A blue and white logo

AI-generated content may be incorrect.  
UNIVERSITY OF HERTFORDSHIRE  
School of Physics, Engineering and Computer Science

MSc Data Science and Analytics  
Data Science and Analytics Masters Project (7COM1075-0105)

Date:27/04/2025

“Evaluating Retrieval-Augmented Generation (RAG) based  
Personalized Product Discovery and Automated Content  
Generation”

Name : Anagha Santhosh  
Student ID : 23001866  
Supervisor : Dr. Jan Kim

MSc FPR Declaration

This report is submitted in partial fulfilment of the requirement for the degree of:  
Master of Science in Data Science and Analytics, at the University of Hertfordshire (UH).

I hereby declare that the work presented in this project and report is entirely my own, except where explicitly stated otherwise. All sources of information and ideas, whether quoted directly or paraphrased, have been properly referenced in accordance with academic standards. I understand that any failure to properly acknowledge the work of others could constitute plagiarism and may result in academic penalties.

**I did not use human participants in my MSc Project.**

I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

Table of Contents

[1 Abstract 4](#_Toc196252582)

[2 Introduction 4](#_Toc196252583)

[2.1 Current Issues 5](#_Toc196252584)

[2.2 Project Overview 5](#_Toc196252585)

[2.3 Aims and Objectives 6](#_Toc196252586)

[2.4 Research Question and Novelty 7](#_Toc196252587)

[2.5 Feasibility, Commercial Context, and Risk 8](#_Toc196252588)

[3 LITERATURE REVIEW 9](#_Toc196252589)

[3.1 Traditional versus Modern Recommendation Strategies 9](#_Toc196252590)

[3.1.1 Traditional Strategies: Collaborative and Content-Based Filtering 10](#_Toc196252591)

[3.1.2 Deep Learning and Neural Information Retrieval in Recommendation 10](#_Toc196252592)

[3.1.3 Hybrid Approaches and the Advent of RAG in Retrieval-Based Recommendation 11](#_Toc196252593)

[3.2 Retrieval-Augmented Generation in Action 11](#_Toc196252594)

[3.3 Retrieval models: Dense, Lexical, and Hybrid. 12](#_Toc196252595)

[3.3.1 Lexical Retrieval: Sparse, but Accurate. 12](#_Toc196252596)

[3.3.2 Dense Retrieval: Semantic Generalization 13](#_Toc196252597)

[3.3.3 Hybrid Retrieval: Managing Lexical and Semantic Signals 13](#_Toc196252598)

[3.4 Similarity Functions, and Implications for Ranking 14](#_Toc196252599)

[3.4.1 Similarity Functions in Dense Retrieval 14](#_Toc196252600)

[3.4.2 Lack of Strong Comparative Studies of Similarity Functions 14](#_Toc196252601)

[3.4.3 Diagnostic Evaluation using Correlation 15](#_Toc196252602)

[3.5 Evaluation Metrics in Retrieval and Recommendation 15](#_Toc196252603)

[3.5.1 Overview of Common Retrieval Metrics 15](#_Toc196252604)

[3.5.2 Use in NLP and RAG Evaluation 15](#_Toc196252605)

[3.5.3 Evaluation Challenges in Sparse Ground Truth Scenarios 16](#_Toc196252606)

[3.6 Inter-Model Comparison and Correlation Approaches 16](#_Toc196252607)

[3.6.1 Beyond Metric-Based Evaluation 16](#_Toc196252608)

[3.6.2 Spearman Correlation for Scoring Alignment 16](#_Toc196252609)

[3.6.3 Implications for System Design 17](#_Toc196252610)

[3.7 Explainability and User Trust in Recommendation Systems 17](#_Toc196252611)

[3.7.1 The Shift Toward Explainable Recommendations 17](#_Toc196252612)

[3.7.2 The use of RAG and transparency 18](#_Toc196252613)

[3.7.3 Aligning Performance and User Perception 18](#_Toc196252614)

[3.8 Summary and Research Gaps 18](#_Toc196252615)

[4 Methodology 19](#_Toc196252616)

[4.1 Overview of the Methodological Approach 19](#_Toc196252617)

[4.2 Project Design 20](#_Toc196252618)

[4.2.1 Input Layer – Query Handling 20](#_Toc196252619)

[4.2.2 Retrieval Layer – Semantic and Hybrid Pipelines 21](#_Toc196252620)

[4.2.3 Scoring Layer – Similarity Function Configuration 21](#_Toc196252621)

[4.2.4 Evaluation and Logging Layer 21](#_Toc196252622)

[4.2.5 Optional Generation Layer 22](#_Toc196252623)

[4.3 Data Collection & Preprocessing 22](#_Toc196252624)

[4.3.1 Query Data – Source, Filtering, and Splitting 22](#_Toc196252625)

[4.3.2 Product Metadata – Cleaning and Category Completion 22](#_Toc196252626)

[4.3.3 Dataset Integrity and Ethics 23](#_Toc196252627)

[4.4 Choice of Retrieval Methods 23](#_Toc196252628)

[4.4.1 Semantic Retrieval 23](#_Toc196252629)

[4.4.2 Hybrid Retrieval 24](#_Toc196252630)

[4.4.3 Comparative Rationale 24](#_Toc196252631)

[4.5 Embedding Models and Similarity Scoring 25](#_Toc196252632)

[4.5.1 Embedding Models Used 25](#_Toc196252633)

[4.5.2 Similarity Scoring Functions 25](#_Toc196252634)

[4.5.3 Design Justification 26](#_Toc196252635)

[4.6 Evaluation Strategy 26](#_Toc196252636)

[4.6.1 Standard Retrieval Metrics 27](#_Toc196252637)

[4.6.2 Diagnostic Analysis - Spearman Rank Correlation 27](#_Toc196252638)

[4.6.3 Evaluation Pipeline and Tools 28](#_Toc196252639)

[4.7 Testing and Results Pipeline 28](#_Toc196252640)

[4.7.1 Execution Process 28](#_Toc196252641)

[4.7.2 Mean and Summary Metrics 29](#_Toc196252642)

[4.7.3 Configuration Flexibility 29](#_Toc196252643)

[4.7.4 Diagnostic Usefulness 30](#_Toc196252644)

[4.8 Results Validation 30](#_Toc196252645)

[4.8.1 Ground Truth Alignment 30](#_Toc196252646)

[4.8.2 Per-Query Metric Tracking 30](#_Toc196252647)

[4.8.3 Averaging across Queries 30](#_Toc196252648)

[4.8.4 Diagnostic Validation By Correlation 31](#_Toc196252649)

[4.8.5 Reproducibility and Experiment Tracking 31](#_Toc196252650)

[4.9 Tools and Technologies Used 31](#_Toc196252651)

[4.10 Ethical, Legal, and Social Considerations 32](#_Toc196252652)

[4.11 Practical Constraints and Limitations 33](#_Toc196252653)

[5 QUALITY AND RESULTS 34](#_Toc196252654)

[5.1 Evaluation Setup 34](#_Toc196252655)

[5.2 Retrieval Strategy Comparison 35](#_Toc196252656)

[5.3 Similarity Function 35](#_Toc196252657)

[5.4 Model Comparison 36](#_Toc196252658)

[5.5 Correlational Comparison: 37](#_Toc196252659)

[5.6 Interpretation and Takeaways 37](#_Toc196252660)

[5.6.1 Best-Performing Combinations 37](#_Toc196252661)

[5.6.2 Similarity Function Impact 37](#_Toc196252662)

[5.6.3 Embedding Model Behaviour 38](#_Toc196252663)

[5.6.4 Latency versus Accuracy Tradeoff 38](#_Toc196252664)

[5.6.5 Query Complexity 38](#_Toc196252665)

[5.7 Summary 38](#_Toc196252666)

[6 Evaluation and Conclusion 39](#_Toc196252667)

[6.1 Final Evaluation 39](#_Toc196252668)

[6.2 Project Management 39](#_Toc196252669)

[6.3 Insights Gained 40](#_Toc196252670)

[6.4 Comparison to Literature 40](#_Toc196252671)

[6.5 Challenges Reflection 40](#_Toc196252672)

[6.6 Future Work 41](#_Toc196252673)

[6.7 Conclusion 41](#_Toc196252674)

# Abstract

Product recommendation systems play a fundamental role in improving user experience and increasing engagement in e-commerce. However, many of the traditional approaches to recommendation rely on collaborative filtering or keyword based search, which lack contextual understanding and explainability. Alternatively, Retrieval-Augmented Generation (RAG) systems integrates document retrieval functionality into generative models, but the retrieval aspect is often treated as a fixed or ancillary module. Hence, we are interested in analysing retrieval strategies, specifically similarity scoring in semantic embeddings, hybrid filtering, and reranking in a RAG product recommender and examining their downstream impact on overall RAG product recommendation system efficacy.

The main goal was to investigate and compare semantic ,hybrid and reranking retrieval mechanisms across multiple similarity functions (cosine, inner product and L2 measures) using multiple early-ranking metrics such as Precision@K, Recall@K, and Mean Reciprocal Rank (MRR) using the amazon ESCI\_small dataset while defining queries against ground-truth product IDs. We utilized two embedding models -MiniLM and QAnet to vectorise product descriptions; we introduced a semantic retrieval and a hybrid retrieval that performs lexical filtering using Whoosh, and reranking was applied at the end of the RAG layer using cosine similarity. MLflow was used to log metrics and parameters across the configurations.

Results showed that hybrid retrieval with L2 similarity exceeded all measures of Precision@3, while the semantic-only retrieval method was better as K grew larger. Correlation analysis indicated the retrieval behaviour was similar across embedding models, confirming some overlap in methods was redundant.

This research illustrates the potential importance of retrieval optimisation as part of RAG frameworks and presents a modular evaluation setup for further development. Future work could include query-adaptive tuning, larger datasets, and retrieval systems in multiple languages.

# Introduction

In the current e-commerce landscape, product discovery systems are vital to forming consumers experience and satisfaction and ultimately their commercial success. As online retailers grow to feature millions of items, users are increasingly turning to options like intelligent search engines and recommendation systems as their salvation for appropriate product discovery in a timely manner. When adopting distinct support systems, it is important to acknowledge the limitations of traditional recommendations systems like collaborative filtering and just-in-time rule-based keyword matched strategies. These systems are often limited by their lack of context, cold start problems and uncertain transparency (Zhang et al., 2019). These limitations are interpreted as inefficiencies that, when bundled together can lead to disabling recommendations, lower conversions, and ultimately a decreased user trust in recommending systems.

As consumers become more aware of how their data is used, thinking of recommendations systems as opaque or overly personalized has given rise to the phenomenon referred as the "creepiness factor", where users assume that they are being perpetually observed (Zeng et al., 2021). These assumptions give rise to a need for explainable AI systems that provide accurate predictions, but also in an explainable way that users can internalize trust. Retrieval-Augmented Generation (RAG) serves as an interesting potential in this area because it will make its generation based on newly retrieved documentation such as, product descriptions or product reviews. By using retrieved data to ground its recommended output, RAG is inherently able to give the user a recommendation with clear and transparent source (Lewis et al., 2020).

## Current Issues

Contemporary product recommendation approaches face a two-fold challenge: to provide contextually relevant results and also operate in a transparent manner that fosters user trust. The traditional approaches (collaborative filtering vs. traditional keyword searching) do not understand user intent beyond literal matches. The newest approaches, which are reliant on LLMs, do not understand product category levels beyond ordered details, and are thus shoehorned into reliance on generation methods that can produce hallucinations or unsupported suggestions (Reimers and Gurevych, 2019).

Identifying a category of retrieval similar to our own RAG approaches in product recommendation was particularly challenging. Retrieval approaches are well-known in general open-domain QA, however we could not find studies which examined and isolated the influence of various similarity scoring approaches, nor studies that assessed the efficacy of hybrid lexical-semantic filtering product searches (Karpukhin et al., 2020). Furthermore, previous studies relying on early ranking metrics, from MRR to Precision@K, were concerned mostly with unobserved objectives, even though practical significance does exist for e-commerce platforms where user will expect the right answer as soon as possible (Amazon ESCI, 2023).

The project is a direct approach to these towards a response to many of the above-mentioned gaps, through an examination and comparison of potential retrieval processes in a RAG-based product recommendation system; we will discuss the commercial and user implications of greater improvement of early ranking metrics in the evaluation and conclusion sections.

## Project Overview

For the purposes of this document, we define anything that falls under Retrieval Augmented Generation (or "RAG") as a set of systems that first processes a user query through a retrieval engine for relevant documents or product information and then passes the results to a generative model to produce an informed response that is contextually connected based on the retrieved results (Lewis et al., 2020).

This work will emphasize the comparison of retrieval methods specifically-not generation methods for this implementation of RAG. Based on this, a semantic retrieval module was developed using the dense vector embeddings of the sentence-transformer models (Reimers and Gurevych, 2019) that are indexed with ChromaDB (ChromaDB, 2023), while a hybrid retrieval pipeline was developed using both a lexical filter via the Whoosh search engine (Whoosh, 2022) and semantic search. After results were obtained from both modes of retrieval, results were reranked (early relevance is always key for user-facing applications) for final delivery using cosine similarity to prioritize relevance.

While generation wasn't the primary experimental focus, we included a response generation module via Llama 2 to realize the end-to-end system capabilities. The project also has an automated evaluation pipeline, looping through a ground truth labeled query set from the Amazon ESCI dataset (Amazon ESCI, 2023). Evaluation metrics (e.g., Precision@K, MRR, and Recall@K) were logged and compared using MLflow (MLflow, 2023), facilitating normalized tracking across experimental settings. Finally, an exploratory effort was conducted to see if variations in query length result in differences in retrieval performance, but the expected relationships were unsubstantiated since all the queries in the dataset were uniformly short, with no observable impact.

## Aims and Objectives

The overarching aim of this project is to create and evaluate a Retrieval-Augmented Generation (RAG) system for product recommendation, with an emphasis on the first stage of retrieval. Specifically, this project will evaluate effect on retrieving relevant product at the top of the ranking by means of different similarity scoring functions and retrieval approaches.

In support of this aim, the following objectives were derived:

* Build a semantic retrieval pipeline with vector embeddings and ChromaDB.
* Create a hybrid retrieval pipeline of lexical filtering with Whoosh and semantic retrieval.
* Use cosine similarity-based reranking to produce a merged candidate result.
* Compare performance of similarity function, such as cosine similarity, L2 (or Euclidean), and inner product.
* Use retrieval methods in the ESCI dataset, provided by Amazon, and evaluate the retrieval metrics of Precision@K, MRR, Recall@K etc.
* Store experiments with MLflow for reproducibility and consistent naming.
* Discuss the short, preliminary investigation of the impact of different query lengths for retrieval.

These objectives provide the theme for the experiments and evaluations that were carried out, and are discussed in later sections.

## Research Question and Novelty

This project is driven by the following research questions:

* RQ1: What are the effects of similarity function choice (e.g., cosine similarity, inner product, or L2 distance) on quality of Retrieval-Augmented Generation (RAG) system results for recommending products?
* RQ2: Do lexical filtering and semantic retrieval in combination lead to early ranking advantage compared to only using semantic retrieval?

The novelty of this project is based in that it will focus solely on the retrieval stage of the pipeline in RAG-based systems, which is often privileged as a fixed or auxiliary stage. Although previous work has focussed on the quality of generated content, this project will take a retrieval-first approach. It will separate and compare similarity functions, introduce a hybrid retrieval pipeline that continues to evaluate relative performance using early-ranking metrics such as Mean Reciprocal Rank (MRR) and Precision@K.

The project provides a more detailed understanding of what drives RAG performance in real-world recommendation scenarios, particularly with short queries and few ground truth labels such as in the ESCI dataset. It also brings attention to the challenges of measuring retrieval performance in sparse-query environments that have historically received less attention in traditional benchmarks. It further supports the usefulness of early-ranking metrics and systematic evaluation approaches.

Moreover, beyond core metrics, the project adds a valuable analytical dimension for comparing embedding models based on their similarity scoring behaviour. By determining per-query cosine similarities between a query and ground truth products, and calculating Spearman correlations, it provides insight into whether related retrieval metrics are based on genuinely aligned model behaviour or pure redundancy based on similarities in model architecture. This is an added analytic utility which enable deeper insight at the level of system.

Finally, the hybrid retrieval system involves lightweight lexical filtering (via Whoosh) and semantic search with cosine-based reranking. This approach is theoretically simple, but is infrequently examined in RAG pipelines and demonstrates a valid option for incorporating improvement to explainability and performance, without relying solely on heavy generative models.

In summary, this retrieval-centric perspective on RAG systems provides not only methodological innovations, but also practical insights for system design, evaluation, and application to real-world problems.

## Feasibility, Commercial Context, and Risk

This project exhibits a high degree of feasibility, both from a technical and economic perspective. It was built with 100% open-source tools: it used ChromaDB for vector storage; Whoosh for lexical filtering; Sentence Transformers for the semantic embedding generation; and MLflow to track its experiments. Since these systems are free from licensing cost, and because they are modular, this solution is easy enough to experiment with as a potential commercial solution, especially for use in budget constrained areas or if scale is a worry.

Commercially, the primary goal of increasing retrieval accuracy within a recommendation pipeline deserves emphasis for its commercial relevance. Even marginal advantages in defining how quickly a user identifies relevant products may lead to improvements in user engagement, increased conversion rates, and clearing more customers from the churn bucket. Additionally, the use of a Retrieval-Augmented Generation (RAG) framework enables recommendations to be explainable -because who doesn't want context and justification for how they arrive at decisions. Furthermore, this generally higher standard of transparency helps businesses address increased consumer expectations for transparency in personalization and from AI-driven automation in decision making.

That said, the project also provided insight into a number of limitations and risks. From a dataset point of view, the evaluation has taken place using a portion of the Amazon ESCI dataset that presented useful information but with challenges. Many queries are mapped to a single ground truth product, reducing the ability to evaluate larger retrieval diversity. Furthermore, the dataset's limited size and limited coverage of categories could impact the transfer of results to commercial systems in the real world. There was some manual cleanup and matching required for the alignments of query-product pairs. We were concerned that to be successful at scale, large preprocessing pipelines would be required to make the data useable.

From a systems perspective, differences observed between the semantic-only and hybrid retrieval approaches were fairly trivial. This may give rise to a business school-style evaluation of the return-on-investment from adopting hybrid retrieval as a strategy. Additionally, while generation itself has been demonstrated, integration into the larger retrieval system was possible using the LLaMA2 model, latency and compute requirements are a mitigating fact for deployment at a production grade; particularly in real-time contexts.

Market-wise, adoption risks manifest in several ways including functionality overlap with existing systems, reluctance to change from engineers, and the competitive landscape from other AI-driven recommendation platforms. Ethical and user-perception risks are also pertinent: users may be sensitive to over-personalization or to profiling, particularly if the system does not exhibit explainability.

These commercial and operational aspects themselves will be share at additional length when presenting the evaluation and conclusion chapter, where the real-world impact of performance within each of the elements is assessed.

# LITERATURE REVIEW

Product recommendation has become an essential function of e-commerce, aimed at providing users with timely and relevant suggestions that drive engagement and purchase behaviour. Established methods such as collaborative filtering and content-based recommendation have provided commercial success, and some value as measures and methods, but with some significant limitations in terms of cold-start, context-awareness, and transparency (Ricci et al., 2011; Zhang et al., 2019).

As a result of these limitations, recent development in language models and retrieval-based systems have radically changed the way intelligent recommendations can be produced. Specifically, subject to implementation of Retrieval - Augmented Generation (RAG) in the form of a request to a language model combined with precision retrieval, RAG systems promise advantages in retrieval (factual basis, precision) and generation (flexibility, language response). Originally developed for open domain question answering (Lewis et al., 2020), the use of RAG approaches is being adapted for task-specific applications such as customer support, medical diagnosis, and product discovery.

In these hybrid systems, the retrieval function can be seen as the game changer. Although much of the literature on RAG systems focuses on generation, the strategies for retrieval (options include dense (semantic), sparse (lexical), or hybrids) remain the black box and are also less researched, especially for the domain of product recommendation. Likewise, while retrieval functions include similarities such as cosine, inner product, and L2 distance, rarely are these evaluated as an experimental factor, even though they are significant in the determining ranked outputs.

The current project is situated at the intersection of retrieval theory, semantic embedding models and ranking evaluation and aims to examine how retrieval strategy impacts the perfectly formed RAG systems applied to product recommendation. Through the comparison of retrieval strategies and scoring means, the project addresses some key research questions related to relevance, explainability and early ranking effectiveness which are major considerations in these methods applied in the real world as they often rely on user satisfaction for their top-1 or top-3 rankings.

The following review, position, gap, and program of the experiment will consider other previous work and methodologies in these areas.

## Traditional versus Modern Recommendation Strategies

### Traditional Strategies: Collaborative and Content-Based Filtering

The majority of earlier recommendation systems relied on collaborative filtering (CF) or content-based filtering (CBF). In essence, CF uses the interactivity panel of a user-item matrix to predict the user's likeliness for a given recommendation (Resnick et al., 1994). Essentially, collaborative filtering assumes that users who have exhibited similar behaviours will maintain similar interests. CF limitedly achieved such success in earlier commercial product systems that CF methods were often touted as the foundation of most recommendations systems. However, it is worth noting, the collaborative-based filtering algorithm suffers from one main limitation, the cold start problem. There are too few users or items to obtain an historical record with which to recommend. Whereas, CBF attempts to recommend an item that most resembles a prior liked item by looking at framed features such, such as product attributes or metadata in structured form (Lops et al., 2011). The caveat here is that while CBF awaits the user-specific bias to occur for recommendation, CBF more often than not can lead to 'over-specialization' where system outputs are similarly narrow and repetitive recommendations. There are also the usual contextual caveats created by not respecting, nor taking into account, contextual information, which defines both of these methods under the elevations of consistent intent associated with the inquiry prompts, so there are nuances too and inherent fragility of possible outcomes of change.

Therefore, both the collaborative filtering and content-based filtering weaknesses encourage the legal pursuit of sufficient, contextually adaptive and semantically intelligent recommendation methodologies/styles, like perhaps methodologies utilizing deep learning or hybrid methods.

### Deep Learning and Neural Information Retrieval in Recommendation

The increased size and complexity of online catalogues has prompted the adoption of neural networks to model user-item interactions. Examples of architectures that learn an implicit feedback user-item interaction functions are neural collaborative filtering (He et al., 2017) and wide and deep learning (Cheng et al., 2016). These models have been shown to produce better results than traditional CF, especially when given enough data and signals of user behaviour, such as clicks and search queries.

In tandem, neural information retrieval (IR) methods, such as dense passage retrieval (Karpukhin et al., 2020), have already begun to influence recommendation architectures as well. Embedding-based models such as Sentence-BERT (Reimers and Gurevych, 2019), have an embedding-based representation of text to allow retrieval of recommendations based on semantic similarity rather than simple overlap on keywords. This is especially salient in the e-commerce space where additional variability in user queries (e.g., shortness, action-based queries, and unpredictability of language) can make static representations of queries inadequate.

That said, while these deep retrieval models could be of benefit to a recommendation system, it is likely that they are used as static elements in the architecture of a recommendation system. While the system components may evaluate system-level outcomes such as click-through rate or predictable accuracy from recommendations, little effort is done to evaluate the retrieval phase independently in investigating how to better recommend items.

This observation provides the supporting case for the purpose of this project, which considers retrieval performance as an experimental variable on its own.

### Hybrid Approaches and the Advent of RAG in Retrieval-Based Recommendation

Hybrid recommendation systems attempt to balance the strengths and weaknesses of CF (collaborative filtering) and CBF (content-based filtering), most commonly through late-fusion or ensemble methods. Recently, retrieval-augmented generation (RAG) has been embraced in applications where transparency and flexibility are paramount. Introduced for open-domain question answering (Lewis et al., 2020), RAG combines a retriever with a generative language model that retrieves external documents (in this example, product descriptions) by generating responses based on this content.

More generally, RAG has been widely utilized for QA (question-answering) and knowledge-intensive tasks yet not applied as widely in product recommendation tasks. Some have adapted RAG (Izacard and Grave, 2021) for downstream generation quality; but with fewer studies quantifying the differences in ranking quality from different retrieval strategies or similarity functions particularly in low-label settings like Amazon’s ESCI dataset (Reddy et al., 2022).

By shifting the focus of the investigation from generation to retrieval optimization, this project advances the developing field by exploring how hybrid filtering, semantic embeddings, and decisions around similarity scores interrelate with an impact on early-ranking, which is an obvious methodological gap in the literature.

## Retrieval-Augmented Generation in Action

Lewis et al. (2020) presents a model called retrieval-augmented generation (RAG), which augments the input to a generative language model by including externally retrieved content which improves generation in knowledge-intensive tasks. In RAG, retrieval is unified with generation: in practice, this means that RAG models build an augmented model using a retriever (typically dense-embeddings-based, such as DPR [Karpukhin et al., 2020]) with an encoder-decoder generative language model (e.g. BART, T5) and generate outputs which are fluent, aware of their context, and grounded in the most relevant retrieved documents, which reduces hallucinations and latent to more accurate factual consistency.

While RAG represents an exciting advance, it is relatively recent and this has led to very recent studies exploring the technique as a new toolkit to deploy in open-domain question answering (Izacard and Grave, 2021; Min et al., 2023), summarization, or dialogue systems. In these retrieval-augmented generation tasks, the model is allowed to retrieve many passages per query, and then synthesize multiple retrieved passages into a coherent sequence of text via sequence-to-sequence modelling, with occasional fine-tuning of the retrieval and generation components. The vast majority of studies only evaluate generation quality (e.g. via automatically calculated metrics such as BLEU, ROUGE or factual accuracy), and did not use a retrieval model other than as fixed, or lightly fine-tuned.

On the other hand, the use of RAG for product recommendation is much less common in the research literature. Although most commercial recommendation engines are retrieval-based, these engines aren't considered generative. While RAG has been used in e-commerce or product-based applications (see Amazon QA or BEIR tasks), the research papers focus primarily on document retrieval / semantic matching, rather than defining end-to-end RAG pipelines focusing on recommendation.

Additionally, RAG systems are never evaluated in terms of early ranking metrics (Precision@K or MRR), while we know that in recommendation situations, user satisfaction is tremendously sensitive to the quality of their top-ranked outputs (Reddy et al., 2022). This is an important methodological gap that this project intends to fill.

Finally, RAG literature often neglects the choice of retrieval scoring function. Most dense retrieval systems default to cosine similarity, however, there are very few studies interested in how other functions such as inner product, and L2 distance, might affect retrieval effectiveness, especially within narrow domains or sparse label conditions. Hybrid retrieval strategies that include sparse lexical filters (e.g. BM25, keyword matchers) along with semantic reranking are similarly absent from RAG experiments, even if hybrid approaches (like the use of both lexical and semantic information) have long been used in classical Information Retrieval (IR) and search pipelines.

This study applies RAG to product recommendation and, because the retrieval phase will be varied as an experimental variable, it provides a key insight into an under-represented dimension of the architecture, where retrieval has been used as an experimental variable. The use of lexical–semantic hybrid search, the comparison of similarity functions, and measurements of model scoring behaviour based on correlation are all intended to provide greater insights to the dynamics of RAG's retrieval phase beyond examining generative results.

## Retrieval models: Dense, Lexical, and Hybrid.

### Lexical Retrieval: Sparse, but Accurate.

Lexical retrieval approaches, like BM25 (Robertson and Zaragoza, 2009), compute document rankings through exact term matching and term frequency statistics. Lexical retrieval approaches are robust in situations that require high-precision retrieval, typically in cases where query terms and document terms draw from the same vocabulary. Lexical retrieval methods remain a standard for search engines and e-commerce sites primarily for their interpretability and efficiency. That said, it is worth noting, lexical approaches to retrieval are extremely brittle when presented lexical paraphrased queries, synonyms, and cannot cleanly extract semantic relevance for document rankings typically found in natural language generation contexts.

Lightweight and low-latency developments, like Whoosh, and other lexical engines will continue to be deployed in production contexts that prefer keyword indexes but also tolerate low-latency keyword queries. Lexical filters may be very useful in product recommendations systems where retrieving large product catalogs will need to be be done first, before searching/semantic reranking of the product listings.

### Dense Retrieval: Semantic Generalization

Unlike sparse retrieval, dense retrieval represents both queries and documents as dense vectors in a joint semantic space. Models like Dense Passage Retrieval (DPR) (Karpukhin et al., 2020), and Sentence-BERT (Reimers and Gurevych, 2019) enable semantic matching even when lexical matches are minimal or non-existent. Dense retrievers have emerged as popular choices in open-domain QA and have plenty of literature references with benchmarked datasets like Natural Questions and MS MARCO (Bajaj et al., 2016).

In exchange for their unique capabilities, dense retrieval requires additional computational resources, and their ranking decisions tend to be less interpretable compared to the lexical operations. Moreover, they may struggle without fine-tuning for domain-specific vocabularies or short queries, a natural characteristic in product recommendation-based use cases.

### Hybrid Retrieval: Managing Lexical and Semantic Signals

Hybrid retrieval solutions attempt to combine the two methods mentioned above. A typical approach is to apply a sparse lexical filter first, and subsequently recalibrate the remaining set based on a dense similarity score (Ma et al., 2021). This model works well with large candidate sets, such as an e-commerce product catalogue, as the first lexical filter limits potential noise while maintaining relevance to the structured metadata.

While hybrid pipelines are prevalent in traditional IR systems, they are seldom assessed in RAG frameworks, particularly in recommendation situations. Most RAG studies focus the evaluation on how the generation went after the retrieval mechanism (usually dense) was fixed. This study attempts to deal with this issue by implementing and comparing both dense-only and hybrid pipelines and particularly looking at early-ranking effectiveness via MRR and Precision@K.

Furthermore, by injecting hybrid retrieval into (1) a short-query, (2) sparse-label dataset (ESCI), it assesses its appropriateness in proceeding the retrieval when not all relevance signals are known – highlighting hybrid search's practical potential for improving top-K relevance in constrained recommendation contexts.

## Similarity Functions, and Implications for Ranking

### Similarity Functions in Dense Retrieval

For dense retrieval systems, the similarity function specifies how query embeddings and document embeddings are contrasted when retried. The cosine similarity is by far the most often used because it links angular distance measurements to differentiating retrieval of items with similar relevance characteristics in high-dimensional embedding spaces. The cosine similarity function is implemented in models such as Sentence-BERT (Reimers and Gurevych, 2019). In such models, the embeddings are normalized to unit length which makes cosine an intuitive and efficient metric.

However, cosine similarity is not the only option. The inner product (or dot product) calculates the raw projection of one vector onto another while allowing models to encode both direction and magnitude into the embeddings. This is particularly useful since the magnitude of the embedding may provide useful information about relevance or confidence (Li et al., 2020). Some argue that the models trained with inner product loss functions may preserve ranking gradients better for large-scale retrieval tasks (Xiong et al., 2020).

Conversely, L2 distance (Euclidean distance) treats retrieval as a nearest-neighbour problem situated in Euclidean space. L2 is less often used in NLP than in vision, but does suit environments when absolute positions of the embeddings matter; L2 has been used in initial retrieval studies, and in hybrid embedding spaces (Johnson et al., 2017).

### Lack of Strong Comparative Studies of Similarity Functions

The bulk of evaluations in dense retrieval literature explores the effect of different embedding architectures, while the choice of similarity function is mostly fixed and considerably under-evaluated. Denser retrieval benchmarks (e.g., MS MARCO, BEIR) tend to rely on cosine similarity as the default similarity function, and cases where the implementation of an alternative metric improving ranking performance have not been explored, especially in sparse or low-label cases.

The dearth of comparative research was a principal reason for undertaking this project. This project reports findings regarding retrieval effectiveness based on scoring functions (cosine similarity, inner product, and L2 distance), and this contributes information on how different scoring functions affect early precision, mean reciprocal rank, and overall alignment in the outcome of retrieval results. This work is valuable because it evaluated retrieval of short-query, product-task data where norms for embedding and semantically-nuanced retrieval relevance may affect differentiation of scores when ranking.

### Diagnostic Evaluation using Correlation

In addition to measuring which function retrieved the highest ranked ground truth, this project also incorporated correlation analysis (Spearman) to measure whether different scoring functions had similar or dissimilar ranking behaviour. For instance, two models may both achieve the same Precision@K but differ on how they score or rank the individual items. Measuring correlations is useful to separate out real alignment from coincidental agreement, and also a more useful diagnostic measure of performance.

We want to note that this additional analysis is valuable as a methodological contribution to the literature, and draws attention to the fact that the quality of retrieval should be measured not only in terms of quality of outcomes, but also the scoring behaviour or approach behind generating these outcomes.

## Evaluation Metrics in Retrieval and Recommendation

### Overview of Common Retrieval Metrics

In evaluating the effectiveness of retrieval-based systems, both ranking and user relevance are metrics that need to be psychologically correct. Traditionally, research in information retrieval has relied heavily on metrics such as Precision, Recall, and F1-score (Manning et al., 2008), mostly in binary relevance settings. However, in some ways, these P/R metrics did not adequately consider the rank order of relevant results-an important component of user relevance in the context of recommendation and search systems where users can only act on at most the top few retrieved results.

In this respect, the earliest ranking metrics that were introduced such as Precision@K and Mean Reciprocal Rank (MRR) have become standard categories of metrics when the focus is to retrieve relevant results early. Precision@K measures the proportion of relevant documents within the top K results, and is particularly useful in product recommendation contexts where a user may only view, or click, on a small number of recommended items. MRR captures the position of the first relevant document in the ranked list and provides a reward for those systems that can return the correct item first.

### Use in NLP and RAG Evaluation

In open-domain QA and dense retrieval settings, **MRR** and **Recall@K** are frequently

reported to evaluate models like DPR (Karpukhin et al., 2020) and ColBERT (Khattab and Zaharia, 2020). However, within RAG systems generally, especially when it comes to recommendation tasks, retrieval evaluation can be difficult to characterize as ranking is often neglected. Many studies focus on the quality of the generation outputs, judging the final output based on measures like ROUGE or BLEU or factual consistency analysis, and neglect to study the retrieval phase of these systems (Lewis et al., 2020; Izacard and Grave, 2021).

This gap is of particular concern in recommendation tasks, where the retriever is responsible for filtering to actionable product candidates, and, therefore, the quality of its top-ranked outputs has a direct impact on user satisfaction and system effectiveness.

### Evaluation Challenges in Sparse Ground Truth Scenarios

One of the factors that makes this project interesting is the handful of labels in the Amazon ESCI dataset (Reddy et al., 2022)-many queries will only map to one relevant product. For relevant queries in which there is only one label, retrieval-based metrics that are recall biased become unstable or meaningless and measures of precision provide a more consistent signal. Additionally, while measures like NDCG@K will provide a rating system weighting based on position, they assume graded relevance and are complicated by the absence of any single label effect for many products and queries.

For the evaluation, the use of Precision@K and MRR is therefore particularly valid. Both measures provide the possibility to compare different retrieval strategies and similarity functions in a way that is consistent: they can indicate performance at both a system-level and per-query-level. In this way,the chosen measures will meet the practical evaluation goals, and be in line with an underdeveloped aspect of methodology in RAG methods.

## Inter-Model Comparison and Correlation Approaches

### Beyond Metric-Based Evaluation

Although performance can be assessed with standard retrieval metrics like Precision@K and MRR, these metrics take performance only so far. They can tell at a high level, how many relevant items appear in top-ranked positions, but they do not expose the retrieval behaviour underlying these performance measures-specifically, how two different models score or rank the same candidates overall. This is perhaps more interesting when models are achieving a similar performance (as can regularly occur in dense retrieval), raising questions about whether these models are functionally identical or demonstrate distinct behaviours (Thakur et al, 2021).

Yet despite this important area of focus for research in information retrieval, the direct comparison of retrieval behaviour of models has not typically been discussed in literature. Most prior studies on dense retrieval or RAG examined retrieval outcome or generative quality, without considering whether their systems have similar or different agreement scoring behaviour using alternative embedding mechanisms.

### Spearman Correlation for Scoring Alignment

This project uses Spearman rank correlation to compare similarity scores between two different embedding models over the same product-query ground truth pairs. Spearman’s ρ is a measure of the strength of monotonic relationship between two ranked lists (Spearman, 1904), which is useful for determining if the two models rank candidates similarly; the models’ absolute values (similarity scores) and top-K hits may vary, but the underlying rank could be similar.

This rank correlation analysis takes inspiration from other domains, notably representational similarity analysis in computer vision and interpretability studies (Li et al., 2016; Kornblith et al., 2019), but has been less prevalent in information retrieval and recommendation systems. Lin (2008) showed the use of rank correlation approaches in Information Retrieval evaluations; Lin makes the case for using rank correlation as a method to glean deeper ranking behaviour which could often be obscured by summary metrics.

As a correlation-based comparison, this project engenders whether two different embedding models replicate consistent ranking behaviours or whether they simply rank similarly in outcome by chance. This analysis and interpretation adds depth to traditional retrieval evaluation by providing a more interpretive method to evaluate model behaviours beyond just metrics scores.

### Implications for System Design

There are important implications to determine the degree of behavioural overlap between models for system architecture and resource distribution. If two embedding methods have highly correlated scores, there may not be much benefit to using ensemble or hybrid methods. In contrast, if the scores of two models have low correlation, regardless of their similar top level performance, this could imply that the models are capturing different semantic signals, and therefore might be more useful as complementary retrievals in multi-stage retrieval systems or re-rankers.

Developing any kind of correlation analysis into the evaluation process invites a methodological extension to retrieval benchmarking-where we are not just evaluating "how well" the systems perform, but asking the question "how similarly" they behaved.

## Explainability and User Trust in Recommendation Systems

### The Shift Toward Explainable Recommendations

As recommendation systems have grown integrated into digital life, the need for explainability has increased significantly. Users no longer accept being presented with black-box predictions, and are beginning to expect systems to provide rationales for recommendation outputs (Tintarev and Masthoff, 2007). This is especially true in areas that involves sensitive decision making, like healthcare, finance, or e-commerce, as the recommendations could influence a significant transaction or sensitive decision.

Traditionally, collaborative filtering, and recommendation systems recently based on deep learning all present a black-box model with little to no insight into why an item was recommended. many studies and research has pointed to lowered trust or engaging with the system less when reasoning and explanations are not presented, and when recommendations are overly appropriate or 'creepy' (Zeng et al., 2021).

### The use of RAG and transparency

Retrieval-Augmented Generation (RAG) may provide a solution by opening up the systems from the black-box to allow for the rationale of outputs to be clearly grounded to visible, retrievable content ie. user review(s), product description(s), or FAQ's ect. In this case, generative models still do not create recommendations from absolutely nothing, as there is evidence retrieved first, and then contextually presented in the response. This parlays in to mass transparency in the use of natural language, and makes it easier for users to trace the reason behind each suggestion.

While RAG was originally developed with open-domain QA, it has potential for product recommendation as well, setting the stage for explainable personalization, whereby systems can explain the recommended products in addition to making them--by pulling reviews that are similar to like visual features. This builds user trust as well as accountability for the algorithm in cases where regulatory issues arise (Shin, 2021).

### Aligning Performance and User Perception

The tension between modular complexity and explainability is a perennial concern. A highly optimized deep model, with its high degree of convergence around a particular optimization parameter will generally out-perform a simple or less optimized model on accuracy. But maximally optimizing a model to the point where its inner workings are not accessible or understandable can suffer from a hit to user perception. With regard to explainability, it does not mean the level of personalization cannot be included, with systems having support for relevant recommendations versus users fearing they are being watched or over-personalized.

In this project, the hypothesis is that reviews will provide additional context in the cycle of retrieval and generation with the ability to surface an additional level of traceability in generated responses. This was more of a common sense and ethics driven approach rather than most technically optimal solution that simply caters to emerging user conscious citizenship and the ethics of algorithmic recommendation.

## Summary and Research Gaps

This literature review has examined the historical and cutting edge developments on product recommendation, information retrieval, and retrieval-augmented generation. Existing techniques like collaborative and content-based filtering are historically relevant but rigid and cannot accommodate the semantic richness and complexity needed in transparent, explainable applications. In the present, deep learning and dense retrieval models have (somewhat) reduced these limitations by making automation more distinct from the rationale and doubts of a candidate set, but they still limit transparency and put an emphasis on unseen biases and performance measurements as opposed to the explainability and trust principles of the user.

While there are possibilities with the use of RAG architectures to ground generated outputs and documents to retrieved non-redundant information, the majority of the work to date has focused on the generative aspect of RAG-mainly in open-domain question answering-rather than on retrieval. The few studies on retrieval specifically place retrieval as an immobile stage in search of an auxiliary process with little emphasis on comparative retrieval analysis, and it typically stops short of including comparative retrieval evaluation through close-scored overlap, retrieval algorithm, and recommender system with context of retrieval as a delimiter.

Similarly, there is a strong lack of empirical work evaluating:

How different similarity functions (e.g., cosine, inner product, L2) compare with quality of retrieval

How hybrid retrieval pipelines perform in RAG systems

How correlation studies can provide a behavioural comparison of models beyond traditional evaluations

* How explainability can illuminate issues of system transparency and user trust within downstream evaluation of the recommendation workflow.

This project attempts to rectify these issues by redefining the retrieval stage of a RAG pipeline as an experimental variable rather than as static utilitarian. It offers both empirical contrasts and methodological extensions, such as correlation-based evaluation and hybrid re-ranking approaches which makes a contribution to retrieval science, as well as has the a secondary motive of producing a transparent recommendation system that aligns closely with user goals.

# Methodology

## Overview of the Methodological Approach

This project uses a retrieval-centered experimental methodology to investigate the effectiveness various retrieval configurations provide in a Retrieval-Augmented Generation (RAG) system for recommending products. Unlike typical RAG pipelines with an emphasis on generating output, this methodology shifts the focus towards systematic analysis of the retrieval stage whilst utilising both semantic embeddings along with lexical filtering and similarity/scoring techniques.

The project evaluated semantic only and hybrid lexical-semantic retrieval pipelines. The evaluations controlled for various similarity functions (i.e., cosine similarity, inner-product, and L2-distance) to measure the impact of the functions on early-ranking effectiveness. The project used two different embedding models to assess the impact of the model architecture and training objective on retrieval behaviour.

A series of automated experiments were performed using a labelled query-product dataset taken from the ESCI collection from Amazon (Reddy et al., 2022). Performance was measured using early-ranking metrics, such as Precision@K, Mean Reciprocal Rank (MRR), and Recall@K, as well as conducting a new layer of diagnostic evaluation using Spearman rank correlation to determine alignment in the scoring behaviour of the models.

The design, execution and evaluation stages were structured using open-source tools and reproducible pipelines, including MLflow for tracking experimentation, the sentence-transformer libraries for generating embeddings, and a vector database for indexing and querying retrieval.

This approach was taken to allow for both quantitative performance comparisons, as well as qualitative understandings of the behaviour of models, thus allowing to respond to the research questions both with traditional IR evaluation and as an outlook on experimental design.

## Project Design

This project was designed based on a modular retrieval pipeline that aimed to understand how different retrieval strategies and similarity functions behave in a Retrieval-Augmented Generation (RAG) context to product recommendations. This design intentionally separates and tests independently the key components of retrieval behaviour to elicit performance metrics and diagnostic insights.

The system consists of five phases:

### 4.2.1 Input Layer – Query Handling

Query input is derived from the ESCI-small dataset (Reddy et al., 2022) that was pre-processed and cleaned further in this project via language filtering, grouping, and stratified sampling (see Section 4.3). Each query represents a short, intent-rich phrase (e.g., "face cream for dry skin") that has a uniquely known ground truth product category label.

During runtime, users can enter free, unscripted queries that are encoded in the same way as the product descriptions were embedded, by the same sentence-transformer model, which guarantees they are in a comparable semantic space. The runtime-time generated query embedding is compared to the product embeddings that were precomputed and stored in ChromaDB. The top K product matches are then returned with regard to vector similarity according to the configured scoring function.

### 4.2.2 Retrieval Layer – Semantic and Hybrid Pipelines

Two retrieval pipelines are implemented:

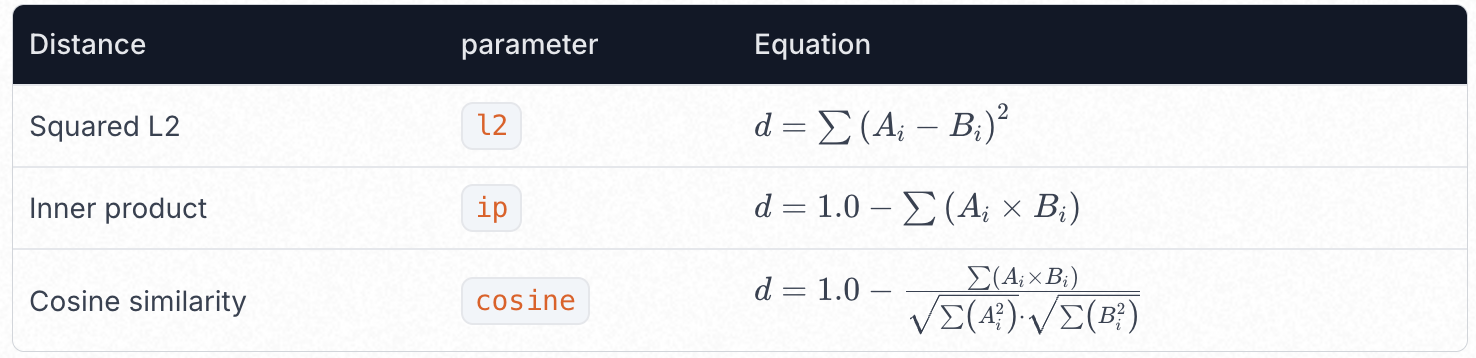
* Semantic Retrieval gets the top K results based on the vector similarity of the query and product embeddings with no lexical filtering.
* Hybrid Retrieval first applies lexical filtering with the Whoosh search engine to state-space reduce the candidates through keyword-based filtering (on the product title and description) and then re-ranks the filtered candidates semantically with relation to vector similarity to the encoded query.

This dual pipeline structure allows for comparison between pure dense retrieval and more tuned hybrid approaches such as those aligned to typical e-commerce search engines.

### 4.2.3 Scoring Layer – Similarity Function Configuration

To assess the influence of similarity scoring on retrieval behaviour, three **ChromaDB collections** were created, each configured with a different ‘hnsw:space’ value:

* cosine (Cosine similarity)
* ip (Inner product)
* l2 (Squared L2)



**Figure 4‑1 Distance metrics and their parameter names in Chroma**(Source:Chroma,2024)

Each product description was embedded once and stored identically across these collections. At query time, the same query embedding is compared within each space independently, isolating the effect of the **similarity function alone**.

### 4.2.4 Evaluation and Logging Layer

Each retrieval result is assessed on:

* Precision@K
* Mean Reciprocal Rank (MRR)
* Recall@K

Moreover, Spearman rank correlation is computed between rank outputs across pairs of models to metrics. This facilitates deeper comparison between models on a behavioural level beyond standard metrics.

Every experiment is tracked using MLflow, including dataset variant, embedding model, similarity function, as well as retrieval type. This offers complete traceability and reproducibility of results.

### 4.2.5 Generation Layer

An LLM-based generation layer (LLaMA 2 through Ollama) was added for demonstration purposes to generate human-readable summaries from extracted product descriptions. This allows for future extension towards explainable product recommendation, though the generation aspect was not tested in this study.

## Data Collection and Preprocessing

### Query Data – Source, Filtering, and Splitting

The original queries for this project were derived from the ESCI-small dataset, which is curated in the BLAIR benchmark (Hou et al., 2023), from the Amazon product search task outlined by Reddy et al. (2022). These queries are real-world, user-inputted terms like “face cream for oily skin” or “long-lasting lipstick” and have high-confidence “Exact” ground truth label.

### 4.3.2 Product Metadata – Cleaning and Category Completion

Metadata for product was derived from the Amazon Reviews 2023 dataset and was first associated through item IDs (parent\_asin). A specialized cleaning pipeline was applied to process prominent product fields before embedding:

* **Fields utilized:** Title and description alone were used for semantic embedding.
* **Removal of HTML:** Text was parsed using BeautifulSoup to remove HTML tags and escape characters.
* **Lowercasing and handling missing values** were used to make the data uniform and stable in terms of vectorization.
* **Deduplication:** Duplicate records were excluded to prevent retrieval biases or training leakage.

For purposes of diversification across categories, the sample metadata was cross-checked with all product categories in the complete set. In instances where certain categories were not adequately represented in the sample set, more products were added by hand to ensure that all areas were covered.

The product ID, cleaned description, and category in final metadata were exported to final\_meta\_sample.json for embedding and retrieval indexing.

### 4.3.3 Dataset Integrity and Ethics

All data for use in this study is from open-access data available for academic research purposes. No personal data is contained in either queries or metadata. Queries are synthetic models of user search patterns, and product data refers to public-access commercial listings. Therefore, there was no need for ethics approval. By creating a language-filtered, balanced within categories, and ground-truth-anchored dataset, this project provided a solid, reproducible basis for carrying out retrieval experiments in a controlled environment.

## Choice of Retrieval Methods

This work was presenting two major retrieval techniques and checking how the choice of retrieval method can affect the performance of retrieval-augmented recommendation systems: semantic retrieval and hybrid lexical-semantic retrieval.

Based on the need of the experiment and the configuration of a real-world search engine, these methods have been chosen, so that deep semantic understanding could be compared to structured keyword filtering.

### Semantic Retrieval

In the semantic retrieval approach, product metadata is being encoded with the help of sentence-transformer models and subsequently uploaded to ChromaDB. For every incoming user query at runtime, the embedding is carried out explicitly by the sentence-transformer model (SentenceTransformer.encode()) to make sure that the vector spaces of query and content can be aligned.

The vector representing the query is matched through three similarity functions (cosine, inner product, and L2 distance) in comparison with the pre-embedded product descriptions stored in the ChromaDB collections as the source of the vectors. This allows the user's language to be matched with the catalog content directly without reliance on keyword matching.

Semantic retrieval is especially effective for brief, high-intent queries such as "moisturizer for oily skin" or "budget concealer" where traditional lexical systems may fail the relevance due to the effect of paraphrasing or synonymy.

### Hybrid Retrieval

In order to avoid the reliance on dense retrieval alone, lexical search precision was proposed to be paired with other forms of semantic matching at a more semantic level.

The following is the process flow we used for this:

1. Lexical Filtering: The first action of the system is to filter a product metadata (the meta-data application was Whoosh’s MultifieldParser) through the system to find exact keyword matches for Lexical filtering. This results in a set of filtered candidate product IDs.
2. Semantic Retrieval: In parallel with Lexical filtering, we embed and execute the same query in L2 similarity through a dense retrieval model so that semantically-related products are found and filter in.
3. Merging Candidates:Merging the two datasets from the two methods of filtering in order to resolve any metadata issues.
4. Cosine-Based Reranking:Finally, the pool of candidates were Reranked from the combined items in terms of cosine similarity of embeddings between the product embeddings and query embedding in order to produce the first top-K items returned for the end-user to filter by.

This two-layer retrieval system represents a production search system as it currently exists in the real-world: as a first pass, operationally quick Filters on keywords are pulled, demonstrating the need to then perform an in-detail semantic process for Reranking to continue at a more advanced and precise level of return and, which would be the case in many cases where simple matching on keywords would fall short.

### Comparative Rationale

By implementing both retrieval methods means the project can:

* Investigate the differences in early ranking performance between a semantic-only system, and a hybrid system
* Investigate the sensitivity of similarity scores, under lexical control conditions
* Investigate the effects filtering has on the final ranking results, including where there is very sparse ground truth

These retrieval methods will be isolated into separate callable modules. This means that the benchmarking and analysis can be repeated.

By implementing this two-path retrieval strategy means that the project has a valid experimental case to assess retrieval design decisions-not just models-affects the retrieval performance of retrieval-augmented systems, in recommendation contexts.

## Embedding Models and Similarity Scoring

The efficiency of any retrieval-augmented system is highly dependent on the design quality and the behaviour of the foundational embeddings. For this project, two sentence embedding models were selected to encode the product metadata and the user queries, so we can also compare both semantic representation and similarity behaviour, with controlled retrieval contexts.

### Embedding Models Used

The following pretrained models were used to generate dense representations of product descriptions:

* MiniLM (all-MiniLM-L6-v2)

This model from the SentenceTransformers library was developed for semantic search and is quick and lightweight with a low memory footprint. Since the model is commonly used with retrieval tasks, it has been found to perform well for general-purpose vector comparisons.

* QANet-based model from BLAIR

This model has been trained specifically for product search tasks, and therefore, we expect it to be more sensitive to domain-relevant terms and item-to-query relationships. We were curious to evaluate whether embedding models defined to the domain would result in different retrieval behaviour for the same product metadata rather than the more general models.

For each model we embedded the product metadata, which was stored in a ChromaDB collections. When a user submits a query we embed their query using the same model as the indexed collection to ensure the vectors are aligned.

We built the Retrieval class to accept the model on initialization, which will allow for more attention to embedding strategies in our experiments in a modular way.

### Similarity Scoring Functions

For measuring how the choice of similarity metric impacted retrieval results, we created three unique ChromaDB collections per model using:

* Cosine similarity (hnsw:space = cosine)

Calculates angular similarity between vectors. Often used with normalized embeddings.

* Inner product (hnsw:space = ip)

Calculates direct vector projection: identifies direction and magnitude.

* L2 distance (hnsw:space = l2)

Calculates encapsulated distance between two points in Euclidean space. Common in nearest-neighbor search applications.

Each collection drew from identical product embeddings and differed only with respect to the similarity function selected as part of the indexing and querying processes. As such, this design allowed for isolated measures of how each scoring method impacted:

* The ordering of the retrieved products
* Precision, and MRR metrics
* The alignment of retrieval between models

### Design Justification

Utilizing multiple embedding models and similarity functions will permit the project to:

* Contrast retrieval behaviour across models with different training objectives
* Investigate the positional quality of early rankings by distance metrics
* Examine robustness and consistency of retrieval performance

This design also allows for correlation-based analysis, as Spearman rank correlation can be leveraged against score distributions across various model–metric pairs. This would further clarify whether systems producing similar metrics also rank products similarly - or whether the performance hides internal behaviour differences.

## Evaluation Strategy

Assessing retrieval performance in a RAG system is essential in order to understand how well relevant information retrieved for generator purposes. The main focus of this project is building out the retrieval component successfully for product recommendation, so we assess early-ranking performance - and how fast and accurate the correct predicted product is retrieved.

The assessment encapsulates conventional IR metrics and a diagnostics correlation analysis which will provide a well-rounded sense of performance and internal behaviour of the system.

### Standard Retrieval Metrics

There are three ranking-based metrics which were used throughout the project:

* Precision@K

Measures the ratio of relevant results to all results based on only the top-K items returned. Precision@K is important for recommendation systems since users are only provided with a limited amount of results returned.

* Mean Reciprocal Rank (MRR)

Measures the rank of first relevant item in the ranked list by averaging the reciprocal rank across all queries. MRR is effective at measuring early relevance, or how fast a relevant result appears.

* Recall@K

Measures the ratio of relevant items retrieved to all known relevant items. Recall@K is useful, but is limited if the product with respect to a query only maps to one example in very sparse-label scenarios.

The metrics above were computed via a structured ground truth mapping where every query mapped to one or more product IDs (see Section 4.3). For sampled query sets, measures were logged by different similarity functions and embedding models to afford side-by-side comparison.

### Diagnostic Analysis - Spearman Rank Correlation

A similar diagnostic analysis was conducted earlier to further confirm retrieval behaviour consistency across configurations beyond the metric-based evaluations above and included:

* Similarity scores representing the encoded query with all retrieved items from the two models
* Then computing the Spearman rank correlation (ρ) for these score vectors

Spearman rank correlation is a method to understand if two systems rank products in the same order based on the scores they compute no matter their score or what the final top-K output is. This is useful when:

* Both models produce similar MRR or Precision@K values together but differ internally
* There is a need to understand behavioural redundancy between systems or divergence

This diagnostic analysis supports increased granularity into retrieval quality, and allows further insight beyond predictive metrics alone.

### Evaluation Pipeline and Tools

For reproducibility and for tracking experiments:

* All evaluation run information was tracked with MLflow: query ID, embedding model, retrieval strategy, similarity function, and reported metrics
* Evaluation results were stored per query allowing aggregation and per-query evaluations
* The evaluation pipeline supports automated loop over any whole query set and configurations, which allows scalability.

Together these tools made rigorous, scalable comparison across different variables at scale possible while maintaining experiment traceability.

## Testing and Results Pipeline

To realize the goals of automated testing and giving a repeatable and scalable evaluation of the retrieval configurations, an automated testing and logging pipeline based on Python and MLflow was designed. The automated testing pipeline was designed to loop over a set of ground-truth query-product pairs, and was capable of calculating various ranking-based metrics based on different retrieval settings.

### Execution Process

To execute the evaluation process, we first read a properly-structured CSV file wherein, the query-product mapping was filtered and pre-processed (see Section 4.3). The evaluation process only used queries that were annotated as part of a training set (split="train") for the calculation of the metrics; this was to ensure a consistent evaluation scope each time.

For each query we followed these steps:

1.Query Embedding: The input query was encoded, utilizing the chosen sentence-transformer model (e.g., all-MiniLM-L6-v2).

2.Data Retrieval: The query embedding was sent to the semantic retrieval module (Retrieval class) that harvests the top-k from the relevant ChromaDB collection, and the configured similarity metric (e.g., L2).

3.Evaluation: The returned product IDs from the previous retrieval, were subsequently compared against the known-ground-truth. Because of the assumed ranking problem for the returned product IDs we calculated Precision@K, Recall@K, and MRR.

4. MLflow Logging: We logged per-query metrics, query length, model name, total items retrieved, and configuration parameters (similarity type, K-value).

This approach is designed to facilitate independent evaluations of each query and independent evaluations of the fine-grained and per-query logs that could also be pulled together (and analyzed in a more full context) later.

### Mean and Summary Metrics

As well as being able to evaluate metrics per query, this process calculates aggregate values for all evaluated queries, including:

* Mean Precision@K
* Mean Recall@K
* Mean MRR

These mean values are also logged in MLflow and will serve as the basis for making model-to-model and configuration-to-configuration comparisons across experiments.

### Configuration Flexibility

The process allows easy changes to the following:

* Embedding model (model\_name)
* Similarity metric: Metric.L2, Metric.COSINE, etc.
* Number of retrieved items (kr)
* Evaluation cut-off (k)

The process allows distance between evaluations of these parameters through a simple variation of each from experiment to experiment. This simplifies a systematic sweep of experimental parameters, returning more robust conclusions from the metric output observation.

### Diagnostic Usefulness

Query lengths are passed through in the logs for correlation with retrieval performance. As no real conclusions were drawn from this data with uniformly short queries, it provides opportunities for more future experiments about query complexity sensitivity.

## Results Validation

There are different ways to check both the integrity and validity of the results developed with retrieval evaluations. In this project, multiple validation strategies were applied incorporating both experimental design execution and metric dependability and relevance.

### Ground Truth Alignment

All evaluations were performed against pre-determined query–product pairs from the ESCI-small dataset, we only retained those marked “Exact” (see Section 4.3). This provided a highly confident, unambiguous relevance mapping for evaluation metrics like Precision@K and MRR.

Queries evaluated against the known correct product ID(s), and negative samples were drawn from the same product category to represent real-world distractors; this engaged more reality into our evaluation, making the ranking results stronger.

### Per-Query Metric Tracking

Queries were evaluated one at a time and the metrics were tracked at the query level using MLflow. This level of tracking provided the following capabilities for the project:

* Post-hoc inspection of any unexpected patterns in performance
* Visualizations of how metrics differ across query types or lengths
* Ensuring metric consistency for moves made in parameter tuning.

By tracking metrics per query and per run, the workflow offered ongoing traceability and reproducibility of metrics.

### Averaging across Queries

We evaluated our metrics across 200 queries; every query was a real query containing a single product ID, as opposed to deriving a few metrics based on a small number of queries or selected examples. We averaged the metrics such as Precision@K and MRR across the entire set of products; results from these evaluations are statistically stable, and it avoids outlier data bias.

### Diagnostic Validation By Correlation

In addition to aggregate metrics, the project also employed Spearman rank correlation on the similarity scores generated by different models or scoring configurations. This validated not just what was returned, but how consistently the results were ranked across different systems.

For example, two models could produce the same Precision@3 for some queries, but rank the items for those queries in entirely different orders. Correlation analysis could help us to identify those systems as being either redundant or behaviourally distinct, further exposing the interpretive strengths of our results.

### Reproducibility and Experiment Tracking

All whatever evaluations were:

* logged through MLflow
* parameterized for controlled experimentation
* run through scripts that loaded a fixed query–product dataset

This establishes that results are fully reproducible, including with different similarity functions, embedding models, and evaluation K-values. Hyperparameters e.g. embedding model names (x2), query lengths (x2), run configurations (e.g. K, algorithm, embedding model, etc) were also logged for entire transparency.

## Tools and Technologies Used

This project was built in a modular, Python-based environment and was focused on the use of open-source, well-documented, and community-supported tooling. The system architecture was designed to facilitate flexible experimentation of embedding models, retrieval configurations, and evaluation strategies, and, the tooling that were selected reflect those design goals.

For indexing and retrieving dense product embeddings, ChromaDB served as the primary vector store. Its support for multiple similarity metrics through the hnsw:space configuration to create separate collections based on cosine similarity, inner product and L2 distance afforded precision and isolation when comparing entities across similarity spaces. ChromaDB's Python API integrated simply into the retrieval pipeline, facilitating switching experimental settings and guaranteeing ease of use.

SentenceTransformers was used to embed both product descriptions and user queries. Two pretrained models were utilized during experimentation, all-MiniLM-L6-v2 to embed general context as semantic encodings and multi-qa-mpnet-base-dot-v1 that provided enhanced matching by being query-aware. The SentenceTransformers library offered uncomplicated heat mapping capabilities through its easy interface to generate sentence-level embeddings, which was also a critical aspect of maintaining a consistent vector space across the retrieval and evaluation modules.

As the hybrid part of the retrieval, the system also included Whoosh, "an effective and efficient, lightweight, and free (open source) full-text search engine in pure-Python." Whoosh was utilized to conduct lexical filtering of product metadata through multi-field keyword queries, eliminating candidate items before reranking and multi-match generation. Its simplicity of installation and minimal requirements helped Whoosh provide an immediate capability for embedding within the hybrid retrieval pipeline without additional search infrastructure.

MLflow was used extensively for recording metrics, model parameters, and evaluations. Every run of an experiment recorded important variables, including the retrieval configuration used, embedding model, number of results retrieved, query length, and metrics computed such as Precision@K and MRR. This represented a useful way to record experiments and compare results across experiments, and it maintained transparency and reproducibility.

Other libraries, such as Pandas, BeautifulSoup, and Langdetect were used to prepare data. Pandas was great for filtering, merging, and exporting datasets. BeautifulSoup was handy for cleaning product descriptions of HTML artifacts. Langdetect was used to normalize language by removing non-English queries in preprocessing.

The development environment was simple - using Python 3.10+ via virtual environments, pip for managing dependencies, and development was done using Jupyter Notebooks for exploratory work and PyCharm or VS Code for module development. All experiments were run locally, cached the embeddings (to save resources), and stored logs in a systematic way for reproducibility.

Together these provide a lightweight, yet robust framework for experimentation, evaluation, and analytics, permitting iteration at great speed (while maintaining the integrity and traceability of results).

## Ethical, Legal, and Social Considerations

With the nature of this project drawing on real users' queries and product metadata, ethical, legal, and social implications were addressed throughout project development and experimentation. Our consideration focused primarily on data usage, upholding data privacy, and attempting to reduce bias or possible harm with the recommended output.

The data that was collected for this project originated from open datasets for academic research. The queries came from the ESCI-small dataset released with the Amazon product search benchmarks, and the relevant product information originated from the Amazon Reviews (2023) dataset, both of which were located on HuggingFace and Github with permissive usage, meaning the datasets can be used freely.

The datasets do not contain personal identifiers, user profiles, or tracking data, which are completely anonymized. Therefore, in-line with standard data uses we are able to mitigate risk regarding data privacy, meaning formal ethics approval was not necessary.

We filtered for English-language content during preprocessing using automated language detection. As this interest in language is for consistency and interpretability, and it is a limitation that limits linguistic inclusivity, we accept that we may have filtered out valid queries in other languages. Although within the purview of this project and the community iteration engaged in during consultation, we recognize that in other contexts, this limitation may affect generalizability especially for deployment in a multilingual or internationally based system.

Further, product recommendations were solely based on semantic similarity between the user's query and product descriptions or reviews. This support of recommendation does not eliminate outright biased outputs, but rather reduces the probability, since biases may still be present in the training data, or the embedding models themselves. For example, underrepresented items might favour more popular or descriptively rich products if there is insufficient metadata for embedding quality to be strong. While this would not create risk, and is acceptable in a contained research project, if a deployed system, this could lead to some element of fairness issue later on especially if high-stakes contexts involved as in healthcare or financing.

From a social perspective, the system design sought to provide the explanation for recommendations by anchoring outputs back to metadata as a form of evidence. This is an important approach to support explainability as it is becoming urgent, as users are increasingly aware of how AI systems affect their choices, in ways that they may not understand or which they do not trust. While we did not test this at scale in this project, this is similar to setting an ethical design principle in terms of building user trust, algorithmic accountability, and informing a user-centered personalization process.

In summary, the project made every reasonable attempt to be ethically sensitive to data usage norms, to observe ethical practices, to encourage individual interpretation of limitations around the language scope and models' potential biases, and create confidence in the transparency and traceability of system outputs through evaluative approaches and logging. These represent reasonable practices in responsible AI development, and provides a foundation for thinking about refinement and scaling in ethically complex contexts.

## Practical Constraints and Limitations

Though the project was designed with many deliberations to create a controlled experiment in an effort to conduct modular testing, there were more practical constraints and contexts in the way that hindered the thus-completed, the generalizability of the findings, and identified constraints to the scope of the term.

The primary limitation was the availability and sparsity of ground truth labels in the ESCI dataset. Many queries are mapped to a only a single exact product, limiting the intention to measure recall, and assess the diversity of correct retrieval. The inability to support evaluation of ranking metrics such as Recall@K, are most useful with multi-relevant-item data. Therefore, Precision@K and MRR were appropriate measures for this use case.

Another limitation was in both the data size and sampling strategy. The outcome was limited to the sampling of 400 queries (200 train + 200 test) for evaluation because of a focus on modular experimentation and having a number of repeated metrics being logged each time unchecked configuration changes were made. Thus, the 400 queries used for evaluation purposes allowed for looping efficiency and fine timing metrics across configurations and keep cost-sliding metrics separate, but again, if we have only 400 queries ,it is going to be limited when staking broader claims to statistical power or robustness confidence. That said, the sampled queries were sampled using a form of stratification logic and validated to ensure all product category types were included so they were representative with respect to testing.

From a systems architecture perspective, the project operated in local, non-distributed, environments, and scalability was never fully realized. Embeddings were generated and stored on local storage, and vector retrieval happened using a single-node ChromaDB instance. This type of routing was effective for research-scale sized data but didn't consider the configuration issues or latency for real-time, production-scale system deployments.

Another limitation is that ,Generative module was implemented via an existing LLaMA-based generative capture module, however, it was never evaluated rigorously. This project focused only on retrieval performance, thus we did not evaluate the quality, or fluency, or factual consistency, of product summaries generated content.

In terms of model variation, we were only able to compare two embedding models for the Capstone. These two models represented a general-purpose embedding strategy (MiniLM) and a task-specific embeddings (QANet). Comparing the two models as two examples of model embedding behaviour could have rich insights on redundancy or robustness of embeddings with a bigger and wider range of models.

Finally, time constraints through the project timeline, prohibited user-centered or qualitative validation, such as listening to employee perspectives on explainability, or human satisfaction with ranked products outputted. While the pipeline has built-user feedback mechanisms (such as collecting employees feedback from the cited qualitative capture study) these components of the work were out of the current work's scope and timeline.

Notwithstanding these limitations, the project provided an overall configurable, reproducible, and well-documented evaluation of retrieval strategies within the context of RAG-based recommendation, and laid suitable groundwork for scaling, extension, and user-facing refinements in future iterations.

# QUALITY AND RESULTS

## Evaluation Setup

We conducted the evaluation using our ESCI-small dataset (Reddy et al., 2022); we filtered this dataset to retain only those queries labeled “Exact” and originating from an English-language query. We grouped queries by uniqueness and then split the queries into training and test sets, with 200 queries in each group. Mappings for ground truth were aligned to that of product metadata from Amazon Reviews 2023.

Queries and product descriptions underwent preprocessing: detection of original language and filtering in the case of queries, while product descriptions were stripped of any html tags that were lower cased and deduplicated. Only product descriptions underwent embedding, of which query embedding was done at runtime relative to the product using the same model for consistency.

The following tools were utilized:

* ChromaDB for vector searching (Cosine, L2, Inner Product)
* Whoosh for Lexical filtering in hybrid retrieval
* MLflow for experiment tracking
* Sentence Transformers (MiniLM, QANet) for embedding

We tracked our evaluation metrics (Precision@K, Recall@K, MRR) across the different configuration of the two retrieval strategies.

## Retrieval Strategy Comparison

We compared the following two retrieval strategies:

* Semantic Retrieval: This strategy consisted solely of semantic searching based on SentenceTransformer embeddings, which were stored in ChromaDB.
* Hybrid Retrieval: This strategy combines lexical filtering (Whoosh) and a semantic retrieval, followed by re-ranking based on cosine similarities.

For both MiniLM and QANet, hybrid retrieval produced better results than semantic at low K values (especially for K=3, and K=5). For example with MiniLM : L2 similarity, Precision@3 increased from 0.6349 (semantic) to 0.6557 (hybrid). The recall was comparable suggesting hybrid retrieval improves precision without sacrificing coverage.

This pattern seems to align with the idea that lexical filtering functions to remove noise from dense retrieval methods, particularly in the case of loosely relevant items due to phenomena such as semantic drift from the embeddings alone. The hybrid retrieval results were especially advantageous for shorter queries that were highly relevant where overlap in the lexical space was potentially a greater phenomena for specifying relevant candidates.

At higher value of K (K=10, for example) the performance gap diminishes between semantic and hybrid. While hybrid is helpful in retrieving the top-3 or top-5 candidates, semantic retrieval can be sufficient at larger K values as it captures wider coverage

## Similarity Function

To analyze the affect of the similarity scoring, we configured each ChromaDB collection with three different distance metrics:

* Cosine Similarity
* Inner Product (IP)
* L2 Distance

And we evaluated these collections under both semantic and hybrid pipelines using the MiniLM and QANet.

Key Observations:

* In most settings, there was very little difference in performance across types of similarity.
* However, when instrumented in hybrid retrieval with MiniLM with K=3, L2 distance had the highest Precision@3 (0.6557), which was greater than cosine and IP on this measure by a good margin.
* For semantic search, difference across cosine, IP, and L2 distance, was minimal, especially at K=10 when they were very close.

In summary, while the similarity type may not affect broad recall with the related library, it may affect early precision with filtered sections or hybrid retrieval. These results are consistent with past findings (Karpukhin et al., 2020) which suggest that small differences in similarity scoring can affect the top items retrived - and is more impactful with re-ranking.

## Model Comparison

MiniLM and QANet were modeled and evaluated under identical conditions across all similarity metrics and retrieval strategies. The goal was to identify whether or not the choice of model had an impact on system performance and whether each model had been indexed by contrasting embedding spaces - resulting in contrasting retrieval behaviours.

Key Findings:

* Across each of the semantic and hybrid retrieval pipelines, MiniLM and QANet returned similar results across most configurations.
* For example, with K=3 using L2, MiniLM (Precision@3: 0.6557) and QANet (Precision@3: 0.6322) performed similarly.
* Regardless, MRR values continued to hover close (and in some cases be identical), suggesting that each model retrieves relevant items with the same efficacy.
* The similarities in retrieval scores revealed levels of similarity, however we theorized there would be a redundancy in their retrieval behaviour, which warranted a further inspection by generating similarity scores and doing Spearman rank correlation on the similarities scores from queries (950) to those from their ground truths (7,920 product descriptions).

## Correlational Comparison:

For a diagnostic experiment, we calculated cosine similarity scores between each query and it's ground truth product descriptions, again using Spearman's rank correlation to calculate the agreement between the two models similarity ranks.

The results suggested moderate to high levels of correlation in the vast majority of cases, which confirmed there is a level of correspondence between how similarly our two models are ranking the ground truths, in part explaining the convergence between results reported by retrieval metrics.

Practical Implication:

Selecting the model should not significantly influence the output of retrievals for brief queries based on keywords like those in ESCI but future work could examine how these models differ in retrievals that are more complex or of a conversational nature.

## Interpretation and Takeaways

The evaluation experiments indicated some interesting takeaways about the behaviour and performance of the different retrieval configurations of the RAG system:

### Best-Performing Combinations

For semantic retrieval methods, the best-performing configuration was K=3 for the MiniLM model using L2 similarity scores. This configuration produced the highest average precision and a strong MRR score, indicating that it was effective at pulling relevant product ids early in the ranked listing.

Hybrid retrieval consistently showed advantages in early precision with both embedding models by combining lexical filtering with semantic retrieval (i.e., using Whoosh to filter). This suggests using keyword-based filtering helps with early ranking quality by removing candidates.

### Similarity Function Impact

L2 similarity outperformed cosine and inner product scoring at lower K values consistently, particularly in the hybrid retrieval situation. While performance differences in metrics decreased as K values increased. This behaviour reiterates that early ranking metrics, such as Precision@3 and MRR, are considered to be more sensitive to the choice of similarity function, also established in the literature.

### Embedding Model Behaviour

MiniLM and QANet retrieval quality was comparable across all similarity functions and metrics despite their structural differences. The convergence of performance demonstrates that both models operate similarly in respect to capturing semantic relevance, for the current domain and dataset.

To further verify this Spearman rank correlation was calculated for the two models similarity scores to the same set of queries and products. The strong correlation values verified the two models have analogous ranking characteristics, especially when referring to queries with a single ground-truth product.

### Latency versus Accuracy Tradeoff

While hybrid retrieval improved ranking accuracy slightly, it also added latency due to the additional filtering step. When it is essential to have a real time response, pure semantic retrieval still is a reasonable alternative because it has lower latency and equal MRR and recall.

### Query Complexity

A preliminary analysis of query length was conducted in order to understand the possible effects of query length on retrieval quality. However, because all of the queries in the ESCI dataset were shorter in length, we did not find evidence to suggest a statistically significant relationship. Future work may want to use a dataset with a greater range of query complexity to explore this relationship.

## 5.7 Summary

The chapter provided a detailed assessment of the retrieval options employed in the RAG-based product recommendation system. Using a comparative approach, a semantic retrieval, a hybrid retrieval, and a fusion-based retrieval, the evaluation provided the following conclusion:

* Best Performing Configuration: MiniLM with the L2 similarity using K=3 had the highest average precision value in semantic mode, and hybrid mode, and demonstrated strong MRR scores to support quality of early ranking.
* Hybrid Retrieval Advantage: Hybrid retrieval methods had improved early precision consistently when compared to semantic retrieval methods - suggesting the value of the use of lexical filtering.
* Similarity Scoring Impact: L2 distance was better than cosine and inner product methods, and although the results converged at larger K values, L2 provided a better distance solution at smaller K.
* Model Comparison: The retrieval performance was similar for both MiniLM and QANet across all experimental configurations. We verified that they provided the same ranking by performing a spearman correlation on the retrieval results.
* Fusion Strategy - Alpha Tuning: The fusion score created from an alpha weighted combination of lexical and semantic similarities demonstrated marginal but meaningful improvement using alpha ≈ 0.45 performing best.
* Query Length Effect: We found no differences in retrieval performance based on the length of the query level. Given the ESCI query dataset had all short queries, this finding is expected.

These results explicitly confirm the objectives of the project and prove that retrieval configuration, particularly similarity scoring and hybrid reranking, is instrumental in optimizing RAG systems for product recommendation.

# Evaluation and Conclusion

## Final Evaluation

This project endeavored to examine the effects of retrieval strategies-those being semantic, hybrid, and fusion-based-within an Retrieval-Augmented Generation (RAG) system designed for product recommendation. The main objectives were achieved: retrieval mechanisms were implemented, contrasted across similarity functions (cosine, inner product, L2), and tested by means of early ranking metrics including Precision@K, Recall@K, and MRR.

The findings verified that retrieval quality is the determining factor for generation relevance, especially for short-query e-commerce. Hybrid retrieval brought modest gains in initial precision, while fusion-based reranking (with alpha tuning) brought fine-grained control over ranking quality. While at times modest, the gains were uniform over K values as well as over embedding models, and showed that retrieval pipelines could indeed be optimized separately from the generation layer.

Despite constraints with time and resources, the overall approach was realistic and produced a functional end-to-end system, with standardized evaluation using a sample subset of the curated Amazon ESCI dataset. The automated evaluation loop and MLflow tracking enabled repeatability and allow viewed metrics to be delved into across model runs.

## Project Management

The project underwent an iterative and modular approach to project management and resourcing. Originally the timelines shifted on the retrieval design and presumably embedding integration, with subsequent stages in evaluation automation and advanced experimentation - for example, alpha tuning and correlation analysis. All major tasks were completed on time, although some delays were experienced due to expanded scope adjustments, such as adding some comparative retrieval methods and adding a similarity diagnostic to be looking for value as summaries and added more depth to the analysis in the report.

Utilizing a version-controlled code repository, combined with utility classes designed to be modular, and MLflow as a means of ensuring a reproducible approach for tracking experiments explicitly, also contributed to ensuring monitoring was efficient while also allowing visualization of our results. Proposed project risk such as data volume, overhead involved for models to estimate similarity, and using an evaluation framework were addressed by sampling early to project potential runtime and user burden, awareness of preprocessing models employed, and utilized lightweight models such as MiniLM.

## Insights Gained

The primary technical lessons included:

* Sensitivity of Similarity Function: Retrieval performance was sensitive to similarity function choice, particularly in early ranks (e.g., K=3). Even in hybrid setting, L2 generally outperformed cosine and IP.
* Lexical-Semantic Fusion: Adding in reranking to combine lexical and semantic techniques adds a measurable increase in precision/recall gains for limited architectural complexity.
* Model Interchangeability: While it may be evidence of proficiency, the embedding models applied (MiniLM and QANet) produced retrievals that acted similarly. The earlier Spearman correlation analysis was helpful to quantify these similarities.

Perhaps of equal importance in the overall process was understanding and reinforcing the role of automating evaluation, explainability and prompt explainability in AI systems, and the influence of modular experimentation pipelines in creating reproducible research.

## Comparison to Literature

Most studies using RAG approaches consider only the quality of the generations (Lewis et al., 2020; Izacard and Grave, 2021), whereas this project relates to a growing line of work treating retrieval as a tunable component (Karpukhin et al., 2020). The use of lexical filters matches the methods define in traditional IR approaches (Robertson, 2009) while the alpha-weighted reranking compared to multi-source fusion methods in information retrieval.

Unlike the studies that evaluate against large-scale QA datasets (e.g., Natural Questions), this work uses retrieval-based RAG for e-commerce, a domain that is not well represented. The alignment with ESCI dataset practices additionally solidifies this project within real-world recommendation-setting-like evaluation practices.

## Challenges Reflection

Several issues informed the design approach the project:

* Data Sparsity: Many of the queries had one ground-truth product, meaning that a limited variety was possible in the evaluation. This was slightly mediated through an increased test sample size and using multiple metrics, such as MRR.
* Query Complexity: All of the queries were consistently short, so we also limited the assessment of how metric performance varied with respect to query length.
* Reranking Tradeoffs: Reranking metrics were improved using hybrid methods and fusion methods. However, the additional latency and complexity would be worth considering in the context of deployment.
* Interpretability: There is also the problem of explainability of product suggestions using retrieved reviews, which remains more of a challenge when these are noisy or in contradiction to these.

These issues will be resolved with explicit dataset considerations, reranking design attenuated towards modular design, and evaluated in transparent ways.

## Future Work

There is a great deal of opportunity for future work:

* Scaling Corpus Size: Evaluating the method on a full-scale product corpus, segmented by category may give insights into improved generalizability.
* Adaptive Alpha Tuning: Research into alpha for query sensitivity could be addressed through reinforcement or learning-to-rank models.
* Integrating within Generative Agents: The method could be embedded within a conversational agent to evaluate the entire end-to-end explainability of and user experience (uX).
* Multi-lingual Retrieval: Growing the domain of recommendations using multi-lingual transformers could broaden the impact of generalizability across different markets.

These means of future work could help to elevate the academic opportunity and commercial direction of RAG based product recommendation systems.

## Conclusion

This project provided a retrieval-centric exploration of product recommendation using a RAG architecture. After diligent implementation of various retrieval systems based on semantic, hybrid, and fusion retrieval - and evaluation of these retrieval systems with solid early ranking metrics - it was shown that retrieval decisions are fundamental to recommendation quality.

The project achieved its objectives in delivering the following:

* a functional retrieval pipeline with a range of similarity functions.
* an automated evaluation suite with integration into MLflow.
* experimental insight into model behaviour, reranking, and metric sensitivity.

While it was limited by scale, the project provided a methodologically, practically, and academically relevant exploration of the retrieval stage of RAG systems. The findings yield actionable design decisions for researchers and practitioners seeking to optimize recommendations in low context query settings.