Designing Interfaces for Multimodal Vector Search Applications*

Owen P. Elliott[1,*,†], Tom Hamer[2] and Jesse Clark[1]

[1] *276 Flinders Street, Melbourne, VIC 3000, Australia*
[2] *15 Kearny St, San Francisco, CA 94108, USA*

Abstract

Multimodal vector search offers a new paradigm for information retrieval by exposing numerous pieces of functionality which are not possible in traditional lexical search engines. While multimodal vector search can be treated as a drop in replacement for these traditional systems, the experience can be significantly enhanced by leveraging the unique capabilities of multimodal search. Central to any information retrieval system is a user who expresses an information need, traditional user interfaces with a single search bar allow users to interact with lexical search systems effectively however are not necessarily optimal for multimodal vector search. In this paper we explore novel capabilities of multimodal vector search applications utilising CLIP models and present implementations and design patterns which better allow users to express their information needs and effectively interact with these systems in an information retrieval context.

Keywords

Multimodal, CLIP, Information Retrieval, Vector Search

1. Introduction

Different search backends lead to differing search experiences. This necessitates considered implementation of methods of interaction. Modern multimodal search applications leverage artificial intelligence (AI) models capable of producing representations which unify different modalities. While a multimodal vector search system can be treated as a drop in alternative to a traditional keyword search engine, merely using it as a direct replacement doesn't exploit its full potential. The fundamental components of a standard search interface have remained largely unchanged since early research into interfaces for statistical retrieval systems, such as inverted indices with TF-IDF[1] or BM25[2]. Emerging areas, such as generative AI, have driven the development of new Human Computer Interaction (HCI) paradigms. Chatbot agents such as OpenAI's ChatGPT[3] have exposed users to new ways of seeking information with natural language[4, 5]. Multimodal vector search systems offer a similar green field for HCI research.

In this paper we explore techniques and interface elements for multimodal vector search in online image search applications[1]. In particular, we focus on multimodal systems built with CLIP models[6], however much of the content generalizes to other multimodal models (such as ImageBind[7] or LanguageBind[8]). We provide visual examples of UI implementations and define the concepts of query refinement, semantic filtering, contextualisation, and random recommendation walks as they pertain to multimodal information retrieval. We aim to provide practical implementations who's complexity can be hidden from the user making them suitable for non-expert users.

CKIM 2024 MMSR Workshop, Oct 25, 2024, Boise, Idaho, USA
*Corresponding author.
owen@marqo.ai (O. P. Elliott); tom@marqo.ai (T. Hamer); jesse@marqo.ai (J. Clark)
https://owenelliott.dev/ (O. P. Elliott)
0009-0002-3798-5870 (O. P. Elliott); 0009-0002-8775-9724 (T. Hamer); 0009-0006-5156-8554 (J. Clark)

[1] Many of the elements discussed here are implemented in our demo UI for hands on experimentation https://customdemos.marqo.ai/?demokey=cikm2024mmsr

2. Properties of Multimodal Models and Representations

To develop effective methods of interaction for multimodal vector search applications, it is essential to understand the properties of multimodal models and representations. In this section, we discuss the properties of CLIP models and vector representations for multimodal search.

2.1. Properties of CLIP Models

CLIP models are a class of models trained to encode images and text into a shared embedding space[6]. CLIP models are trained on large datasets of text and image pairs[9] to maximize the cosine similarity between matching image-text pairs and minimize the similarity between non-matching pairs, typically done with in-batch negatives. This allows for the model to be used for a variety of tasks such as zero-shot classification and retrieval.

2.2. Vector Representations for Multimodal Search

Multimodal models, such as CLIP, create vectors for each modality that exist within a shared space. Multiple vectors of one or more modalities can be combined into a single representation via weighted interpolations, such as linear interpolation (lerp) or spherical linear interpolation (slerp)[10].

Given a set of n vectors $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n \mid \|\boldsymbol{v}_i\| = 1\}$ in $\mathbb{R}^d$, and their corresponding weights $W = \{w_1, w_2, \ldots, w_n \mid w_i \in \mathbb{R}\}$, we can define lerp and slerp as follows:
Linear Interpolation (lerp):

$$\boldsymbol{v}_{\text{lerp}} = \sum_{i=1}^{n} w_i \boldsymbol{v}_i$$

Then, normalize the result to obtain the final result:

$$\hat{\boldsymbol{v}}_{\text{lerp}} = \frac{\boldsymbol{v}_{\text{lerp}}}{\|\boldsymbol{v}_{\text{lerp}}\|}$$

Spherical Linear Interpolation (slerp):

Spherical linear interpolation does not apply natively to n vector combinations, an iterative approach can be used to merge vectors hierarchically. The algorithm for hierarchical slerp is presented in Algorithm 1.

Algorithm 1 Hierarchical slerp Interpolation

```
Require: Set of unit vectors $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n\}$ and weights $W = \{w_1, w_2, \dots, w_n\}$
Ensure: Interpolated vector $\boldsymbol{v}_{\text{slerp}}$
 1: Initialize $V^{(0)} \leftarrow V$, $W^{(0)} \leftarrow W$
 2: while length of $V^{(k)} > 1$ do
 3:     Initialize $V^{(k+1)} \leftarrow []$, $W^{(k+1)} \leftarrow []$
 4:     for i = 1 to $\lfloor$length of $V^{(k)}/2\rfloor$ do
 5:         Compute weights sum: $w_{\text{sum}} \leftarrow w_{2i-1}^{(k)} + w_{2i}^{(k)}$
 6:         Compute interpolation parameter: $t \leftarrow \frac{w_{2i}^{(k)}}{w_{\text{sum}}}$
 7:         Compute interpolated vector: $\boldsymbol{u}_i^{(k)} \leftarrow \text{slerp}(\boldsymbol{v}_{2i-1}^{(k)}, \boldsymbol{v}_{2i}^{(k)}, t)$
 8:         Update weights: $w_i^{(k+1)} \leftarrow \frac{w_{\text{sum}}}{2}$
 9:         Append $\boldsymbol{u}_i^{(k)}$ to $V^{(k+1)}$, $w_i^{(k+1)}$ to $W^{(k+1)}$
10:     end for
11:     if length of $V^{(k)}$ is odd then
12:         Append the last vector and weight unchanged to $V^{(k+1)}$ and $W^{(k+1)}$
13:     end if
14:     Update $V^{(k)} \leftarrow V^{(k+1)}$, $W^{(k)} \leftarrow W^{(k+1)}$
15: end while
16: return $V^{(k)}[0]$
```

where the function $\text{slerp}(\boldsymbol{v}_0, \boldsymbol{v}_1, t)$ is defined as follows:

$$\text{slerp}(\boldsymbol{v}_0, \boldsymbol{v}_1, t) = \frac{\sin((1-t)\Omega)}{\sin\Omega}\boldsymbol{v}_0 + \frac{\sin(t\Omega)}{\sin\Omega}\boldsymbol{v}_1$$

and $\Omega = \arccos(\boldsymbol{v}_0 \cdot \boldsymbol{v}_1)$.

Combined representations via lerp and slerp merge understanding from multiple fields and modalities into a single unit normalized vector which can be compared to other merged vectors or individual vectors produced by the same model. This property arises naturally with CLIP models however techniques such as Generalized Contrastive Learning (GCL) can also be used to directly optimise for this[11].

3. User Interface Elements and Implementations

In this section, we present user interface elements and their implementations for multimodal vector search applications. These elements are inspired by the nature of CLIP models and properties of multimodal representations discussed in Section 2.1 and Section 2.2.

3.1. Query Refinement

Query refinement is not something new in the field of information retrieval, however multimodal vector search enables novel and effective implementations. By merging the query with additional queries, users can provide more context to the search engine, which can lead to more relevant results. This can be done iteratively by interpolating additional query vectors with positive or negative weights. Vectors for queries can be merged with approaches such as lerp or slerp as discussed in Section 2.2. Many existing search UIs treat search as a single shot process, similar to what is done in information retrieval benchmarking, in reality though, this is not reflective of real world scenarios. Users interact with retrieval systems in a search session where multiple queries are executed[12, 13], iterative refinement ties into this concept and bears semblance to other models of information retrieval such as berrypicking[14].

One way in which we can present this functionality is through additional input fields which enable query refinement with natural language as shown in Figure 1. Each input corresponds to a term which is vectorised and combined via linear interpolation with weights, "more of this" query terms are assigned a positive weight and "less of this" query terms are assigned a negative weight.

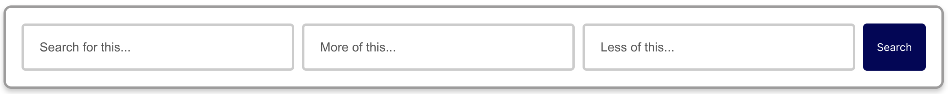

Figure 1: Multiple search fields for query refinement.

Formally, for a CLIP model M with text encoder M_{txt}, we can create refined queries from multiple queries as follows:

$$q_{\text{refined}} = \mathrm{N}\left[\left(\left(M_{\text{txt}}(\textit{dining chair}) \cdot 1.0\right) + \left(M_{\text{txt}}(\textit{scandinavian design}) \cdot 0.6\right) + \left(M_{\text{txt}}(\textit{upholstery}) \cdot -1.1\right)\right]$$

where $\mathrm{N}[\mathbf{v}]$ denotes the unit normalized version of the vector $\mathbf{v}$. This vector q_{refined} becomes the query vector for the search engine. The weights are abstracted from the user allowing for iterative refinement on results with natural language as shown in Figure 2.

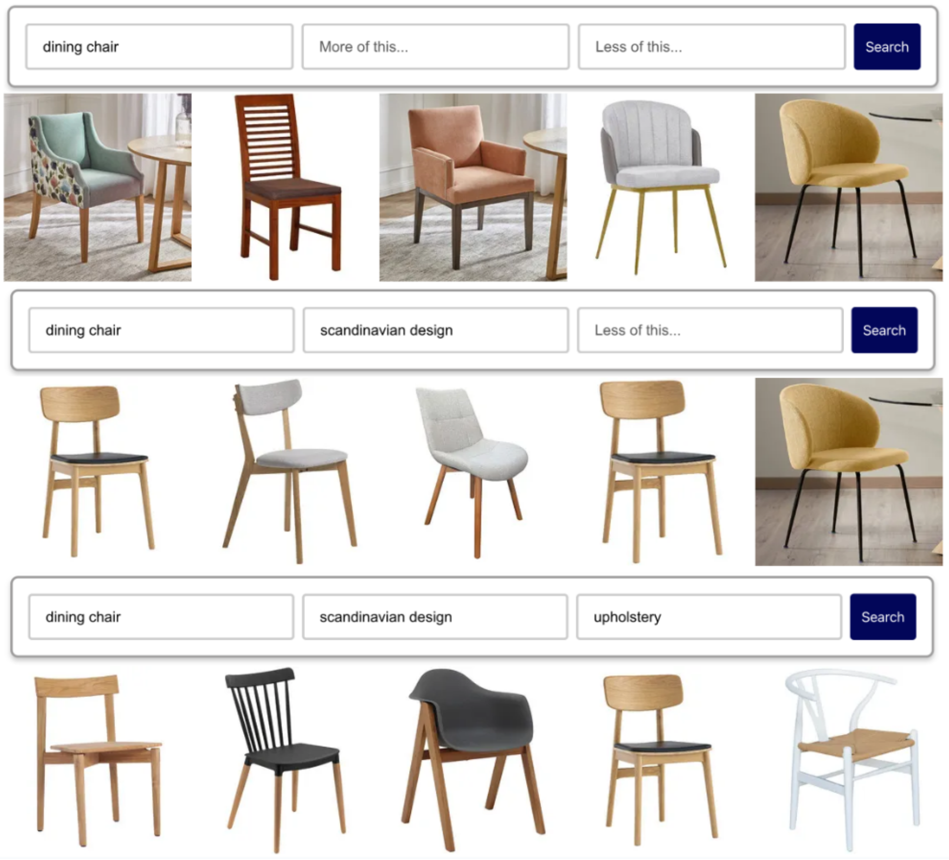

Figure 2: Iterative refinement of search results with multi-part queries. Data presented here is from an online furniture retailer.

3.1.1. Removing Low Quality Items

Query refinement can also be applied in marketplaces with large amounts of user generated content where quality of product listings can be dubious. By merging a query with a negatively weighted query term concerning quality we can dissuade the search from items relevant to the query indicating a lack of quality in the visual component of the listing. Queries can be merged with vectors such as

$(M_{\text{txt}}(\textit{low quality, low res, burry, jpeg artefacts}) \cdot -1.1)$. In a marketplace setting this can be used to encourage higher quality listings with more professional or appealing photos as shown in Figure 3.

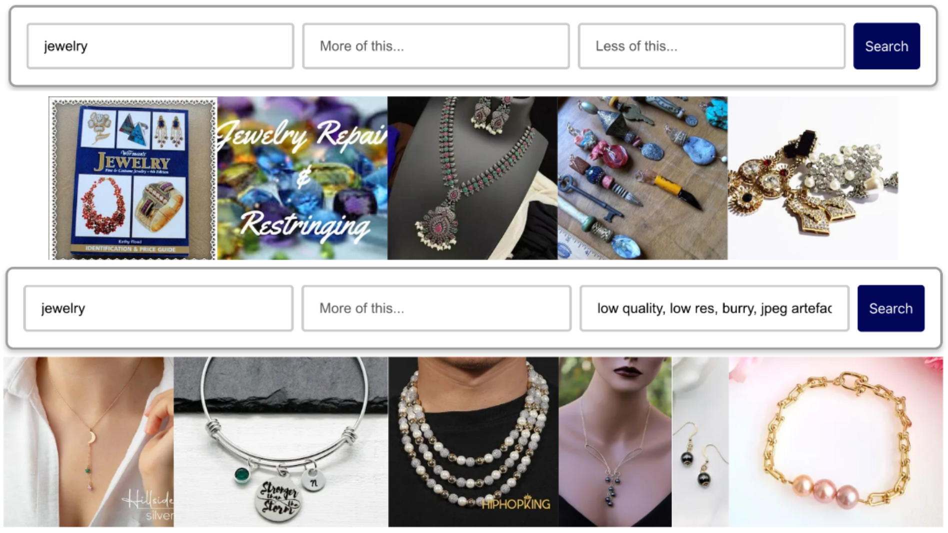

Figure 3: Query refinement to remove low quality items from search results.

3.2. Query Prompting and Expansion

In Section 2.1 we refered to how CLIP models are trained, providing an intuition as to the nature of the text that is in domain for these models. In search, we often encounter short queries of one or two words which don't provide the level of specificity which would be typically considered in domain for CLIP models given the captions they are trained with, this is similar to the problem of using CLIP for zero-shot classification. When performing zero-shot classification with CLIP, dataset labels are typically a single word, which does not align with the text captions seen in the model's training data. To work around this, labels are prefixed with additional text to convert it into a caption[15]. A simple prefix for class labels in zero-shot classification is "a photo of a" or "an image of a"[16].

We draw influence from CLIP zero-shot classification implementations and present "semantic filtering" as an approach to align queries with in domain captions and create query expansions with minimal user input. Semantic filtering alters the semantic representation of a query to control results in a similar manner to traditional filtering, without the need to label metadata. It provides a structured way to perform query expansions[17, 18, 19] to short queries without requiring an expert user to design a verbose query. This approach also draws inspiration from more modern prompt engineering strategies used with Large Language Models (LLMs)[20]. The goal is to expand this user submitted query with additional text within the model's context window. For example, to semantically filter to a boho style the, a query could be expanded with "A bohemian (boho) style image of a <QUERY>, rich in patterns, colors, and textures" where <QUERY> is the user submitted query.

The process of prompt design can be abstracted from the user, we can retain familiar UI elements while altering their backend implementation to expose new functionality to the user. This can be done by providing a set of predefined prompts which can be selected by the user to modify the query. A traditional selector as shown in Figure 4 is a suitable element to expose this functionality.

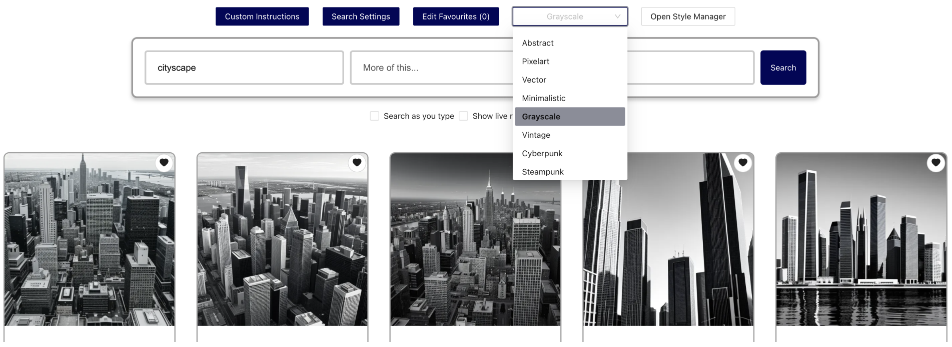

Figure 4: Query prompting with predefined prompts. In this example we use "A black and white, monochromatic image of a <QUERY>".

3.2.1. Realtime LLM Assisted Query Expansion

Semantic filtering can also be performed online with the inclusion of vision capable LLMs. Using direct or indirect user feedback on search results with a visual component we can prompt LLMs to extract query expansion terms to better align a user's search term with their desired information. This is useful when a user may not know the best way to describe a visual style they are looking for or if they are unaware of the semantic capabilities of the underlying search engine. The process is depicted in Figure 5.

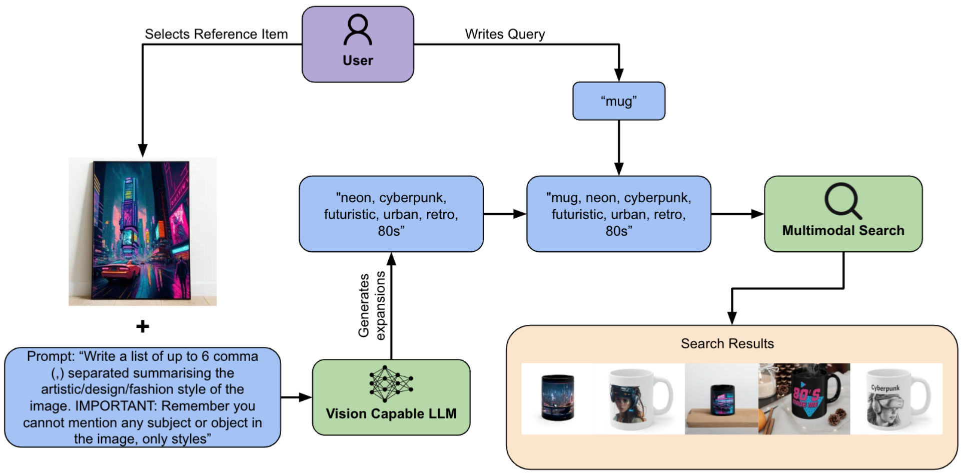
Selects Reference Item
User
Writes Query
"mug"
"neon, cyberpunk, futuristic, urban, retro, 80s"
"mug, neon, cyberpunk, futuristic, urban, retro, 80s"
Multimodal Search
+
Generates expansions
Prompt: "Write a list of up to 6 comma (,) separated summarising the artistic/design/fashion style of the image. IMPORTANT: Remember you cannot mention any subject or object in the image, only styles"
Vision Capable LLM
Search Results

Figure 5: Online query expansion via semantic filtering with LLM generated expansion terms from user preferences.

3.3. Realtime Personalisation and Contextualised Search

Taking influence from the field of relevance feedback[21, 22], vectors of existing documents in the index can be harnessed as query expansion terms in realtime, steering search results towards analogous items. Contextualisation can be broadly categorised into two types:

- **Intra-category Contextualisation:** These contextualise with items from the same category. For instance, recommending another watch based on a user's preference for a specific watch model.
- **Inter-category Contextualisation:** Here, contextualisations span different categories. An example might be tailoring search results for "couch" by a user's affinity for certain rug patterns or style of coffee table.

Intra-category contextualisation is the simpler of the two cases and can be achieved by combining a query with information from documents from its own result set, a well established pattern in relevance feedback. Inter-category contextualisation is more challenging; it is not something that is easily done with lexical search implementations, however with multimodal embedding models, information can be combined across categories. These contextualisations can be implemented with explicit, implicit, or pseudo relevance feedback.

Intra-category contextualisation can be achieved by merging the query vector with one or more results from the existing result set, the original query retains the majority of the weight, as shown in Figure 6.

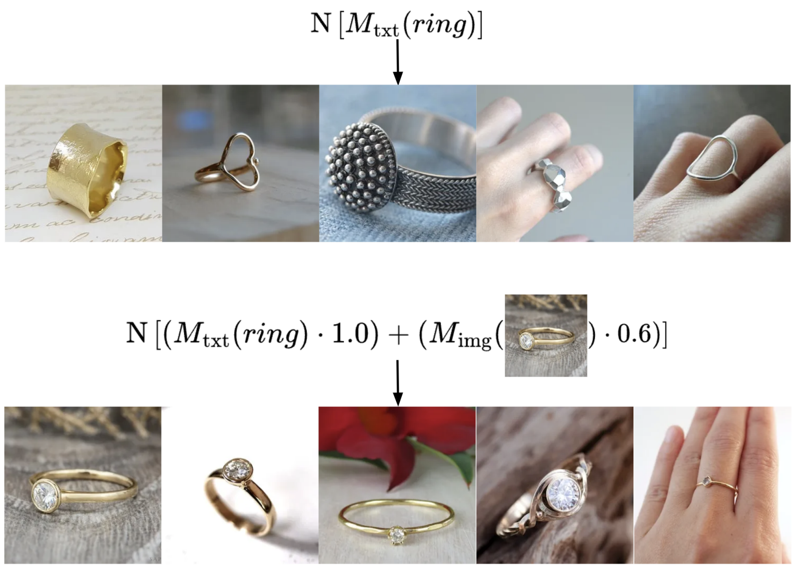
$\mathrm{N}\left[M_{\mathrm{txt}}(ring)\right]$
$\mathrm{N}\left[(M_{\mathrm{txt}}(ring) \cdot 1.0) + (M_{\mathrm{img}}(\quad) \cdot 0.6)\right]$

Figure 6: Contextualisation of a search for a watch with a similar watch model.

The ability of CLIP models to capture complex inter-category relationships can be applied to disconnected pieces of information, in Figure 7 we show that text queries can be contextualised with cross-modal information, in particular that a search for a backpack can be tailored with an image of a forest.

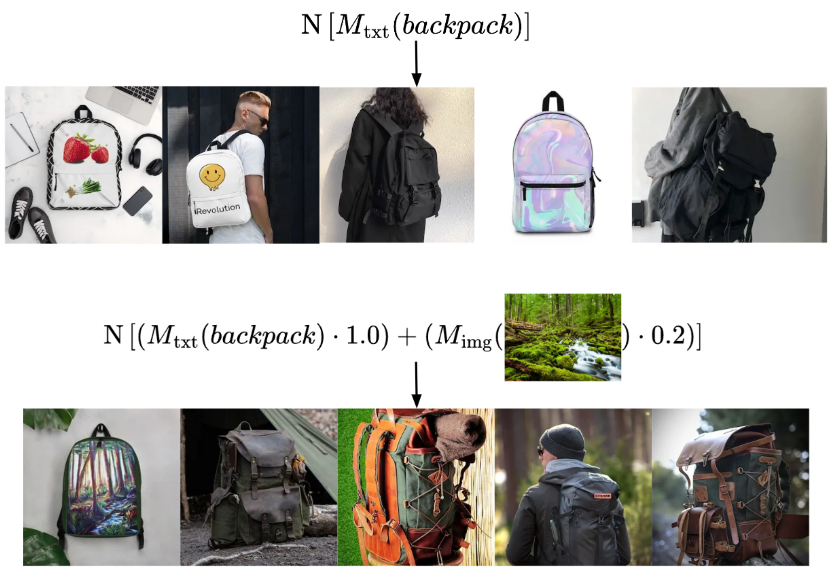
$\mathrm{N}\left[M_{\mathrm{txt}}(backpack)\right]$
$\mathrm{N}\left[(M_{\mathrm{txt}}(backpack) \cdot 1.0) + (M_{\mathrm{img}}(\quad) \cdot 0.2)\right]$

Figure 7: Contextualisation of a backpack search with an image of a forest setting, where a more rugged backpack would be suitable.

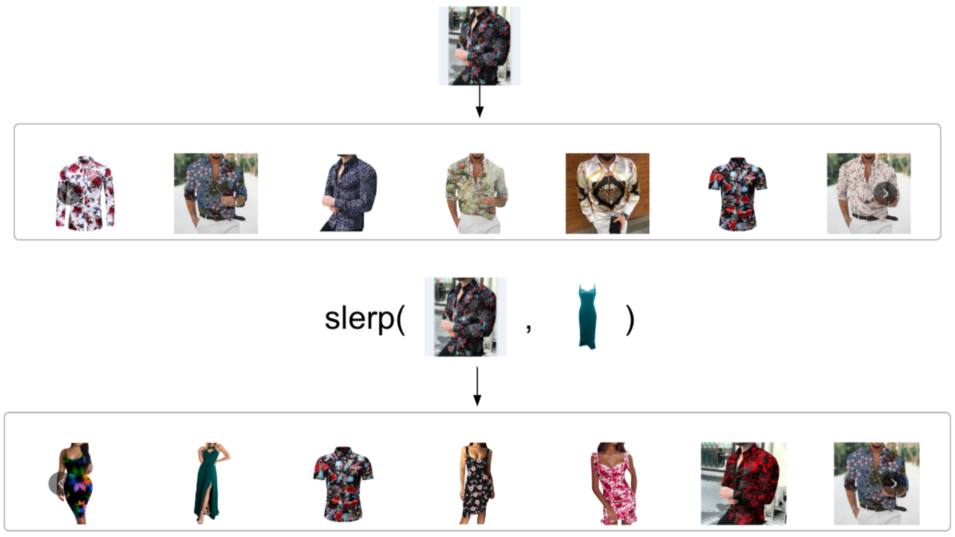

Figure 8: Recommendation ensembling effect between two product embeddings using slerp. Data used originates from a global online e-commerce retailer.

3.4. Recommendations as Search

Recommendations are an application of search. To formulate recommendations as a search problem we consider a query vector q in $\mathbb{R}^d$ which exists in the same embedding space as a corpus of vectors X; where in search q would be derived from a user submitted query, in recommendations this vector is derived from some other source, or combination of sources, which orients the vector towards suitable item recommendations. This formulation can be applied to multimodal search applications with models like CLIP; the high dimensional embedding spaces is sufficiently expressive, with enough degrees of freedom, to create these representations. Formulation of recommendations as a search problem is trivial for similar items however raises challenges for diversification of recommended items. We present two approaches to tackle this issue:

- **Vector Ensembling:** Merging vectors for disparate items to ensemble content.
- **Random Recommendation Walks:** Traversal of the vector space for adjacent items to explore diverse but related content.

3.4.1. Vector Ensembling

A recommendation vector can be constructed from document vectors, pieces of user information, or any combination of any number of both. Combination can be done with techniques such as lerp or slerp as discussed in Section 2.2. Interpolation between vectors of the same class (e.g. all document embeddings) with equal weights seeks a middle point between their representations which provides an ensembling effect where distinct classes of items can be retrieved by a single vector with some shared qualities. Using slerp preserves the geometric relationship between constituent vectors in the hypersphere, calculated as $\mathbf{v}_{\text{ensembled}} = \text{HierarchicalSlerp}(V, W)$ where $\forall w \in W,\ w = 1$. This is useful in online recommendations applications where interactions from clicks or add-to-carts (ATCs) can be used to build a dynamic list of products to ensemble when generating recommendations. An example of this ensembling effect is shown in Figure 8.

Utilising existing document vectors for the search means that recommendations can be done in realtime and has no cold-start problem for new products or users. Information can be gathered from a session on the fly without prior knowledge about the user[23].

3.4.2. Random Recommendation Walks

To diversify recommendations we must deviated from the immediate neighbourhood of our query vector without disregarding relevancy. Random walks can achieve this by finding neighbours to our initial recommendation vector, selecting neighbours, and exploring outwards from these neighbours (using their embeddings as queries). We present a process for performing random recommendation walks in Algorithm 2 and Algorithm 3.

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 $root$
```

Algorithm 3 Get Layer

```
Require: item, $C$: maximum children per node, visited: set of visited vectors, $k$: nearest neighbours
 1: $\mathbf{v}, children \leftarrow$ item
 2: $results \leftarrow$ NN$(\mathbf{v}, k)$ {Nearest neighbours search for $k$ neighbours}
 3: $filteredResults \leftarrow \{\mathbf{r} \in results \mid \mathbf{r} \notin$ visited$\}$
 4: if $filteredResults = \emptyset$ then
 5:    $children \leftarrow \emptyset$
 6:    return $\emptyset$
 7: end if
 8: $sampledResults \leftarrow$ RandomSample$(filteredResults, C)$
 9: Initialize $layer \leftarrow \emptyset$ {Empty list}
10: for each $\mathbf{r} \in sampledResults$ do
11:    $visited \cup \{\mathbf{r}\}$
12:    rData $\leftarrow \{(\mathbf{v}, \{\})\}$ {Vector and empty list for children}
13:    $layer \leftarrow layer$ || r_data {Append rData to $layer$}
14: end for
15: $children \leftarrow layer$
16: return $layer$
```

In practice, this output can be represented in a variety of formats. A typical grid or carousel layout can be used to display the results of the random recommendation walk. Another more tailored visualisation is to retain the tree structure created by the traversal as shown in Figure 9. These trees can be interactive to enable exploratory search and discovery.

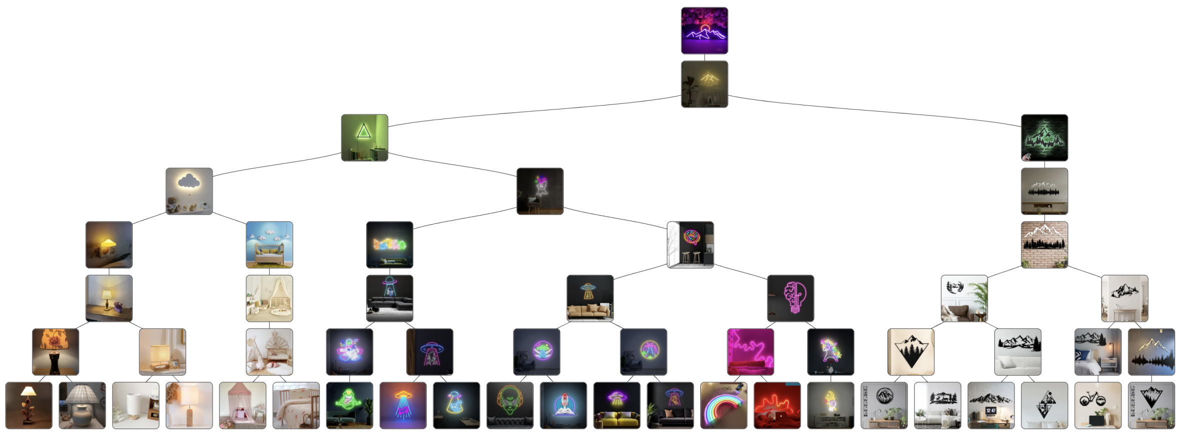

Figure 9: A recommendation tree generated by a random walk from neon lights. The walk explores adjacent concepts in neon lighting, general lighting, and interior design.

4. Conclusion

In this paper, we have explored the unique capabilities and enhanced user experiences offered by multimodal vector search systems, particularly those leveraging CLIP models. By understanding the properties of these models and their vector representations, we proposed novel user interface elements that can effectively facilitate the expression of information needs in a multimodal context. Techniques such as query refinement, semantic filtering, contextualisation, and recommendations offer the potential to improve search relevance and user satisfaction. The implementation of linear interpolations and spherical linear interpolations with hierarchical slerp, provides robust methods for combining vectors across different modalities. This allows for more nuanced and contextually relevant search results, demonstrating the unique properties of multimodal vector search when compared to traditional lexical search systems. Additionally, the introduction of vision capable LLMs for realtime query expansion further extends how multiple modalities can be leveraged in search experiences.

While our study focuses on CLIP models, the principles and techniques described are broadly applicable to other multimodal models. The proposed user interface elements and implementations are broadly applicable in various multimodal search applications. By presenting these multimodal search capabilities and their implementations, we hope to further understanding and ideation around how users can be enabled in describing their information need. Our goal is to deliver more intuitive and effective search experiences for users.

Acknowledgments

Thanks to Farshid Zavareh for the implementation of the hierarchical slerp algorithm (Algorithm 1).

References

[1] K. Sparck Jones, A statistical interpretation of term specificity and its application in retrieval, Journal of documentation 28 (1972) 11–21.

[2] R. Baeza-Yates, B. Ribeiro-Neto, et al., Modern information retrieval, volume 463, ACM press New York, 1999.

[3] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell,

A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, Łukasz Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, J. H. Kirchner, J. Kiros, M. Knight, D. Kokotajlo, Łukasz Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. Mossing, T. Mu, M. Murati, O. Murk, D. Mély, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, L. Ouyang, C. O'Keefe, J. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, Michael, Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. Tezak, M. B. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter, S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, B. Zoph, Gpt-4 technical report, 2024. URL: https://arxiv.org/abs/2303.08774. arXiv:2303.08774.

[4] A. Leonard, Conversational ai: How (chat) bots will reshape the digital experience, SALES and SERVICE (2019) 81.

[5] M. Skjuve, A. Følstad, P. B. Brandtzaeg, The user experience of chatgpt: Findings from a questionnaire study of early users, in: Proceedings of the 5th International Conference on Conversational User Interfaces, CUI '23, Association for Computing Machinery, New York, NY, USA, 2023. URL: https://doi.org/10.1145/3571884.3597144. doi:10.1145/3571884.3597144.

[6] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International conference on machine learning, PMLR, 2021, pp. 8748–8763.

[7] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, I. Misra, Imagebind: One embedding space to bind them all, 2023. URL: https://arxiv.org/abs/2305.05665. arXiv:2305.05665.

[8] B. Zhu, B. Lin, M. Ning, Y. Yan, J. Cui, H. Wang, Y. Pang, W. Jiang, J. Zhang, Z. Li, W. Zhang, Z. Li, W. Liu, L. Yuan, Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment, 2024. URL: https://arxiv.org/abs/2310.01852. arXiv:2310.01852.

[9] C. Schuhmann, R. Beaumont, R. Vencu, C. W. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. R. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, J. Jitsev, LAION-5b: An open large-scale dataset for training next generation image-text models, in: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022. URL: https://openreview.net/forum?id=M3Y74vmsMcY.

[10] K. Shoemake, Animating rotation with quaternion curves, SIGGRAPH Comput. Graph. 19 (1985) 245–254. URL: https://doi.org/10.1145/325165.325242. doi:10.1145/325165.325242.

[11] T. Zhu, M. C. Jung, J. Clark, Generalized contrastive learning for multi-modal retrieval and ranking, 2024. URL: https://arxiv.org/abs/2404.08535. arXiv:2404.08535.

[12] J. Teevan, C. Alvarado, M. S. Ackerman, D. R. Karger, The perfect search engine is not enough: a study of orienteering behavior in directed search, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 415–422. URL: https://doi.org/10.1145/985692.985745. doi:10.1145/985692.985745.

[13] R. Jones, K. L. Klinkner, Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs, in: Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM '08, Association for Computing Machinery, New York, NY, USA, 2008, p. 699–708. URL: https://doi.org/10.1145/1458082.1458176. doi:10.1145/1458082.1458176.

[14] M. J. Bates, The design of browsing and berrypicking techniques for the online search interface, Online review 13 (1989) 407–424.

[15] O. Saha, G. Van Horn, S. Maji, Improved zero-shot classification by adapting vlms with text descriptions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 17542–17552.

[16] OpenAI, Prompt engineering for imagenet, 2024. URL: https://github.com/openai/CLIP/blob/main/notebooks/Prompt_Engineering_for_ImageNet.ipynb, accessed: 2024-08-06.

[17] E. M. Voorhees, Query expansion using lexical-semantic relations, in: SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University, Springer, 1994, pp. 61–69.

[18] E. N. Efthimiadis, Query expansion., Annual review of information science and technology (ARIST) 31 (1996) 121–87.

[19] J. Xu, W. B. Croft, Improving the effectiveness of information retrieval with local context analysis, ACM Trans. Inf. Syst. 18 (2000) 79–112. URL: https://doi.org/10.1145/333135.333138. doi:10.1145/333135.333138.

[20] G. Marvin, N. Hellen, D. Jjingo, J. Nakatumba-Nabende, Prompt engineering in large language models, in: International conference on data intelligence and cognitive informatics, Springer, 2023, pp. 387–402.

[21] G. Salton, C. Buckley, Improving retrieval performance by relevance feedback, Journal of the American society for information science 41 (1990) 288–297.

[22] R. W. White, G. Marchionini, Examining the effectiveness of real-time query expansion, Information Processing & Management 43 (2007) 685–704.

[23] H. Cui, J.-R. Wen, J.-Y. Nie, W.-Y. Ma, Query expansion by mining user logs, IEEE Transactions on Knowledge and Data Engineering 15 (2003) 829–839. doi:10.1109/TKDE.2003.1209002.