencing quality

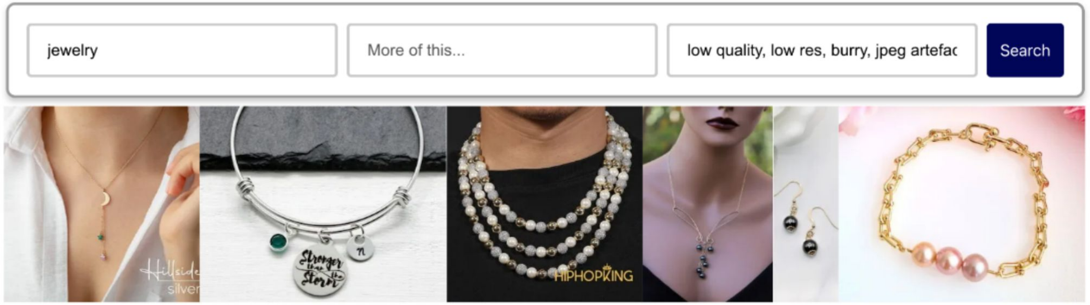
Negatively weighted query terms can be used to steer search away from undesirable traits, this is especially powerful when searching user generated content of dubious quality

Use with user generated content

- Dubious quality of items
- No metadata to ascertain “quality” from
- Can be tuned post indexing
- Can be optimised to be query dependant
- Aligns content surfaced in search with brand



jewelry | More of this... | Less of this... | Search



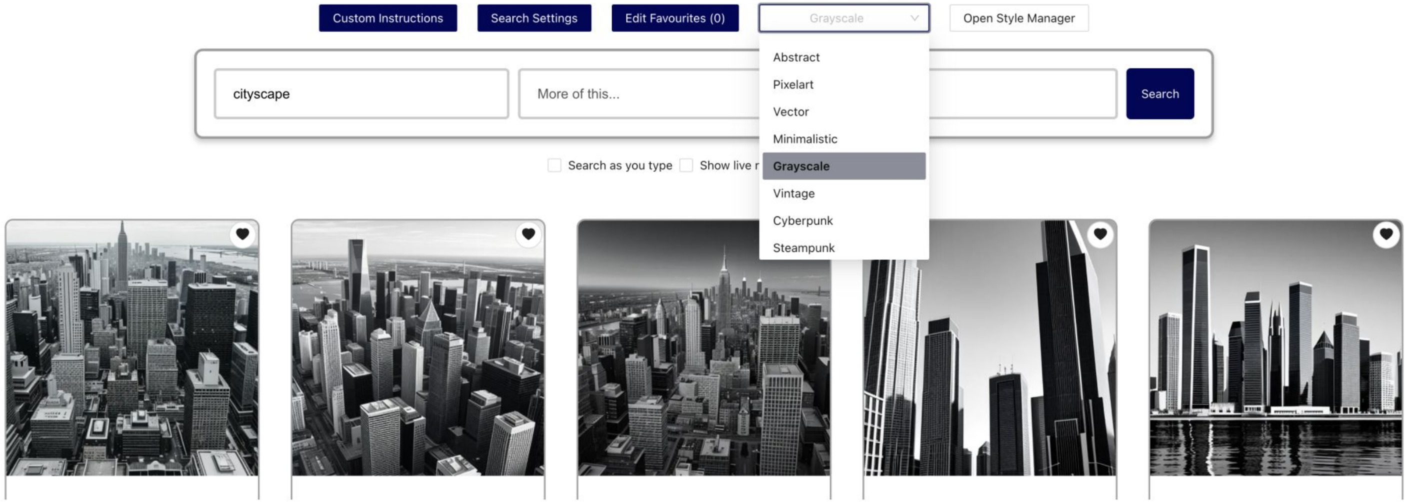
jewelry | More of this... | low quality, low res, burry, jpeg artefac | Search

Implementing search with CLIP models

CLIP models exhibit specific characteristics which can be utilised to better represent a users information intent

Taking inspiration from zero-shot tasks

- The nature of training data can be exploit to guide behaviour.
- Similar to prompt engineering with LLMs.
- Prompts can be placed around queries.
- Exposed in UI as any traditional selector element.



Semantic filtering

Leveraging the understanding of CLIP models and the caption style nature of their training data we can prompt to create semantic filters

- Intuitive to create
- LLMs can create and refine them effectively
- Interpretable
- Can be generated by non-expert users

```
FASHION_STYLE_MAPPING = {
   "streetwear": "An image showcasing a <QUERY> in a streetwear style, perfect for urban fashion",
   "high_fashion": "A high-fashion, editorial image of a <QUERY>, emphasizing luxury and trend-setting designs",
   "comfy": "A cozy, comfortable-looking image of a <QUERY>, ideal for casual wear",
   "summer_fashion": "A summer-inspired image of a <QUERY>, light and perfect for warm weather",
   "winter_fashion": "A winter-themed image of a <QUERY>, highlighting warmth and layering",
   "vintage_fashion": "A retro-styled image of a <QUERY>, with a vintage or nostalgic flair",
   "runway": "A runway image of a <QUERY>, capturing the essence of high fashion shows",
   "bohemian": "An image of a <QUERY> with a bohemian style, emphasizing free-spirited aesthetics",
   "sporty": "A sporty, athletic image of a <QUERY>, suitable for active lifestyles",
   "elegant": "An elegantly styled image of a <QUERY>, perfect for formal occasions",
   "casual": "A casual, everyday image of a <QUERY>, for a relaxed look",
   "avant_garde": "An avant-garde, boundary-pushing image of a <QUERY>, for cutting-edge styles",
   "vintage": "A vintage-inspired image of a <QUERY>, with a nod to past fashion trends",
   "retro": "A retro-styled image of a <QUERY>, featuring elements from past decades",
   "minimalist": "A minimalist image of a <QUERY>, focusing on simplicity and clean lines",
}
```

Diversified Recommendations

The challenge with recommendations when formulated in the lens of search is diversification, wider traversal of the vector space via random walks or additional mechanisms are required

We need to explore further into the neighbourhood

- Random walks
- Breadth first random walk progressively reaching further into the neighbourhood
- Retains continuity, children are connected to parents
- Stays relevant without showing many identical items

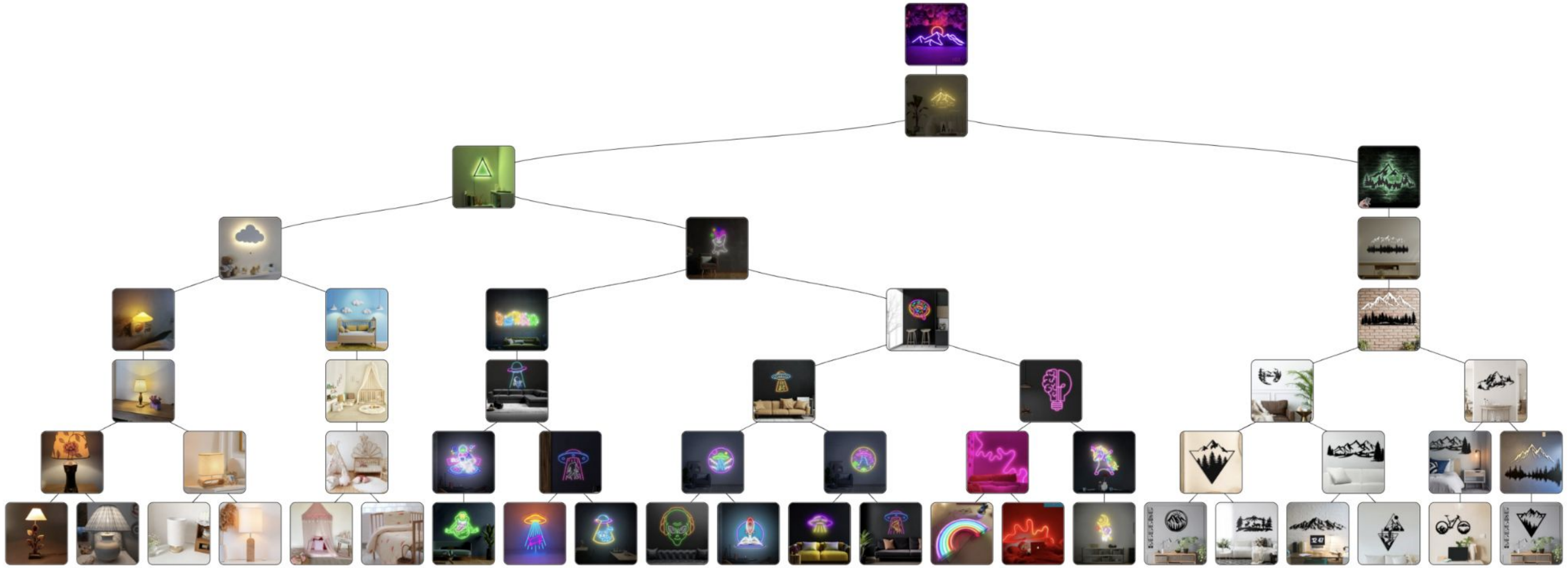
Algorithm 2 Generate Recommendation Tree with a Random Walk

Require: $\mathbf{v} \in \mathbb{R}^d$, L: number of layers, C: maximum children per node, k: nearest neighbours
Ensure: $root$: Tree structure with children up to L layers deep

```
1:  Initialize $root$ with $(\mathbf{v}, \{\})$ as the vector and an empty list for children
2:  Initialize $visited$ set with $\{\mathbf{v}\}$
3:  Initialize $currentFront$ as a queue containing $root$
4:  for $\ell = 1$ to $L - 1$ do
5:      Initialize $nextFront$ as an empty queue
6:      while $currentFront \neq \emptyset$ do
7:          Dequeue $currentItem$ from $currentFront$
8:          $children \leftarrow$ GetLayer$(currentItem, C, visited, k)$
9:          for each $child \in children$ do
10:             Enqueue child into $nextFront$
11:         end for
12:     end while
13:     $currentFront \leftarrow nextFront$
14: end for
15: return $root$
```

Diversified Recommendations

The challenge with recommendations when formulated in the lens of search is diversification, wider traversal of the vector space via random walks or additional mechanisms are required

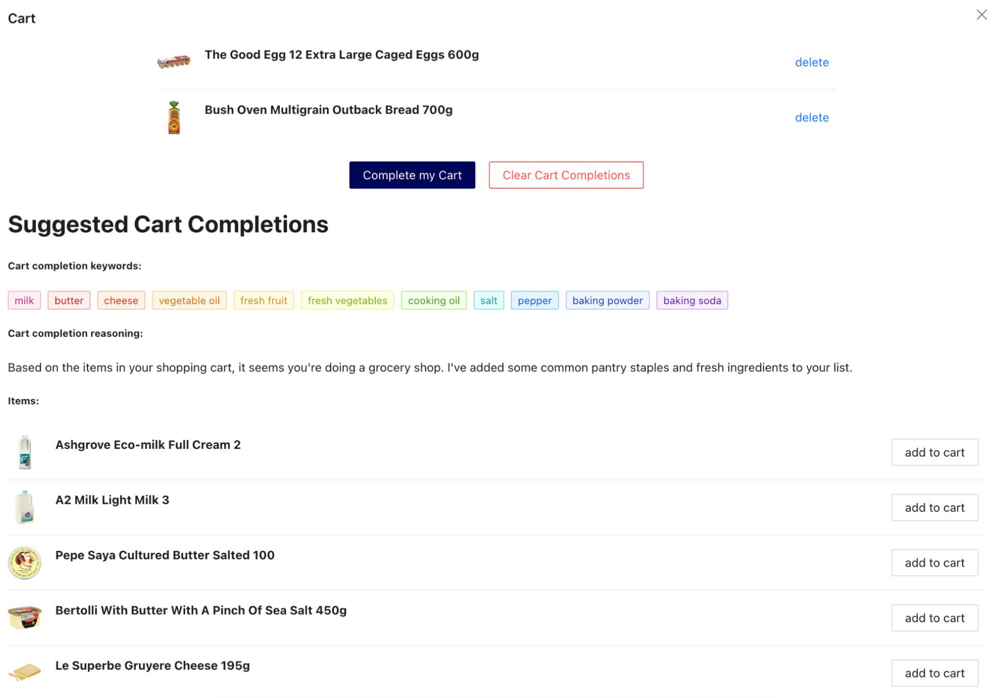


Complementary recommendations

We can leverage the world model of LLMs to understand a users intent a make an educated guess at what they might wish to also find (based on interactions or cart)

Complementary Items

1. Use a set of items as context
2. Generate queries for complementary items
3. Slerp vectors from interacted or selected items as context (with a low weight) with the generated query vectors
4. Execute one or more searches



Cart

The Good Egg 12 Extra Large Caged Eggs 600g — delete

Bush Oven Multigrain Outback Bread 700g — delete

Suggested Cart Completions

Cart completion keywords:

milk | butter | cheese | vegetable oil | fresh fruit | fresh vegetables | cooking oil | salt | pepper | baking powder | baking soda

Cart completion reasoning:

Based on the items in your shopping cart, it seems you're doing a grocery shop. I've added some common pantry staples and fresh ingredients to your list.

Items:

- Ashgrove Eco-milk Full Cream 2 — add to cart
- A2 Milk Light Milk 3 — add to cart
- Pepe Saya Cultured Butter Salted 100 — add to cart
- Bertolli With Butter With A Pinch Of Sea Salt 450g — add to cart
- Le Superbe Gruyere Cheese 195g — add to cart

Contributions

Contributions

- An algorithm for multi-vector slerp
- An algorithm for random walks and diversified recommendations
- Guides for query prompting to semantically filter
- Multimodal LLM aided approach to real-time personalisations

Further work

- Continued deployment and production testing of these approaches
  - Multipart weighted queries to negate qualities and LLM assisted complementary item recommendations are deployed

Thank you!

Contact us:

Owen Elliott
owen@marqo.ai

Tom Hamer
tom@marqo.ai

Jesse Clark
jesse@marqo.ai