HEC MONTRÉAL

Comparative Performance Analysis of Retrieval Models on Textual Data with Diverse Linguistic Profiles

by

Gabriel Jobert

Under the supervision of
Eva Portelance

Department of Decision Sciences
(Data science)

In Partial fulfillment of the requirements for
the Degree of Master of Science (M.Sc.)

August 2025

Abstract

Retrieval-augmented generation (RAG) systems rely on high-quality information retrieval components to supply relevant evidence to language models. This thesis conducts a comparative evaluation of four retrieval models – the classical sparse lexical model BM25 and three neural retrievers (Dense Passage Retriever, Contriever, and ColBERT) – across five question answering datasets spanning diverse domains and query types (MS MARCO V2, SQuAD 2.0, ObliQA, CoQA, and HotpotQA). We assess each retriever's ability to return correct answer-supporting passages and analyze how the models' performance is affected by differences in the datasets' linguistic profiles. We further observe that complex information needs – such as multi-hop questions (ObliQA, HotpotQA) requiring reasoning over multiple documents and conversational queries (as in CoQA) that involve dialogue context – present additional challenges for all models. These findings highlight that the linguistic profiles and domain of the dataset significantly influence retrieval model performance. Overall, this study provides practical insights for the design of RAG systems: adapting the retriever in function of the textual characteristics and target domain can substantially improve the retrieval of relevant evidence, thereby enhancing the accuracy and reliability of downstream generative models.

Keywords: Information Retrieval; Retrievers; Question Answering; Semantic, Linguistic.

Research methods: Empirical Comparison; Quantitative Analysis.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my thesis supervisor, Professor Eva Portelance, for her support, valuable advice, and encouragement throughout the research and writing of this thesis. Her expertise and insight played a major role in the development of this work. I am also grateful to the faculty and staff of the Department of Decision Sciences at HEC Montréal for providing me with a stimulating academic environment. Special thanks go to my family and friends for their constant support and patience. Their encouragement has given me the strength to persevere through the challenges. Finally, I would like to thank all the authors and researchers whose work laid the foundations for this thesis.

Use of generative-AI

In the course of this thesis, I made use of generative-AI tools to ensure that the documentation was as complete as possible by identifying relevant papers to read and integrate into the final research. I also employed these tools to refine the English writing of this document, particularly to correct syntax and improve sentence structure. Finally, while developing the programming components, I used them to assist in debugging certain errors that I was unable to resolve independently. This approach is consistent with the requirement that any use of generative-AI in research uphold the highest standards of academic integrity, data protection, and reliability. I personally verified and validated all code, analyses, and written text to ensure that the results are accurate, appropriate, and free from any misuse of intellectual property. In doing so, I adhered to the principles set out in the "Lignes directrices sur l'utilisation de l'intelligence artificielle générative pour la recherche" of HEC Montréal, which emphasize transparency and responsibility while maintaining full accountability for the final content.

Contents

Chapter 1

Introduction

Retrieving relevant information from large collections of text (also known as Information Retrieval (IR)) has been a cornerstone of information systems for decades. The evolution of retrieval models has been central to this progress. Early systems in the 1950s and 1960s used Boolean models, where documents were retrieved based on exact keyword matches using logical operators, but these offered no ranking of results. The 1960s introduced the Vector Space Model, pioneered by Gerard Salton, which used sparse representations—term-document matrices weighted by schemes like TF-IDF—to rank documents based on cosine similarity. Later, probabilistic models such as the Binary Independence Model and BM25 [2] improved retrieval by estimating the likelihood of relevance and ranking accordingly. These models dominated for decades, forming the backbone of traditional IR systems. However, they still relied on sparse representations, where each term was treated as a discrete feature with little understanding of semantics. The 2010s saw a shift toward dense representations enabled by deep learning [9], where words, sentences, and documents are embedded into continuous vector spaces using models like word2vec, BERT, and ColBERT [37, 15]. These dense retrieval models capture semantic relationships and enable matching based on meaning rather than exact terms. While sparse methods remain efficient and interpretable, dense models have significantly advanced retrieval performance, especially in open-domain question answering and conversational systems. Today's best systems often combine both, using hybrid retrieval approaches to leverage the strengths of sparse lexical matching and dense semantic understanding.

In parallel, the rise of large pre-trained language models (LLM) has raised interest in RAG, a paradigm that integrates a retrieval model into the generation pipeline. Rather

than relying solely on parametric knowledge stored in model weights, a RAG system explicitly retrieves external context (e.g. documents or passages) and conditions the generative model on this retrieved evidence when producing an answer or completion [8]. This approach has proven especially important for knowledge-intensive NLP tasks and dialogue systems, allowing models to access up-to-date or domain-specific information at inference time. By augmenting generation with retrieval, RAG addresses several limitations of standalone generative models – for instance, it mitigates factual hallucinations and outdated knowledge issues by grounding outputs in relevant source text, and it provides natural avenues for attribution of facts to sources [8, 7]. The retriever is therefore a crucial component in the RAG pipeline: the quality of what the model can produce is directly tied to the relevance of the documents fetched. Effective retrievers enable the overall system to generate more accurate, specific, and contextually appropriate responses, as demonstrated by Lewis et al. [8] in their seminal RAG framework which combined a neural retriever with a sequence-to-sequence generator and achieved state-of-the-art results on open-domain QA benchmarks.

Modern (dense) retrieval systems can be further categorized by their architecture and training strategies. A fundamental design choice is between bi-encoder (dual-encoder) models versus cross-encoder models. In a bi-encoder architecture, the query and document are encoded independently by two neural networks (often sharing the same parameters) into vector embeddings; retrieval then reduces to a fast similarity search (e.g. inner product or cosine similarity) between the query embedding and a large set of pre-computed document embeddings [9]. This two-tower design is highly efficient at runtime and scales to millions of documents with sub-second latency using approximate nearest neighbor indexing. However, because the query and document representations are learned separately, bi-encoders may not capture all fine-grained interactions between query terms and document content. Cross-encoder models, on the other hand, apply a single transformer-based encoder to the concatenated query–document pair and directly output a relevance score [34]. This one-tower approach allows the model to consider rich cross-attention interactions between the query and a given document, often leading to superior ranking accuracy. For instance, a BERT-based cross-encoder can learn to pick up subtle phrase correspondences and context dependencies that a dual-encoder might miss, yielding better accuracy in ranking [34]. The drawback is that a cross-encoder must process each query–document pair at inference time, making it orders of magnitude slower and impractical for exhaustive search over large corpora. That is not in the scope of this thesis but in practice, a common compromise is a multi-stage retrieval pipeline: a fast bi-encoder first retrieves a small set of candidates, which are then re-ranked by a more expensive cross-encoder to refine the results. Such hy-

brid strategies leverage the strengths of both approaches—efficient broad retrieval followed by fine-grained reranking—to maximize overall performance.

Understanding the behaviors of retrievers is essential for the design of effective retrieval-augmented generation pipelines and motivates a thorough examination of how different retriever strategies perform across various data formats and tasks. The objective of this study is to evaluate how different types of retrieval models perform when exposed to question answering tasks drawn from datasets with distinct textual and structural properties. To this end, I compare multiple retriever architectures —from sparse to dense retrevier models— across a diverse set of Question-Answer (QA) datasets, each dataset representing a unique domain and linguistic profile. Rather than focusing solely on model architecture or answer generation, this work adopts a retrieval-centric perspective. Specifically, I assess the ability of each retriever to correctly identify context passages that support answers to given questions, independent of downstream language model generation.

The guiding hypothesis is that retrieval effectiveness is not uniform across datasets. Their linguistic characteristics may interact with the inductive biases of sparse or dense retrievers in distinct ways. By systematically applying each model to all datasets, analyzing their performance and the nature of the passages they select, I aim to uncover how retrieval mechanisms are influenced by linguistic features.

My research demonstrates that retrieval model effectiveness is not uniform across datasets; instead, it depends strongly on the linguistic profiles and domain of the textual data. The evaluation confirms that lexical and neural retrieval techniques have complementary strengths: dense semantic retrievers excel in certain scenarios by identifying relevant passages through meaning similarity, whereas sparse lexical methods like BM25 remain superior in cases that demand exact keyword matching. This outcome validates the central hypothesis that differences in linguistic and domain attributes of the dataset can significantly influence retriever performance, underscoring the need to align the choice of retrieval approach with the nature of the data.

The key findings of this research are as follows. First, the neural retrievers generally outperform BM25 on open-domain and general knowledge queries (e.g., MS MARCO V2 and SQuAD 2.0), thanks to their ability to capture paraphrased or synonymous content beyond exact term overlap. Second, on a highly specialized dataset with domain-specific jargon (a regulatory compliance QA setting represented by ObliQA), the traditional BM25 significantly outperformed the standard dense retrievers, underscoring the importance of

exact term matching for rare technical terms; however, the advanced late-interaction model ColBERT was able to achieve even higher accuracy in that same domain by leveraging fine-grained token-level interactions (at the cost of greater computational overhead). Third, the unsupervised Contriever model proved remarkably robust across different domains, exceeding the supervised Dense Passage Retriever – the latter tended to underperform on these varied tasks, suggesting that without targeted domain training, a supervised dense retriever may not generalize well. Finally, all of the models struggled on questions that require multiple pieces of evidence or conversational understanding: multi-hop queries (such as those in HotpotQA) and conversational QA dialogues (like CoQA) posed significant challenges, indicating that complex reasoning and context integration remain difficult for current retrieval methods.

The structure of the thesis is as follows: Chapter 2 reviews the foundational literature on retrieval models and datasets, and presents the experimental design. Chapter 3 analyzes the results of my comparative evaluation. Finally, Chapter 4 concludes with a discussion of the findings and potential directions for future research. The source code is available on GitHub at github.com/GabrielJobert/Comparative-Performance-Analysis-of-Retrieval-Models-on-Textual-Data-with-Diverse-Linguistic-Profile and on Google Colab here.

Chapter 2

Methodology

In what follows, I first describe the retrieval models I used, then detail the datasets selected for evaluation and their preprocessing, finally followed by the evaluation set-up.

2.1 Retriever Models

This section provides an overview of all the Retriever Models used in this experiment.

Table 2.1: Summary of the retriever models used and their caracteristics

Model	Encoder Type	Training Requirements	Efficiency	Advantages	Limitations
BM25	Sparse (lexical term matching, probabilistic)	None	High (fast inverted-index search)	Precise term matching; Interpretable scoring No training needed (domain agnostic) ; Highly efficient at scale	Requires query–document term overlap; Tune parameters (k_1, b) for optimality
DPR	Dense bi-encoder (Two different BERT-based encoders for Q&A)	Supervised (requires large labeled QA pairs for training)	Moderate (precompute embeddings for corpus; ANN index for fast vector search)	Semantic matching (handles synonyms)	Requires extensive training data (needs fine-tuning for new domains); Computationally expensive (BERT encoding and indexing; Large memory footprint)
Contriever	Dense bi-encoder (Same BERT-based encoders for Q&A)	Unsupervised (Unsupervised contrastive learning on raw text)	Moderate (ANN-based dense retrieval, similar as DPR)	No supervised data needed (easily adaptable to new domains); Semantic matching from unlabeled text	May underperform supervised models on very specialized domains; Relies on general-domain pretraining; Computational cost still significant for large corpora
ColBERT	Dense late-interaction (BERT-based, token-level matching)	Supervised (trained on query–document relevance pairs)	Two-stage: ANN search on token embeddings + re-ranking	Fine-grained semantic token matching (finds rare domain-specific terms; handles multi-part queries); High accuracy retrieval (rivals cross-encoders on complex queries)	Memory-intensive index (stores embeddings for all document tokens); Higher query latency than single-vector models; Requires domain-specific training data for best performance

2.1.1 BM25

BM25 (Best Match 25) is a probabilistic information retrieval model that stands as a foundational method in modern search and retrieval systems. Developed as part of the Okapi family at City University London, BM25 is widely regarded as the classical baseline for sparse retrieval and in RAG pipeline in general due to its effectiveness, interpretability, and domain-agnostic applicability [1, 2, 7]. It is also the default ranking function in many search engines, including Elasticsearch [4].

In general, BM25 is typically used as the first-stage retriever, rapidly filtering a large corpus to a manageable subset of candidate documents based on exact term matches. These candidates can then be passed to dense retrievers or language models for further semantic reranking and answer generation [7, 8].

This hybrid approach leverages BM25's high accuracy for relevant documents, ensuring that rare but crucial terms are not overlooked, while subsequent dense retrieval or generative steps address its semantic limitations. BM25 is particularly valuable for domain-specific terminology, where precise term matching remains essential [7, 4].

In my case, BM25 will serve as an interpretable baseline for comparison during my experiments and will represent the category of sparse retrieval model.

Theoretical Foundation

BM25 builds upon the TF-IDF (Term Frequency-Inverse Document Frequency) paradigm, introducing three core enhancements:

- **Term Frequency Saturation:** Diminishing returns for repeated term occurrences, preventing overemphasis on high-frequency terms.
- **Document Length Normalization:** Adjusts scores to avoid bias toward longer documents, which naturally contain more terms.

- **Probabilistic Weighting:** Incorporates the rarity of terms across the corpus to prioritize informative terms.

Given a query $Q = \{q_1, ..., q_n\}$ and a document D, the BM25 score is defined as:

$$\text{BM25}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where:

- $f(q_i, D)$: Frequency of term q_i in document D.
- $|D|$: Length of document D (in tokens).
- avgdl: Average document length in the corpus.
- $k_1 \in [1.2, 2.0]$: Controls term frequency saturation (default: 1.5).
- $b \in [0, 1]$: Controls document length normalization (default: 0.75).

The inverse document frequency (IDF) is calculated as:

$$\text{IDF}(q_i) = \log\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

where N is the total number of documents and $n(q_i)$ is the number of documents containing q_i [3].

This formulation ensures that rare but relevant terms are weighted more heavily, while frequent terms contribute less to the final score. Document length normalization prevents longer documents from being unfairly favored simply due to their verbosity [4].

Strengths and Limitations

Strengths

- **Precise Term Matching:** BM25 excels at retrieving documents with exact query term matches, making it highly effective for keyword-based search [5].

- **Interpretability:** The scoring mechanism is transparent and easily understood.
- **No Supervision Required:** BM25 does not require labeled data or model training, allowing for immediate deployment across domains.
- **Efficiency:** Sparse representations enable rapid scoring and retrieval, even at scale.

Limitations

- **Semantic Limitations:** BM25 cannot capture synonyms or semantic similarity, leading to missed relevant documents when queries and documents use different vocabulary [5]. For example, a search for "automobile" may not retrieve documents containing only "car" unless both terms are present in the query or document.
- **Parameter Sensitivity:** Optimal performance may require careful tuning of k_1 and b, which can be non-trivial [5].
- **Handling Rare Terms:** IDF boosts rare terms, but may not fully address retrieval for queries dominated by very infrequent terms, sometimes requiring query expansion techniques [7].
- **No Personalization or Context Awareness:** BM25 does not adapt to user profiles or contextual intent.

BM25's strengths make it a robust baseline, especially in domains with well-defined terminology and where exact matches are critical. However, its inability to capture semantic relationships means it may underperform on queries requiring understanding of synonyms or paraphrases [5].

2.1.2 Dense Passage Retriever (DPR)

DPR is a foundational neural retrieval model that has significantly advanced the field of open-domain question answering and retrieval systems. Unlike traditional sparse retrieval models based on lexical overlap, DPR leverages dense vector representations to enable semantic matching between queries and documents [9].

In comparative evaluations, DPR has been shown to outperform traditional sparse retrievers like BM25 on open-domain QA benchmarks, especially when queries and relevant

passages exhibit significant lexical variation. However, the effectiveness of DPR in specialized domains is closely tied to the availability of high-quality, domain-specific training data, while training itself can be costly and often requires carefully annotated examples that many users may not have access to.

For my experiment, I used `dpr-question_encoder-single-nq-base` which is a model trained using the Natural Questions dataset, a dataset that contains real user questions issued to Google search, and answers found from Wikipedia by annotators. [35, 36]

Theoretical Foundation

DPR is built upon a bi-encoder architecture, utilizing two independent BERT-based encoders: one for questions (queries) and one for passages (documents). Each encoder transforms its input into a fixed-dimensional dense vector. The similarity between a query and a passage is then computed as the dot product of their respective embeddings, which is proportional to the cosine similarity between them:

$$\text{sim}(q, p) = E_Q(q)^\top E_P(p) \tag{2.1}$$

where $E_Q(q)$ and $E_P(p)$ denote the dense vector embeddings of the query q and passage p, respectively. Since both vectors are typically ℓ_2-normalized during training or inference, this dot product effectively measures their cosine similarity—higher values indicate greater semantic alignment in embedding space.

The training objective is to maximize the similarity between a question and its corresponding positive (relevant) passage while minimizing its similarity with a set of negative (irrelevant) passages. This is accomplished using a contrastive loss, often implemented as a variant of the negative log-likelihood loss:

$$\mathcal{L} = -\log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{p^-} \exp(\text{sim}(q, p^-))} \tag{2.2}$$

where p^+ is a positive passage and p^- denotes one or more negative passages.

Intuitively, the contrastive loss encourages the model to embed a given query closer to

its correct passage and farther away from incorrect ones. By doing so, the learned representation space becomes structured such that semantically similar pairs (e.g., a question and its answer-supporting paragraph) cluster together, while dissimilar ones are well separated. This approach use the assumption that semantic relevance can be captured as geometric proximity in vector space.

A key innovation in DPR training is the use of hard negative mining [10]. Instead of sampling negatives randomly, hard negatives are selected as passages that are top-ranked by a strong baseline retriever (such as BM25) but do not contain the correct answer. This strategy encourages the model to better distinguish between highly similar but non-relevant passages, leading to more robust retrieval performance.

Additionally, DPR benefits from in-batch negatives, where other positive passages in the same mini-batch serve as negatives for a given query. This further increases the diversity and difficulty of negative samples.

For large-scale retrieval, DPR relies on efficient approximate nearest neighbor search algorithms, with FAISS [11] being a popular choice. After encoding all passages in the corpus into dense vectors, these embeddings are indexed using FAISS, which supports scalable similarity search even over millions of documents.

Strengths and Limitations

Strengths

- **Semantic Matching:** DPR's dense representations allow for retrieval based on meaning rather than surface-level token overlap, effectively addressing the vocabulary mismatch problem common in sparse retrieval.
- **Generalization:** The model can retrieve relevant passages even when queries and documents use different wording or synonyms.
- **Scalability:** With ANN search, DPR can efficiently operate on very large corpora.

Limitations

- **Supervised Data Requirement and Domain Adaptation:** DPR requires large

amounts of labeled question-passage pairs for effective training. Its performance may degrade in specialized domains without further fine-tuning on domain-specific data.

- **Computational Cost:** Encoding and indexing large corpora require significant computational resources, both during training and inference.

2.1.3 Contriever

Contriever is a recent and influential unsupervised dense retrieval model that leverages contrastive learning to train high-quality sentence and passage representations without the need for manually annotated data [12]. This innovation positions Contriever as a bridge between traditional unsupervised sparse retrieval methods, such as BM25, and supervised dense retrievers, offering substantial improvements in retrieval quality while maintaining broad applicability across domains.

For my experiment, I am using the standard Contriever model, which is trained on large, unlabeled corpora such as Wikipedia and CCNet, relying solely on the inherent structure of the text to generate training pairs.

Theoretical Foundation

At the heart of Contriever is a contrastive learning framework that enables the model to learn semantic similarity directly from raw text. The model architecture is based on a single BERT-like encoder that maps input text spans into dense vector representations. Unlike supervised dense retrievers such as the previously mentioned DPR, which require explicit query-passage pairs, Contriever constructs positive pairs from unannotated documents using random cropping: two overlapping or adjacent spans from the same document are considered semantically similar (positive), while spans from different documents serve as negatives.

The training objective is to maximize the similarity between positive pairs and minimize it between negatives. This is formalized as a contrastive loss:

$$\mathcal{L} = -\log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{p^-} \exp(\text{sim}(q, p^-))} \tag{2.3}$$

where $\text{sim}(q, p)$ denotes the dot product or cosine similarity between the encoded representations of q and p.

A critical component of Contriever's training is the use of the Momentum Contrast framework [13]. Momentum Contrast maintains a dynamic queue (memory bank) of negative samples and employs a momentum encoder to produce stable representations for negatives. This allows the model to efficiently utilize a large and diverse set of negatives in each training step, which is essential for learning robust and discriminative embeddings.

In practical deployment, Contriever encodes both queries and passages into dense vectors using a single shared encoder, unlike DPR which relies on two separate encoders for queries and passages. But like DPR, encoded vectors are also indexed using approximate nearest neighbor search libraries such as FAISS [11], enabling efficient retrieval at scale. In addition, its training and inference pipelines are highly parallelizable and benefit from modern hardware acceleration, making it suitable for large-scale applications.

Contriever represents a breakthrough in unsupervised dense retrieval, combining the strengths of contrastive learning, large-scale pretraining, and efficient negative sampling. Its empirical performance across diverse domains, adaptability to new data, and ease of deployment make it a compelling choice for modern retrieval-augmented generation pipelines, especially when domain-specific labeled data is limited or unavailable.

2.1.4 ColBERT

ColBERT (Contextualized Late Interaction over BERT) is a neural retrieval model that introduces a novel "late interaction" mechanism, achieving a unique trade-off between the speed of bi-encoder architectures and the expressive power of cross-encoder models [14, 15]. ColBERT's design enables efficient and effective large-scale passage retrieval, making it particularly suitable for retrieval systems and domain-specific search applications.

In this work, I utilize ColBERTv2˜[**?**]Santhanam2022, the latest version of the ColBERT model. ColBERTv2 builds upon the original ColBERT architecture with substantial improvements in retrieval effectiveness and efficiency, including enhanced quantization and a new indexing pipeline.

Theoretical foundation

Late Interaction Mechanism and Model Architecture As mentioned in the intro, traditional neural retrieval models typically fall into two categories: bi-encoders and cross-

encoders. ColBERT bridges these approaches by encoding queries and documents independently but preserving token-level embeddings for both. Specifically, a BERT-based encoder produces a contextualized embedding for each token in the input sequence. This results in a matrix of embeddings for both the query $Q = (q_1, ..., q_m)$ and document $D = (d_1, ..., d_n)$:

$$E_Q = [e_{q_1}, ..., e_{q_m}], \quad E_D = [e_{d_1}, ..., e_{d_n}]$$

The core of ColBERT's late interaction is the **MaxSim** operator. For each query token, it computes the maximum similarity (typically dot product or cosine similarity) with any document token:

$$\text{Score}(Q, D) = \sum_{i=1}^{m} \max_j (e_{q_i} \cdot e_{d_j})$$

This operation allows ColBERT to capture the strongest semantic alignment for each query token, enabling fine-grained matching that single-vector bi-encoders cannot achieve.

To further enhance efficiency, ColBERT projects the high-dimensional BERT embeddings (e.g., 768 dimensions) to a lower dimension (e.g., 128) using a learned linear projection. This reduces memory requirements and speeds up similarity computations [14, 15].

Two-Stage Retrieval and Efficient Indexing ColBERT is designed for practical deployment in large-scale retrieval scenarios. Its retrieval pipeline typically involves two stages:

1. **First-stage Retrieval**: A fast ANN search retrieves a set of candidate documents using compressed document token embeddings. ColBERT leverages FAISS with IVFPQ (Inverted File with Product Quantization) [16, 11], which partitions the embedding space into clusters and quantizes embeddings for efficient storage and search.
2. **Second-stage Re-ranking**: For each candidate, ColBERT computes the MaxSim-based late interaction score between the query and the full set of document token embeddings, providing a fine-grained ranking.

This design enables ColBERT to precompute and store all document token embeddings offline, drastically reducing online computation. The use of IVFPQ and quantization allows ColBERT to index millions of documents with manageable memory and latency, making it suitable for both academic and production-scale retrieval systems.

Training Objective and Optimization ColBERT is trained with a pairwise ranking loss, typically a margin-based or softmax loss, to distinguish relevant (positive) from non-relevant (negative) documents for each query. The training process involves:

- Sampling queries with corresponding positive and negative documents.
- Encoding queries and documents with the BERT backbone and projection layer.
- Computing MaxSim scores for positive and negative pairs.
- Optimizing the model to maximize the margin between positive and negative scores.

Recent extensions, such as relevance-guided supervision and ColBERTv2, introduce further improvements in efficiency and retrieval quality by refining the training objective and incorporating lightweight architectural modifications [15].

Advantages for Domain-Specific and Complex Queries

ColBERT's token-level matching is particularly advantageous for domains with specialized vocabulary, multi-word expressions, or compositional queries:

- **Domain-Specific Terminology**: In technical domains, crucial information may depend on rare or highly specific terms. ColBERT's MaxSim operation ensures that even if only a few query tokens have strong matches in a document, those matches are emphasized in the scoring.
- **Complex Queries**: For queries requiring multi-faceted reasoning or matching across several concepts, ColBERT can capture partial matches for each component, improving retrieval robustness.
- **Paraphrase and Synonym Handling**: By leveraging contextualized embeddings, ColBERT can match semantically similar but lexically different expressions, outperforming purely lexical methods like BM25 in many scenarios [14].

Empirical studies on benchmarks such as BEIR and MS MARCO demonstrate that ColBERT achieves high accuracy, often rivaling or surpassing other dense retrievers and hybrid systems, especially for complex or domain-specific queries [15, 16].

Trade-offs

ColBERT's late interaction mechanism offers substantial gains in retrieval quality but introduces several trade-offs:

- **Memory Overhead**: Storing token-level embeddings for all documents increases memory usage compared to single-vector bi-encoders. Quantization and dimensionality reduction mitigate this but may slightly impact retrieval accuracy.
- **Query-Time Computation**: The MaxSim operation, though efficient, is more computationally intensive than a single dot product, especially for long queries or documents. However, this is still orders of magnitude faster than full cross-encoder inference.
- **Implementation Complexity**: Efficiently managing large embedding tables and ANN indices requires careful engineering, particularly for production-scale deployments.

Despite these challenges, ColBERT represents a significant advance in neural retrieval, combining deep contextual understanding with scalable, efficient retrieval. Its late interaction mechanism enables superior handling of complex and domain-specific queries, making it highly relevant for large-scale semantic search. As retrieval-augmented generation becomes increasingly central to knowledge-intensive NLP applications, models like ColBERT will play a key role in bridging the gap between efficiency and expressive power.

Limitations Encountered During the Project

During the course of this work, the ColBERT model, which was central to my experimental setup, suddenly stopped functioning in my environment. Initially, the model was working correctly and produced results for ObliQA, HotpotQA and CoQA. However, after librairies updates in the Python ecosystem (PyTorch, NumPy, Transformers), critical compatibility issues arose that prevented the model from running as expected. In order to run, ColBERT requires specific versions of these libraries and no solution has been found.

The model continued to break even after downgrading to previous versions because the Colab environment automatically updated certain packages, which aggravated these version conflicts and made it difficult to maintain a stable configuration.

As a result of these technical limitations, the indexing and retrieval pipeline could not be executed anymore. Because of the need to submit the thesis in time, this prevented the successful production of results on the two main datasets selected for evaluation, namely MS MARCO and SQuAD. Consequently, the complete empirical phase of this research could not be carried out as originally planned.

2.2 Datasets

This section describes the datasets used in my experiments, highlighting their characteristics, domains, and relevance to the evaluation of retrieval systems. QA datasets are used for retrieval evaluation because each question is inherently paired with supporting passages, providing a clear signal to assess whether retrievers can identify relevant evidence independent of answer generation. MS MARCO V2 was selected for its noisy, real-world search queries and diverse web passages, offering a realistic test of retrievers in open-domain settings. SQuAD 2.0 was included as a controlled benchmark with formal Wikipedia text and general knowledge domain. ObliQA brings highly complex, domain-specific regulatory language and multi-passage reasoning, testing retrieval under legal and technical constraints. CoQA introduces multi-turn conversational dependencies, challenging retrievers to handle coreference and dialogue context. Finally, HotpotQA explicitly requires multi-hop reasoning across documents, making it ideal for examining how retrievers link dispersed pieces of evidence. A table summarizing the high level characteristics of the dataset (Table 3.1) is available at section 3.1.

2.2.1 MS MARCO V2

MS MARCO V2 (Microsoft MAchine Reading COmprehension) is a large-scale, real-world dataset designed to advance research in information retrieval, question answering, and machine reading comprehension. It is widely recognized as an industry-scale benchmark for practical retrieval systems and has played a central role in both academic and applied IR research [17, 18].

Dataset Composition and Key Features

MS MARCO V2 consists of over one million anonymized queries sampled from real Bing user logs, making it one of the most extensive and diverse QA datasets available [17, 18]. Each query is paired with context passages extracted from web documents using the Bing search engine. These passages are then human-judged for relevance, and human-generated answers are provided when possible [17, 19].

Domain Heterogeneity and Example Queries A defining characteristic of MS MARCO V2 is its domain heterogeneity. Because queries are sourced from actual Bing users, the dataset naturally includes questions from a variety of fields. For example:

- *Legal:* "What is the statute of limitations for breach of contract in California?"
- *Technical:* "How to reset a Cisco router to factory settings?"
- *General:* "Best ways to cook quinoa?"

This diversity ensures that retrieval models trained or evaluated on MS MARCO V2 are exposed to the broad spectrum of language and information needs encountered in real-world applications [20]. Another interesting property of these questions are that they tend to be shortened or incomplete as they are Bing queries, effectively testing the robustness of models I will test.

Implications of Unanswerable Questions Notably, approximately 35% of the queries in MS MARCO V2 are unanswerable—meaning that no sufficient information could be found in the retrieved passages to generate a valid answer. This aspect more accurately reflects the challenges faced by deployed IR and QA systems, where not every user query can be answered with available information. For retrieval models, this means that effective thresholding and answerability detection are crucial: systems must learn not only to retrieve relevant passages but also to abstain or signal uncertainty when no answer is present. [17]. However, as my experiment is focused on how the textual characteristics of the datasets influence the performance of retrievers and not on the models capacity to detect when no answer can be found, I will not includes these unanswerable questions.

Example Entry and Data Fields

Each entry in the MS MARCO V2 dataset is represented as a JSON object with the following primary fields: `query_id`, `query_type`, `query`, `passages`, `answers`, and `wellFormedAnswers`[19]. Below, I present an example entry and describe each variable:


```
{
"query_id": "123456789",
"query_type": "DESCRIPTION",
"query": "What is the statute of limitations for breach of contract
   in California?",
"passages": [
{
"passage_id": "987654321",
"is_selected": true,
"url": "https://www.examplelawsite.com/statute-limitations",
"passage_text": "In California, the statute of limitations for
   breach of written contract is four years, while for oral
   contracts it is two years."
},
{
"passage_id": "987654322",
"is_selected": false,
"url": "https://www.otherlawreference.com/contract-claims",
"passage_text": "Contract claims must be filed within the time
   limits set by state law."
}
],
"answers": [
"Four years for written contracts, two years for oral contracts."
],
"wellFormedAnswers": [
"In California, the statute of limitations for a breach of written
   contract is four years, and two years for an oral contract."
]
}
```

Data Preprocessing for Retrieval

For each dataset, a custom loader consolidates the data into a unified format suitable for both sparse and dense retrievers. In the case of **MS MARCO V2**, I leverage the HuggingFace implementation of the MS MARCO dataset (v2.1) to load the passages and queries. Each query in MS MARCO comes with a set of candidate passages, among which one or more are marked as relevant via an `is_selected` flag. My preprocessing code iterates through each query's passages, collects all passages into a list, and identifies those marked as relevant. I assign each passage a unique ID (concatenating the query ID with the passage index) and store the text of relevant passages as the query's context list. Another list is created, containing all the passages available (relevant or not) from all the queries to create the pool that the retriever will search into. Queries with missing relevant passages are skipped to avoid including unanswerable cases in the evaluation. The transformed output for MS MARCO thus contains, for each query, the original question text and a list of ground-truth relevant passages in '`raw_inputs`', along with metadata specifying the query ID, the IDs of the relevant passages, and the full pool of passage candidates. This ensures that during retrieval, I can evaluate whether the retriever returns any of the known relevant passages. For BM25, passages remain as plain text (to be tokenized by whitespace), while for the other models, these passages will later be encoded into dense vectors—my unified format supports both by storing the raw text which can be tokenized by the respective models. After transformation, the dataset is composed of 55578 queries and 118524 passages.

2.2.2 SQuAD 2.0

SQuAD 2.0 (Stanford Question Answering Dataset 2.0) is a large-scale, controlled benchmark specifically designed to evaluate both the span extraction ability and answerability detection of machine reading comprehension models [21, 22]. It extends the original SQuAD 1.0 by introducing adversarially crafted unanswerable questions.

Dataset Composition and Key Features

SQuAD 2.0 comprises over 150,000 QA pairs, of which approximately 50,000 are unanswerable questions written by crowdworkers to appear similar to answerable ones [21, 22]. Each QA pair is associated with a context paragraph drawn from Wikipedia articles. For an-

swerable questions, the answer is a contiguous text span within the context. All questions and answers are written/validated by crowdworkers, ensuring high linguistic diversity and quality. Unanswerable questions are crafted to be relevant and plausible, often containing distractors or requiring nuanced understanding to detect impossibility.

Adversarial Questions and Comparison with SQuAD 1.0 A distinctive feature of SQuAD 2.0 is its adversarial annotation process. Crowdworkers were instructed to write unanswerable questions that are relevant to the context and could plausibly be answerable, but for which no correct answer exists in the passage [21]. This includes questions with false premises, entity swaps, or requiring information not present in the paragraph. The annotation process also involved validation steps to ensure that no answer span could be reasonably extracted for these questions [23]. As explained in the MS MARCO section, these unanswerable questions will not be included in my experiment.

Example Entry and Data Fields

Each entry in SQuAD 2.0 is structured as a JSON object with the following fields [25, 24]:

```
{
  "id": "56be4db0acb8001400a502ec",
  "title": "University_of_Notre_Dame",
  "context": "Architecturally, the school has a Catholic character.
     Atop the Main Building's gold dome is a golden statue of the
     Virgin Mary. ...",
  "question": "To whom did the Virgin Mary allegedly appear in 1858
     in Lourdes France?",
  "answers": {
    "text": ["Saint Bernadette Soubirous"],
    "answer_start": 12
}
```

Variable Descriptions:

- `id`: Unique identifier for the QA pair (e.g., `"56be4db0acb8001400a502ec"`).
- `title`: Wikipedia article title (e.g., `"University_of_Notre_Dame"`).

- `context`: The passage from which the answer is to be extracted.
- `question`: The natural language question posed about the context.
- `answers`: For answerable questions, a dictionary with:
  - `text`: List of valid answer spans.
  - `answer_start`: List of starting character indices for each answer in the context.

Data Preprocessing for Retrieval

For **SQuAD 2.0**, which provides each question alongside a single context paragraph, the loader uses HuggingFace's `squad_v2` dataset split. The preprocessing here is straightforward: for each question, I extract the question text, the context paragraph, and the answer(s) if available. The dataset's original fields (e.g. `question`, `context`, `id`, and `answers`) are mapped into my unified format. Each SQuAD question is stored with a context list of length one (the given paragraph). All the passages are from 35 whole article and despite that they does not have the same content, they still have the same name from the respective article they came from. That is why the passages receive an identifier which is composed of the article title + the index of the passage from this article. If a question has no answer, I skip the question. All contexts across the SQuAD dataset are later combined into the retrieval index, meaning a retriever must pick the correct paragraph from among all SQuAD paragraphs. By converting the context into a list (even if singleton) and the answer into a list, I normalize SQuAD's structure to match my pipeline's expected input format. This uniform structure allows using the same retrieval code for SQuAD as for the other datasets without special-casing single-context scenarios. After transformation, the dataset is composed of 5928 queries and 5928 passages.

2.2.3 ObliQA

ObliQA (Obligation-based Question-Answering Dataset for Regulatory Compliance) is a specialized dataset designed to advance regulatory natural language processing (RegNLP) with a focus on compliance verification, information retrieval, and question answering in the legal and financial domain [26, 27].

ObliQA serves as a domain-specific stress test for retrieval-augmented generation systems

in legal and financial contexts. Its complex document structures, multi-hop reasoning requirements, and synthetic yet validated annotation methodology make it a valuable resource for developing and benchmarking advanced regulatory NLP systems.

Dataset Composition and Key Features

ObliQA is constructed from approximately 640,000 words of regulatory text sourced from Abu Dhabi Global Markets (ADGM), encompassing 40 documents that range from 30 to 100 pages each. These documents are characterized by complex internal structures, including numbered clauses, subsections, and extensive cross-references, mirroring the intricacies of real-world regulatory frameworks.

ObliQA contains 27,869 validated question-passage pairs, with the majority referencing a single passage and a substantial portion requiring two or more passages for a complete answer. Table 2.2 summarizes the distribution:

#Passages	1	2	3	4	5	6
Questions	21,187	5,036	1,196	268	121	61

Table 2.2: Distribution of questions by number of passages referenced in ObliQA [26, 27].

The ObliQA dataset maintains the integrity of the original legal documents by preserving their hierarchical and referential structure. The preparation process begins with the collection of .docx source documents, which are manually structured to ensure consistency. Tables and figures are explicitly tagged, and the documents are converted into both plain text and structured JSON formats [26].

Synthetic Annotation Pipeline Unlike crowd-sourced datasets such as SQuAD, ObliQA employs a synthetic annotation pipeline: QA pairs are generated using the `gpt-4-turbo-1106` model, with prompts tailored for both single-passage and multi-passage scenarios. For single-passage questions, the context must implicitly contain the answer. For multi-passage questions, prompts are designed to reflect realistic compliance queries that require synthesizing information from multiple regulatory sections. To ensure high-quality alignment between questions and passages, a Natural Language Inference model (`nli-deberta-v3-xsmall`) is used for validation. Only pairs where the passage(s) entail the question are retained, while contradictory or neutral pairs are filtered or further reviewed [26, 27].

Example Entry and Data Fields

Each entry in ObliQA is structured as a JSON object with the following fields [26]:


```
{
  "QuestionID": "739921c1-385a-4735-a052-dee9fba73602",
  "Question": "What are the key compliance indicators that a Fund
     Manager should monitor to confirm that a Passported Fund is
     being managed and operated within its constitutional framework
     and applicable ADGM legislation?",
  "Passages": [
    {
      "DocumentID": 16,
      "PassageID": "Part 3.6.(2)",
      "Passage": "Each Reporting UAE Financial Institution shall
         establish and implement appropriate systems and internal
         procedures to enable its compliance with the Cabinet
         Resolution and these Regulations."
    },
    {
      "DocumentID": 5,
      "PassageID": "6.1.2",
      "Passage": "The Fund Manager of a Passported Fund must: (a)
         ensure that the Passported Fund is at all times managed and
          operated in compliance with its constitution, in
         accordance with applicable ADGM legislation, and with these
          Rules; and (b) maintain, or cause to be maintained, a
         Unitholder register for the Passported Fund."
    }
  ]
}
```


Variable Descriptions:

- `QuestionID`: Unique identifier for the question (e.g., `"739921c1-385a-4735-a052-dee9fba73602"`).
- `Question`: The compliance-focused question referencing regulatory obligations.
- `Passages`: List of passage objects, each with:

- `DocumentID`: Identifier for the source regulatory document.
- `PassageID`: Section or clause reference within the document (e.g., `"Part 3.6.(2)"`).
- `Passage`: The full text of the regulatory clause or section.

This structure supports both single-passage and multi-hop question answering, reflecting the complexity of real-world compliance queries.

Data Preprocessing for Retrieval

The **ObliQA** dataset was ingested from a JSON file (as no direct HuggingFace loader was available). Each entry in ObliQA consists of a complex question (often multi-sentence) and a set of relevant regulatory text passages needed to answer it. The loader reads each question's entry and collects all provided passages. I do not have explicit labels indicating which particular passage contains the answer – rather, the entire set of passages from the question is considered the supporting context for the question. In my transformation, I therefore treat all these passages as the context for the query. The question text is stored in `raw_inputs["question_text"]`, and the list of all passage texts is stored in `raw_inputs["context_text"]`. I assign each passage a unique ID (using the provided `PassageID` from the dataset, prefixed by the question ID to ensure global uniqueness). All the passages from all the raw_inputs are merged into a pool of passages where the retrievers will search into during the analysis. Because ObliQA's JSON did not include a readily usable answer field (answers in this dataset are often free-form or assumed to be derivable from the passages), I set the answer key to None in the mapping; accordingly, the loader leaves the 'answers' field empty. Again, the absence of answer does not impact the experiment as the focus is on the retrieval part and not on the answer generation part. The unified representation of ObliQA thus contains the question and a list of associated passage texts (often multiple passages per question), and the metadata includes the question's ID and a list of all passage IDs for that question. This format allows BM25 to index all regulatory passages and dense retrievers to embed them. After transformation, the dataset is composed of 2786 queries and 6143 passages.

2.2.4 CoQA

The Conversational Question Answering (CoQA) dataset is a large-scale benchmark specifically constructed to evaluate the ability of models to engage in multi-turn, context-dependent question answering [28, 29]. CoQA addresses a critical gap in QA research by moving beyond single-turn, factoid QA to focus on the challenges of dialogue, context tracking, and pragmatic reasoning.

Its multi-domain scope, conversational structure, and emphasis on coreference and context make it an essential testbed for evaluating the conversational consistency and generalization of retrieval systems.

Dataset Composition and Key Features

With over 127,000 QA pairs spanning more than 8,000 conversations, CoQA is one of the most comprehensive resources for conversational QA. Each data instance is a conversation—a sequence of questions and answers—between a questioner and an answerer, grounded in a passage. This format simulates real-world information-seeking dialogues, where each question may depend on the context established by previous turns, requiring models to maintain a memory of the entire conversation. Unlike extractive QA datasets (e.g., SQuAD), CoQA allows answers to be free-form, not limited to exact spans in the passage. However, each answer is linked to a supporting evidence span (rationale) in the passage, enabling both generative and extractive evaluation [28]. The dataset includes a wide range of conversational phenomena, such as clarification questions, follow-ups, and topic shifts, which are rarely present in single-turn QA datasets.

Conversational Consistency and Turn-Taking A defining challenge of CoQA is the requirement for conversational consistency across multiple turns. Later questions in a conversation often refer back to entities, events, or answers mentioned earlier, using pronouns (“he”, “she”, “it”), definite descriptions (“the animal”), or even implicit references. For example, in the literature and children’s stories domains, questions such as “Where did she go next?” require the model to track the referent of “she” across several previous turns. This necessitates robust dialogue history modeling, coreference resolution, and pragmatic inference.

Moreover, turn-taking patterns vary across domains. In the science and Wikipedia domains, questions may be more fact-based and sequential, while in literature or Reddit, conversations may involve more narrative and contextual dependencies. This diversity in dialogue structure provides a rigorous test for retrieval systems aiming to maintain context and coherence in multi-domain settings.

Example Entry and Data Fields

Each CoQA entry is structured as a JSON object with the following fields [29, 31]:


```
{
  "source": "literature",
  "story": "Mary had a little lamb. Its fleece was white as snow.
     She took it to school one day.",
  "questions": [
    "Who had a little lamb?",
    "What color was its fleece?",
    "Where did she take it?"
  ],
  "answers": [
    {
      "input_text": "Mary",
      "answer_start": 0,
      "answer_end": 4
    },
    {
      "input_text": "white as snow",
      "answer_start": 32,
      "answer_end": 45
    },
    {
      "input_text": "to school",
      "answer_start": 57,
      "answer_end": 66
    }
  ]
}
```

Variable Descriptions:

- `source`: The domain of the passage (e.g., `"literature"`, `"science"`, `"reddit"`).
- `story`: The passage or context for the conversation.
- `questions`: List of conversational questions, each referencing the passage and possibly previous answers.
- `answers`: List of answer objects, each with:
  - `input_text`: The free-form answer.
  - `answer_start`, `answer_end`: Character indices of the evidence span in the passage supporting the answer.

Dialogue Chain and Coreference Example

Consider the following conversation from the "literature" domain [28, 30]:

- Q1: Who had a little lamb? A1: Mary
- Q2: What color was its fleece? A2: white as snow
- Q3: Where did she take it? A3: to school

Here, "its" in Q2 refers to "lamb" in Q1, and "she" in Q3 refers to "Mary" in Q1. Accurately answering Q2 and Q3 requires resolving these coreference links using the conversational history. This example illustrates the necessity for models to track entities and context across multiple turns, a core requirement for effective conversational QA.

Data Preprocessing for Retrieval

For the CoQA conversational dataset, a custom loader converts its multi-turn dialogue format into a retrieval setting. I use the CoQA development set (500 dialogues) and extract one representative query from each conversation to focus on context-dependent retrieval. Specifically, I select the *first question* of each conversation as the standalone query, since it

generally contains on the actual name of the person/place the story is talking about and thus presents the easiest retrieval challenge in this context. The associated passage (the story text for that conversation) serves as the context for that query. All such passages from all CoQA conversations are then merged into a unified retrieval index. A retriever must therefore pick the correct story segment from among 500 total passages. Each query's answer text (from the dataset) is stored for completeness, but since I evaluate retrieval only, the answer field is not used.

2.2.5 HotpotQA

HotpotQA is a large-scale, multi-hop QA dataset that has become a cornerstone for evaluating advanced reasoning and explainability in retrieval systems [32, 33, 32]. Unlike single-hop QA datasets, HotpotQA explicitly requires systems to integrate information from multiple sources, mirroring the complex, distractor-rich context, compositional reasoning found in real-world information retrieval and decision-making tasks [32].

Dataset Composition and Key Features

HotpotQA consists of approximately 113,000 crowd-sourced questions, each designed so that answering requires synthesizing evidence from at least two distinct Wikipedia articles. This multi-article requirement turns HotpotQA into a true multi-hop retrieval benchmark—systems must not only fetch individually relevant passages but also identify and link evidence across distinct documents, testing their ability to coordinate and synthesize information from multiple sources [32].

A unique aspect of HotpotQA is its explicit annotation of supporting facts. For each question, annotators identify the exact sentences within the relevant paragraphs that are essential for answering. This enables strong supervision not only for answer generation but also for explainability, as models can be evaluated on their ability to retrieve and highlight the correct reasoning path [32].

In the distractor setting, each question is paired with ten context paragraphs: two gold paragraphs containing the supporting facts and eight distractor paragraphs selected for their lexical similarity but lack of relevance. This design mimics the challenges of real-world information retrieval, where systems must sift through large volumes of potentially misleading

or tangential information to identify the truly relevant evidence [32, 32].

HotpotQA covers a broad spectrum of multi-hop reasoning patterns:

- **Comparison Questions** (34%): Require comparing attributes or facts across entities (e.g., "Who has won more Olympic medals, X or Y?").
- **Intersection/Conjunctive Questions** (29%): Require finding entities or facts satisfying multiple conditions (e.g., "Which author wrote both X and Y?").
- **Bridge/Multi-Entity Questions** (37%): Require following a chain of references or linking facts across articles (e.g., "Who is the spouse of the director of movie X?").

This diversity ensures that models are tested on a wide variety of inference strategies, from entity resolution to logical conjunction and comparison [32].

Example Entry and Data Fields

Each entry in HotpotQA is a JSON object with the following structure [33]:


```
{
  "_id": "5ad3d5605542996e685252bc",
  "question": "Which team did the quarterback who led the 1985
     Chicago Bears play for in college?",
  "answer": "Brigham Young University",
  "supporting_facts": [
    ["Jim McMahon", 1],
    ["1985 Chicago Bears season", 1]
  ],
  "context": [
    [
      "Jim McMahon",
      [
        "Jim McMahon is a former American football quarterback.",
        "He played college football at Brigham Young University.",
        "He was drafted by the Chicago Bears in the first round of
           the 1982 NFL Draft."
      ]
```

```
    ],
    [
      "1985 Chicago Bears season",
      [
        "The 1985 Chicago Bears season was the franchise's 66th
           season in the National Football League.",
        "The Bears were led by quarterback Jim McMahon.",
        "They finished the regular season with a 15  1   record."
      ]
    ],
    // ...8 distractor paragraphs omitted for brevity...
  ]
}
```


Variable Descriptions:

- `_id`: Unique identifier for the QA pair.
- `question`: The multi-hop question requiring reasoning over multiple paragraphs.
- `answer`: The answer string (not present in the test set).
- `supporting_facts`: List of [paragraph title, sentence index] pairs, indicating which sentences in which paragraphs are required to answer the question.
- `context`: List of paragraphs, each as a [title, list of sentences] pair. The first two are gold paragraphs, the rest are distractors.

In the above example, the system must first identify that Jim McMahon was the quarterback for the 1985 Chicago Bears (from the "1985 Chicago Bears season" paragraph), then determine where he played college football (from the "Jim McMahon" paragraph), thus demonstrating multi-hop reasoning.

Data Preprocessing for Retrieval

For **HotpotQA**, a dataset featuring multi-hop questions, the data required special handling due to its format. Each HotpotQA question in the dev set comes with multiple supporting facts across different Wikipedia articles. I loaded a transformed JSON for HotpotQA where

each entry contains the question, a list of supporting facts (article titles and sentence indices), and the full text of several candidate passages. In my preprocessing, I concatenate the sentences of each article's passage into a single text block and assign that passage a unique ID derived from the article title (I replace spaces with underscores to form an ID). I then determine which of these passages contain the supporting facts: for each article, if its title appears in the question's supporting facts list, I mark that article's full passage as part of the ground-truth context. All passages from all articles are collected as candidate contexts (with their IDs), but only those passages that correspond to the gold supporting articles are recorded as the relevant context for the query. If a question has no identified supporting passages (which can happen in some edge cases or errors in the dataset), I skip that question to avoid evaluating on incomplete data. The transformed HotpotQA entries thus contain the question text and a list of one or more supporting passages' texts as `context_text`. The metadata includes the question ID, a list of supporting passage IDs (as the ground truth set), and a list of all passage IDs considered for that question (to facilitate analysis of retrieval over the full candidate set). All passages are stored as plain text (allowing BM25 indexing by terms, and the others encoding by their neural models), ensuring consistency across retriever implementations. After transformation, the dataset is composed of 6162 queries and 16767 passages.

2.3 Evaluation Setup

After preparing the datasets as described above, I evaluate the retrieval performance of the four models – BM25, DPR, Contriever and ColBERT – in a standardized pipeline. my evaluation procedure constructs a retrieval corpus for each dataset and measures how effectively each model can retrieve the relevant context passages for the queries.

Candidate Passage Indexing

For each dataset, I aggregate all candidate passages from every query into a single corpus (index). This means that, for a given question, the retriever must search among all passages of that dataset – not just the passages originally paired with that query. For example, in SQuAD 2.0 the index consists of every paragraph from the dev set; a retriever processing one SQuAD question must distinguish the correct paragraph from hundreds of others. This open-domain retrieval setup makes the task challenging and allows us to assess retriever gen-

eralization. The pipeline implementation explicitly iterates over each query and collects its associated passages (and IDs) into global lists, which are then used to initialize the retriever's index. In total, the number of indexed passages ranges from a few hundred (for smaller datasets like HotpotQA dev) up to many thousands for larger sets (e.g. MS MARCO). Each passage is identified by a unique ID so that I can trace which query it originated from.

Retrieval Procedure

I evaluate each retriever model by issuing every query to the model and retrieving the top k results, for $k \in \{1, 5, 10\}$. The BM25 model uses tokenized passages to compute term-matching scores, while DPR, Contriever, and ColBERT use their neural encoders to produce dense vector embeddings of all passages and perform similarity search. In my implementation, dense retrieval for DPR and Contriever is accelerated with a FAISS index: after encoding all passages into embeddings, I build an index on these vectors for similarity search. When a query is submitted, these models encode the query into a vector, and the nearest neighbors in the vector index are returned as the top-k passages. This mirrors the typical ANN search approach for dense retrievers. ColBERT, on the other hand, uses its own native indexing process and computes the MaxSim-based late interaction score between the query and the full set of document token embeddings. BM25, by contrast, scores all passages on the fly using the Okapi formula without a learned index.

Metrics Logged

For each query, I determine whether the retrieved set contains at least one of the ground-truth relevant passages identified in my preprocessing. I define a retrieval as "Correct" if any one of the top-k retrieved passage IDs matches an ID in the query's ground-truth context set. This allows credit for retrieving any relevant evidence. I compute this correctness for $k = 1, 5, 10$ for every query, and then aggregate the results. The primary metric I report is Top-k accuracy (the fraction of queries for which a relevant passage is present in the top k). After running all queries, the pipeline calculates the mean accuracy for each model at each k by averaging the "Correct" indicator across queries. These results are saved in summary tables for analysis. Concretely, my code writes out a CSV file listing each query's outcome for each model and each k, then groups these results by model and k to compute the average Correct rate. The outcome is a set of accuracy values per model and per retrieval depth. This evaluation setup is consistent across all datasets, ensuring that I can directly compare

retrievers' performance in different domains. I also log the retrieval time for each query (and cumulatively) to gauge efficiency, though in this thesis my discussion focuses on accuracy metrics. A second and complementary metric will be computed exclusively for multi-hop datasets (where more than one passage is needed to answer the query) where every passages will be required to be retrieved to be counted as "Correct" instead of only at least one. This metrics will be called strict accuracy and will be computed for $k = 5, 10$ for ObliQA and HotPotQA only. All experiments are run on the respective dev or test portions of the datasets, using identical retrieval settings. The results (detailed in Chapter 3) shed light on how each retriever performs and where each has strengths or weaknesses in regard to the type of textual data it retrieves.

Chapter 3

Empirical Results

To provide a clear overview of the data sources used throughout this research, Table 3.1 presents a summary of the datasets considered in this study. This consolidated view aims to facilitate comparison and understanding of the datasets, serving as a reference point for the analysis conducted in the subsequent sections.

3.1 Technical Linguistic Analysis

3.1.1 Passages Analysis

I provide here a rigorous textual and structural profile of the passages of the five QA datasets. This analysis draws upon quantitative metrics from my local textual analysis, complemented by theoretical expectations from their construction.

Summary of passages' domain and style The domain of the source material greatly influences the language in each dataset. ObliQA is an outlier in domain specificity and complexity: its regulatory/legal texts yield a very formal style with dense jargon, whereas SQuAD and HotpotQA use broad-domain Wikipedia text that, while formal and factual, is intended for general readership. CoQA, pulling from varied domains, includes both simple and complex language, but its conversational format means questions are often phrased in a casual, spoken style. MS MARCO spans an extremely wide domain (the entire web), but

Dataset	Domain and Source	Question Style	N of Queries	N of Passages	Multi-hop?	Multi-turn?
MS MARCO V2	Open-domain web (user search queries; real logs)	Brief, often informal queries (natural distribution; some incomplete)	55 578	118 524	Partial (multi-doc retrieval, usually single-hop)	No
SQuAD 2.0	Wikipedia (general knowledge; crowd-sourced Q)	Well-formed factoid questions, independent	5 928	5 928	No	No
ObliQA	Financial regulatory text (legal domain; LLM-generated Q)	Long, formal compliance questions referencing rules	2 786	6 143	Yes (requires 1–6 passages)	No
CoQA	Multi-domain passages (stories, news, etc.; crowd dialog)	Conversational, context-dependent questions in a sequence	500	500	No	Yes
HotpotQA	Wikipedia (open-domain; crowd-sourced Q)	Complex questions combining facts (bridge or comparison questions)	6 162	16 767	Yes (typically 2 docs)	No

Table 3.1: Comparison of datasets by domain, question characteristics, and size. The number of queries and passages are from dev/test set after transformation. Multi-hop refers to needing multiple distinct text pieces for one question. Multi-turn indicates conversational context across questions.

many passages are drawn from informational web pages similar in style to Wikipedia or news; however, the presence of forums and less formal sources, plus the unedited nature of real user queries, injects more linguistic noise (slang, typos, inconsistent grammar) than the other datasets.

Dataset	**Avg Length**	**Avg MLU**	**Flesch**	**TTR**
MS MARCO V2	38	14.3	50.0	0.025
SQuAD 2.0	146	29.6	35.0	0.010
HotpotQA	29	28.5	43.1	0.120
CoQA	309	21.5	64.9	0.105
ObliQA	103	67.0	0.25	0.016

Table 3.2: Comparative metrics across datasets for passages. Avg Length (average number of word by passage) and Mean Length of Utterance (average number of word by sentence) reflect textual extent and syntactic density. Flesch is a score ranging from 0 to 100 denoting readability ease (higher score is easier to read). TTR (Type-Token Ratio) quantifies lexical diversity by dividing the distinct number of words by the total number of words. Closer to 1 means almost every word is unique, lower means more repetition.

As shown in Table 3.2, the datasets exhibit pronounced contrasts in both surface textual metrics and deeper structural patterns, which can be interpreted to forecast theoretical challenges for downstream language understanding tasks.

Textual length and clause complexity. CoQA features by far the longest average textual passages (over 300 tokens), reflecting the accumulation of multi-turn dialogue. However, its average MLU remains comparatively moderate (21.5), suggesting that each utterance tends to be syntactically simpler, consistent with conversational style. In stark contrast, ObliQA, despite shorter passages than CoQA, records an exceptionally high MLU (67.0), signifying highly intricate clause nesting typical of regulatory prose. This indicates that processing ObliQA necessitates parsing of multi-layered conditional and referential constructs, whereas handling CoQA involves maintaining extensive discourse memory across simpler sentences. SQuAD 2.0 and HotpotQA occupy an intermediate zone: both manifest moderately long passages (especially SQuAD) paired with substantial clause complexity (around 28–30 MLU). HotpotQA's relatively short overall length (29 tokens) but high clause density (28.5 MLU) suggests compact compositional questions embedding multiple entities or comparative relations. MS MARCO V2, with shorter passages (38 tokens) and lower MLU (14.3), indicates less syntactically demanding structures but potentially greater fragmentation due to its origin in web snippets.

Lexical concentration and diversity. Type-token ratios (TTR) quantifies lexical diversity by dividing the distinct number of words by the total number of words. HotpotQA and CoQA exhibit markedly higher TTR values (0.12 and 0.105), indicative of richer lexical diversity per segment. This is expected in HotpotQA, where its multi-hop aspect introduce multiple distinct entities in the passages, and in CoQA where varied dialogue turns traverse different referents. By contrast, SQuAD 2.0 and ObliQA show lower TTR (0.010 and 0.016 respectively), reflecting repetitive terminology—consistent with SQuAD's focus on specific Wikipedia paragraphs and ObliQA's domain-restricted regulatory vocabulary. MS MARCO's modest TTR (0.025) paired with an extremely large overall vocabulary (over 80,000 types) underscores broad corpus heterogeneity but repeated local usage.

Readability and formal linguistic burden. Readability indices corroborate these structural insights. ObliQA stands out with a Flesch score near zero, aligning with graduate-level FK and SMOG estimates, confirming the formidable complexity of parsing legal provisions. In contrast, CoQA's high Flesch score (65) situates it near typical conversational prose, while HotpotQA and SQuAD cluster in the low 30s to 40s, matching their encyclopedic, multi-fact sentences. MS MARCO V2, drawn from informal user queries and varied web text, resides in a mid-range readability zone (50).

3.1.2 Query Analysis

Here is the same textual profile as in the precedent section but for the queries of the five datasets. Again, these metrics are from my own local textual analysis.

Dataset	Avg Length	Avg MLU	Flesch	TTR
MS MARCO V2	6	5.5	63	0.12
SQuAD 2.0	12	11	58	0.10
HotpotQA	15	13.5	60	0.15
CoQA	7	6	85	0.35
ObliQA	30	28	21	0.05

Table 3.3: Comparative metrics across datasets for **queries**. Avg Length (average number of words per query) and Mean Length of Utterance (average number of words per sentence) reflect query extent and syntactic density. Flesch is the readability ease score (higher means easier to read). TTR (Type-Token Ratio) quantifies lexical diversity.

Summary of queries' creation In terms of question formation, SQuAD and HotpotQA questions are manually crafted by crowdworkers who had access to relevant information, resulting in grammatically complete questions with a clear factual focus. CoQA's questions are also human-generated but in an interactive setting – thus they are often incomplete on their own and rely on discourse context. MSMARCO's queries, being actual search queries, tend to be the shortest and sometimes not even proper questions (e.g., keywords like "distance earth sun" or imperative forms like "Define photosynthesis"). ObliQA's questions, interestingly, were LLM-generated based on the regulatory text; they read like realistic queries a compliance officer might ask, which makes them longer and syntactically richer than typical crowd-sourced factoid questions.

Query length and form. The queries in these datasets vary widely in length and structure. CoQA questions are extremely short on average (around 7 words, with some follow-up questions as brief as a single word or phrase like "Why?" or "Who?"), reflecting their conversational and context-dependent nature. MSMARCO queries are also very terse (mean $\sim$6 words), often not even full sentences but rather keyword-style search queries (e.g., "weather in dallas tomorrow" or "lyrics to Imagine Dragons Believer"). In contrast, SQuAD 2.0 and HotpotQA questions are longer (approximately 11–18 words on average) and are well-formed natural questions written in a self-contained manner. ObliQA queries are the longest by far (roughly 30 words on average), often comprising complex, multi-clause sentences. This is because ObliQA's questions frequently provide a detailed scenario or context (as one might see in a compliance inquiry) before asking for specific information. These characteristics affect how a retrieval model can interpret the query: a long, well-specified question (e.g., an ObliQA query) presents more lexical information to latch onto, while a clipped query (e.g., "What next?" in CoQA or a two-word MS MARCO query) provides minimal clues and may be ambiguous without additional context.

Vocabulary and readability. The language style of the queries ranges from informal to highly domain-specific. CoQA's questions are written in simple, conversational language, often mirroring spoken English. Their vocabulary is basic (e.g., pronouns like "she," "it," commonplace verbs), and the readability is very high – indeed, CoQA queries score an average Flesch Reading Ease of about 85, the easiest among the datasets. MSMARCO queries are a mixed bag: many are simple factual questions, but others include colloquial expressions, abbreviations, or web slang. This gives MSMARCO a relatively easy reading level (around 63 Flesch Ease) but also the most "uncurated" lexicon – a wide mix of formal and informal

terms and sometimes misspelled or telegraphic phrases. SQuAD and HotpotQA questions, being written by crowdworkers and focusing on Wikipedia topics, fall in the middle: they use standard written English with an academic or encyclopedic tone, yielding moderate readability (Flesch Ease ∼58–60). Their vocabulary includes proper nouns and specific concepts from a broad range of domains, but the questions are generally clear and grammatically well-formed. ObliQA questions are on the opposite extreme from CoQA: they are written in formal legal/financial language with domain-specific terminology (e.g., "ADGM legislation," "Passported Fund") and complex syntax. Consequently, ObliQA queries are the hardest to read (average Flesch Ease ∼21). The lexical diversity of the query sets also reflects their nature: HotpotQA and CoQA have high type–token ratios (HotpotQA ∼0.15, CoQA ∼0.35), since each question often introduces new entities or wording (CoQA's multi-domain conversations could contribute to a particularly diverse vocabulary despite the questions being short but it could also be due to the fact that there are only 500 questions, meaning that de denominator is relatively low). SQuAD and MSMARCO have lower TTR (around 0.10–0.12) because many questions use similar functional words or templates (e.g., "What is...," "Who is..."), and MSMARCO's huge query set contains many rephrasings of common queries. ObliQA's queries have the lowest TTR (around 0.05) – a sign that the same technical terms and phrasings repeat across questions (for example, many compliance questions share words like "ensure," "compliance," "fund," etc.). In summary, CoQA and MSMARCO queries use simpler everyday language, SQuAD and HotpotQA queries use polished informational language, and ObliQA queries use specialized jargon-heavy language. These differences imply that a retrieval system may face very different language-match challenges: e.g., slang or shorthand in MS MARCO might confound a model trained on formal text, whereas ObliQA's heavy jargon could be out-of-vocabulary for models without legal domain training.

Context-dependence and reasoning. Another critical contrast is the degree to which queries are self-contained or require external context and reasoning. CoQA is the only dataset that is truly conversational: each question is part of a multi-turn dialogue and often cannot be understood in isolation. Coreference is rampant – questions include pronouns ("he," "she," "it," "they") or definite references ("the city," "that event") whose meaning depends on previous turns. For example, a CoQA query like "Where did she go next?" is only answerable if the system knows who "she" refers to and what was happening before. This poses a unique challenge to retrieval: the system ideally needs to incorporate conversation history or else risk retrieving the wrong passage. None of the other datasets have this kind of sequential dependency. Each MSMARCO, SQuAD, HotpotQA, or ObliQA query is

a stand-alone question. That said, MSMARCO queries can be context-underspecified in a different way: as user queries, they sometimes assume implicit context or contain ambiguity. A query like "jaguar speed" could refer to the animal or the car, and without disambiguation, a retriever might retrieve irrelevant results. In SQuAD and HotpotQA, ambiguity is less common because questions were created with a specific Wikipedia context in mind, making them fairly precise. HotpotQA and ObliQA do introduce a requirement for multi-hop reasoning. HotpotQA questions are explicitly designed to require bridging two Wikipedia articles or comparing two entities, so the question will often contain two clues (e.g., two entity names) that need to be looked up. The HotpotQA query itself usually makes this clear by mentioning both parts of the needed reasoning (for instance, "Which author wrote both *Book X* and *Book Y*?" names two works). ObliQA questions can also require multi-hop reasoning, although in a more implicit way: the question may stipulate conditions that are located in different sections of a long regulation. For example, an ObliQA query might ask whether a certain scenario violates regulations, implicitly requiring the model to find multiple relevant clauses across the law. In contrast, SQuAD questions are strictly single-hop (each question is answerable from one paragraph) and CoQA, while it involves reasoning across dialogue turns, still confines each question to a single relevant passage (the story given for that conversation). MSMARCO falls somewhere in between: it provides multiple passages per query as context, and sometimes the answer is a synthesis of information (the dataset did not guarantee purely single-paragraph answers). However, in many MSMARCO cases, one of the retrieved snippets contains the answer, meaning the retrieval task is largely about finding one good passage among many. In summary, these facets suggest that different retrieval models may have advantages on different datasets – e.g., a model handling CoQA must be robust to coreference and missing keywords, whereas a model for HotpotQA must handle multi-hop clues, and one for ObliQA needs to grapple with long, detailed queries but with telltale domain keywords.

3.1.3 Implications for retrieval and generation

These textual divergences have direct theoretical consequences for the design of retrieval-augmented or generative language systems. Processing ObliQA entails navigating deeply nested clause structures, resolving dense anaphoric and cross-referential patterns, and normalizing highly specialized regulatory terminology. By contrast, HotpotQA demands compositional semantic interpretation to reconcile multiple distinct factual nuggets within short, dense prompts, requiring retrieval systems to robustly align disparate but jointly informative

contexts.

CoQA's profile foregrounds discourse-level challenges: extensive context windows coupled with pronoun-rich, ellipsis-dependent questions necessitate maintenance of conversational state and coreference chains. Meanwhile, MS MARCO's web-derived content places emphasis on coping with lexical noise, incomplete phrasing, and colloquial style, requiring normalization strategies to stabilize retrieval effectiveness.

SQuAD 2.0, with relatively formal but moderate-length texts and clear factual questions, exemplifies a controlled reading comprehension environment. However, the low TTR coupled with complex syntax still necessitates careful parsing to distinguish nuanced entailment or to identify unanswerability in high-surface-overlap distractor cases.

In summary, the dataset landscape reveals that linguistic structure—be it through syntactic depth, lexical diversity, or discourse continuity—profoundly shapes the retrieval and comprehension hurdles a downstream system must surmount. Effective question answering architectures must therefore explicitly adapt to these structural characteristics, whether through sophisticated multi-hop index traversal, enhanced discourse state tracking, or syntactic parsing sensitive to deeply nested clause embeddings.

3.2 Retrieval Results and Analysis by Datasets

To set the stage for the retrieval analysis, Table 3.4 summarizes the top-1 retrieval performance achieved by each model across the different datasets. This overview provides an immediate comparison of baseline effectiveness, highlighting variations in model performance depending on the dataset characteristics. By presenting these results upfront, the table serves as a reference point for the more detailed discussions and analyses that follow in this chapter.

Table 3.4: Top-1 Retrieval Accuracy (Non-Strict) across Models and Datasets

Model	**MS MARCO V2**	**SQuAD 2.0**	**ObliQA**	**CoQA**	**HotpotQA**
BM25	20.5%	63.6%	53.0%	35.8%	66.4%
DPR	39.6%	41.0%	19.6%	26.8%	46.7%
Contriever	**56.4%**	**61.7%**	42.6%	45.4%	66.9%
ColBERT	N/A	N/A	**60.0%**	**63.8%**	**73.8%**

3.2.1 MS MARCO: Retrieval Results and Analysis

(During the experiments on MS MARCO, the ColBERT model could not be executed due to critical library incompatibilities. Although retrieval with other datasets such as ObliQA, HotpotQA, and CoQA were successful, subsequent updates in the Python ecosystem (PyTorch, NumPy, Transformers) caused the model to stop functioning in the Colab environment. ColBERT depends on specific versions of these libraries, and even attempts to downgrade to earlier versions failed, as Colab automatically updated certain packages. This instability made it impossible to maintain a working configuration, and as a result, no empirical results could be produced for MS MARCO.)

I first examine the retrieval performance on the MS MARCO V2 dataset. Table 3.5 presents the top-k retrieval accuracies for the different models. Perhaps surprisingly, the neural retrievers far outperform the lexical baseline in this open-domain setting. BM25 achieves only about 20.5% accuracy@1 – meaning it retrieves a relevant passage as the first result for roughly one-fifth of the queries – and about 36% by top-10. In stark contrast, the Contriever dense model attains approximately 56.4% accuracy@1, and up to 83.2% of queries have a relevant passage in the top-10. DPR performs in between, with about 39.6% top-1 and 65.8% top-10 accuracy. In other words, Contriever more than doubles BM25's accuracy at rank 1, and it maintains an excellent accuracy at higher ranks, retrieving relevant information for the vast majority of queries by 10 results. DPR, despite being a supervised dense retriever, lags behind Contriever but still substantially outperforms BM25.

Retriever	**Top-1**	**Top-5**	**Top-10**
BM25	20.5%	31.9%	36.3%
DPR	39.6%	59.3%	65.8%
Contriever	**56.4%**	**77.9%**	**83.2%**

Table 3.5: Retrieval accuracy on MS MARCO. Values indicate the proportion of questions for which a relevant passage was retrieved in the top k results (accuracy@k).

Analysis: The strong showing of dense retrievers on MS MARCO can be attributed to the nature of the queries and corpus. Its moderate passages's readability (Flesch 50) and especially its corpus's broad lexical variety (over 80,000 unique terms; TTR 0.025) likely contribute to BM25's struggles in this setting. Many queries use informal phrasing, slang, or synonyms that do not appear verbatim in relevant passages, so exact-match retrieval is brittle. Dense models like Contriever excel at bridging this gap by capturing semantic similarity: even if a query uses a synonym or acronym, the neural embedding may still align with passages containing the expanded form or a paraphrase. Indeed, Contriever's high

accuracy suggests it can retrieve relevant passages even when there is little lexical overlap, a crucial advantage for an open-domain web corpus with diverse expressions. Another factor is scale: with a million passages, many queries have multiple relevant answers. BM25, focusing on a few keywords, might retrieve one relevant passage but rank it lower due to other documents containing those keywords out of context (or it might miss the relevant passage if the wording diverges). Contriever's vector-based retrieval provides a broader semantic net, which in this case yields higher top-1 accuracy and much higher top-5/10 accuracy. DPR, which was trained on a different QA dataset, shows decent performance (better than BM25), indicating that training on question–passage pairs helps, but it does not match Contriever (that may be because of an overfit to the Wikipedia scientific style it was trained on). Likely, Contriever's unsupervised training on large general text made it more robust to the variety in MS MARCO. Another point is that MS MARCO queries can be ambiguous or underspecified; a dense model might encode latent context or popular interpretations of the query, effectively guessing the intent better than exact match. Overall, the results suggest that for a noisy, large-scale retrieval task like MS MARCO, semantic retrievers have a clear edge over traditional lexical methods. BM25's difficulty here underscores how real-world search queries ("free-form" questions, often incomplete) benefit from models that understand meaning beyond literal words.

3.2.2 SQuAD 2.0: Retrieval Results and Analysis

(The same issues encountered with MS MARCO also prevented the completion of experiments on the SQuAD dataset. Despite extensive efforts to restore compatibility, the ColBERT model consistently failed to run due to version conflicts across essential libraries. The automatic updates performed by Colab further aggravated the problem, preventing stable reproduction of the indexing and retrieval pipeline. Consequently, no ColBERT results could be obtained for SQuAD, which limited the scope of the empirical evaluation in this thesis.)

Next, I evaluate retrieval on SQuAD 2.0. Table 3.6 shows the top-k accuracy for BM25, Contriever, and DPR on this dataset. Here we see that BM25 and Contriever both perform very strongly, while DPR is behind. BM25 returns a relevant paragraph as the first hit for about 63.6% of the questions, and by 10 retrieved results it achieves roughly 85.3% accuracy. Contriever's performance is similar at rank 1 (about 61.7% accuracy@1, essentially on par with BM25), but it surpasses BM25 at higher k: around 86.5% of questions have a relevant paragraph in the top-5 results for Contriever, and about 92.5% by the top-10. DPR, on the

other hand, finds the correct paragraph at rank 1 only 41.0% of the time, and reaches 80.0% by top-10, notably lower than the other two methods.

Retriever	Top-1	Top-5	Top-10
BM25	**63.6%**	80.0%	85.3%
DPR	41.0%	69.2%	80.0%
Contriever	61.7%	**86.5%**	**92.5%**

Table 3.6: Retrieval accuracy on SQuAD 2.0. Values indicate accuracy@k.

Analysis: In the SQuAD setting, the retrieval task is relatively easier than in MS MARCO: questions are carefully worded and each has a specific Wikipedia paragraph that contains the answer. The high numbers reflect this — a well-tuned lexical engine or dense model can retrieve the correct paragraph for the majority of questions. Notably, SQuAD's passages have extremely low lexical diversity (TTR = 0.010), meaning many questions share key terms with their answer paragraphs. This repetitive terminology likely underlies BM25's strong performance. Its slight edge at rank 1 suggests that SQuAD questions often share exact keywords with their source paragraph. This makes sense, as crowdworkers who wrote the questions likely paraphrased the content in the passage but still mentioned the key entity names or terms. For example, if a SQuAD paragraph is about Mount Everest, the question is likely to include "Mount Everest" or a very close synonym. BM25 excels at this kind of direct term matching. Contriever, despite not having seen SQuAD during training, benefits from its general semantic comprehension and wide pre-training on Wikipedia. Its accuracy at top-5 and top-10 is excellent, indicating that it can pick up on paraphrases or alternate phrasings that BM25 might miss. The fact that Contriever slightly lags BM25 at rank 1 but overtakes by rank 5 suggests that in cases of exact term overlap, BM25 is immediately effective, but in cases requiring a bit of abstraction (synonyms or rephrasing), Contriever catches up. DPR's lower performance on the other hand, is difficult to explain, given that he was trained on passages from Wikipedia . In summary, SQuAD's results show that both lexical and neural unsupervised approaches can nearly solve the retrieval problem in a controlled domain with explicit query–answer term overlap. The small gap between BM25 and Contriever highlights that lexical methods remain very competitive when the language is clean and the queries are closely tied to the content. Dense retrieval still offers an advantage in accuracy by handling rephrased queries, but the benefit is less dramatic here than in MS MARCO's noisy setting.

3.2.3 ObliQA: Retrieval Results and Analysis

I then examine the retrieval performance on the ObliQA dataset. Table 3.7 summarizes the top-k retrieval accuracies of BM25, DPR, Contriever, and ColBERT on ObliQA. Perhaps surprisingly, the traditional lexical matcher BM25 significantly outperforms the two standard dense retrievers (DPR and Contriever) in this domain, yet the advanced ColBERT model in turn outperforms BM25. BM25 achieves about 53.0% top-1 accuracy (meaning it retrieves a relevant passage as the first result for 53% of the questions), markedly higher than Contriever's top-1 accuracy (approximately 42.6%) and more than double DPR's (19.6%). However, ColBERT yields the highest score, with 60% top-1 accuracy – a +7 point absolute improvement over BM25. This trend continues for broader retrievals: at top-5, ColBERT reaches about 70.7% accuracy, versus BM25's 64% (Contriever 57.1%, DPR 30.8%); by top-10, ColBERT attains around 74.7% accuracy, compared to BM25's 68.8%, Contriever's 64.1%, and DPR's 38.0%. In other words, ColBERT is strongest at accuracy for every ranks, while BM25 maintains a clear edge over the simpler dense models. These results indicate that in the legal/financial language of ObliQA, lexical matching signals are extremely important – but a sufficiently powerful semantic retriever can leverage those signals as well, surpassing even BM25.

Retriever	**Top-1**	**Top-5**	**Top-10**
BM25	53.0%	64.0%	68.8%
Contriever	42.6%	57.1%	64.1%
DPR	19.6%	30.8%	38.0%
ColBERT	**60%**	**70.7%**	**74.7%**

Table 3.7: Retrieval accuracy on **ObliQA** (regulatory text domain). Values indicate the proportion of questions for which a relevant passage was retrieved in the top k results (accuracy@k).

Analysis: The ObliQA domain's extreme linguistic complexity correlates with the distinct performance pattern observed. Its regulatory passages have by far the lowest readability (Flesch = 0, indicating very dense legal prose), with long, clause-heavy sentences and highly specialized vocabulary. In this low-readability, jargon-heavy context, BM25's reliance on exact token matching becomes a strength: any uncommon legal term present in the question can be directly matched to the same term in a candidate passage. This helps explain why BM25 tops the standard dense models here. In contrast, Contriever – being an unsupervised dense retriever – may not encode legal jargon or rare domain phrases effectively without domain-specific training. Indeed, ObliQA questions and passages are relatively long (re-

spectively averaging about 67 and 103 words each) and detailed, often containing multiple constraints or references. BM25 can latch onto any uncommon term or reference number in the query and find passages containing those exact tokens. Contriever's semantic similarity search, by compressing the entire text into a single embedding, might be hindered by the length and complexity of these queries: the model must condense a long, multi-faceted question into one vector, potentially losing some of the granular detail. Indeed, the embedding size used by neural encoders does not change by passage length, so information is lost when passages are longer (this is also true for DPR). Moreover, the language of ObliQA is very complex – the average sentence length is high and vocabulary very domain-specific (the text has a Flesch Reading Ease near 0, indicative of dense legal prose). In such text, subtle differences in wording can carry big changes in meaning (e.g. "shall" vs. "should" or the presence of a specific statute number). BM25's reliance on exact tokens can be an advantage here, as it does not miss those precise matches; a purely semantic model might consider two passages similar in meaning even if one does not contain the specific keyword that makes a passage relevant.

ColBERT, however, mitigates many of these issues. Unlike single-vector models, ColBERT uses a late interaction mechanism with token-level embeddings, which allows it to preserve important rare terms and match them directly between query and passage. This means that if a query contains a unique clause number or legal term, ColBERT can recognize that token's contribution distinctly (via exact or high-similarity token matches) instead of averaging it away. This ability likely enables ColBERT to capture the same domain cues that give BM25 its advantage, while also leveraging semantic context for terms that do not exactly match. In essence, ColBERT bridges lexical and semantic retrieval: it retains a strong focus on exact token overlap when it is informative, but can still generalize for paraphrased or contextually implied matches. This explains why ColBERT achieves the highest accuracy – it benefits from the precise term matching that ObliQA demands, and from semantic similarity on the parts of the query that use more common language.

DPR performs the worst on ObliQA, which is understandable given its training regime. DPR was originally trained on open-domain datasets like Wikipedia (e.g. Natural Questions), so the shift to highly formal, specialized legal language (without any fine-tuning on that domain) leaves DPR struggling. The vocabulary mismatch between general-domain text and formal regulations is extreme; DPR's learned embeddings likely fail to capture critical legal expressions or nomenclature that it never saw during training. Additionally, ObliQA often requires multi-passage reasoning: questions sometimes need information from two or more separate sections of text to answer. A dense retriever not specialized for multi-hop

reasoning may retrieve one relevant passage but miss the other. In contrast, BM25 can independently surface multiple relevant sections as long as the query contains some keywords from each, increasing the chance that at least one of the needed passages appears in the top-k. ColBERT's multi-vector approach similarly can handle multi-faceted queries better than single-vector models: because each query token can retrieve evidence, a complex question with multiple clues can trigger matches to different parts of the corpus. For example, if a question references two distinct regulatory provisions, BM25 or ColBERT might retrieve documents covering each provision among their top results, whereas a single-vector model might fixate on one aspect and ignore the other. Neither Contriever nor DPR inherently understands the structured format of ObliQA's text (numbered clauses, sections, etc.) unless those tokens influence their embeddings, so they don't explicitly benefit from structural cues like clause numbers – whereas BM25 can treat a clause number as just another token to match exactly.

Qualitatively, ObliQA retrieval results reflect these dynamics. BM25's top hits for these questions always contain obvious lexical overlaps with the query (often the exact regulatory subsection identifiers or unique domain terms present in the question). Not surprisingly, those overlaps usually correspond to the ground-truth relevant sections. Contriever, on the other hand, sometimes retrieves passages that are topically related to the query but not actually the specific provision asked about. For instance, if a question is about a particular compliance obligation in Section 3.6(2), Contriever might return a passage discussing general compliance procedures – semantically related to the topic, but not the correct clause that the question targets. This reflects a classic trade-off: Contriever casts a broader semantic net (indeed, by top-10 its accuracy nearly catches up to BM25's), but at the expense of accuracy at rank-1. In a domain like ObliQA, where questions demand pinpointing exact clauses, such semantic generalization can lead the model slightly astray for the top result.

ColBERT appears to overcome much of this trade-off. Thanks to its token-level matching, ColBERT usually retrieves the correct or very closely relevant passage at rank 1 and maintains high accuracy by rank 10. For example, in my experiments ColBERT would often return a passage containing the exact clause or obligation mentioned in the question as the first hit (similar to BM25's behavior on those queries). At the same time, ColBERT is less likely than Contriever to be misled by passages that are only vaguely related: if a candidate passage lacks the specific terms or phrases that are present in the query (and crucial for correctness), the query's corresponding token embeddings will not find high-scoring matches, and that passage's overall score will be lower. In summary, ColBERT manages to be accurate at every k on ObliQA, whereas Contriever skews more toward high-k and BM25 toward

low-k.

To conclude, for ObliQA's domain, lexical retrieval with BM25 was very effective, outperforming the initial dense retrievers. However, the inclusion of ColBERT shows that neural retrieval can match or even exceed lexical matching when the model is appropriately powerful. ColBERT's success here underscores that dense retrievers need not be handicapped by domain specificity – but they must be equipped to handle the domain's vocabulary and the query complexity (either through training or architecture). The simpler dense models (Contriever and especially DPR) struggled without any domain adaptation, highlighting the importance of domain matching. An sparse method like BM25 does not suffer from domain shift because it relies only on surface term overlap, whereas dense models must "know" the domain language to some extent. In this case, ColBERT's late-interaction design effectively gave it a robustness to domain terms that Contriever and DPR lacked. It is likely that further improvements for dense retrievers on ObliQA would require explicit domain-specific training or hybrid strategies (e.g. combining BM25's lexical strength with dense embeddings), as will be discussed in Chapter 4.

Strict vs Non-Strict Comparative Analysis

Retriever	Top-5 Strict	Top-5 Non-strict	Top-10 Strict	Top-10 Non-strict
BM25	47.1%	64.0%	48.5%	68.8%
DPR	21.9%	30.8%	23.7%	38.0%
Contriever	41.7%	57.1%	43.8%	64.1%
ColBERT	**52.8%**	**70.7%**	**54.2%**	**74.7%**

Table 3.8: Top-5 and Top-10 retrieval accuracy on ObliQA under strict and non-strict evaluation.

The performance of all retrievers drops significantly under the strict evaluation criteria on ObliQA. Strict evaluation in this multi-hop setting means that a question is counted as answered correctly only if all its supporting passages are retrieved, whereas the non-strict metric gives credit if at least one relevant passage is retrieved. As shown above, BM25's top-10 accuracy falls from 68.8% under the lenient metric to 48.5% under strict scoring – a drop of over 20 percentage points. Contriever and ColBERT exhibit a similar decline (each losing roughly 20 points in top-10 accuracy), indicating that they too often retrieve only partial evidence when answering ObliQA's multi-passage questions. DPR, which already lags in non-strict accuracy, suffers a smaller absolute drop (38.0% to 23.7%, about 14 points), but

this still represents the largest proportional reduction (nearly a 40% relative decrease). In other words, DPR frequently finds one relevant piece but misses the other required section(s), underscoring its difficulty with retrieving the complete set of evidence in this domain.

These strict vs. non-strict gaps highlight each model's sensitivity to multi-hop requirements and passage granularity. The lexical BM25, despite its strong non-strict performance, fails to retrieve all needed segments for many questions – likely when some clues or references are not explicitly shared between the question and the target passages. Dense semantic retrievers (DPR and Contriever) tend to focus on the query's overall topical gist and can overlook secondary details, leading to incomplete evidence retrieval under strict conditions. ColBERT's smaller penalty suggests it better captures multiple query aspects: its late-interaction mechanism can match different key terms or phrases in the query to different passages, making it more likely to retrieve both relevant documents within the top-k results. Overall, a large strict-versus-non-strict disparity (as seen especially with DPR) implies that a retriever may rely on broad semantic matching or single-hop reasoning—retrieving one highly relevant passage—whereas a smaller disparity (ColBERT) indicates more robustness in covering all facets of a multi-part query. This underscores the importance of a retriever's ability to handle semantic composition: models must not only retrieve something relevant to the question, but also ensure coverage of all required information to succeed in ObliQA's multi-hop setting.

3.2.4 CoQA: Retrieval Results and Analysis

I next consider the retrieval performance on the CoQA conversational dataset. Table 3.9 presents the top-k retrieval accuracies for CoQA. In this case, the ColBERT dense retriever emerges on top, achieving about 63.8% accuracy@1, 77.2% @5, and 79.8% @10. This substantially outperforms Contriever, which attains roughly 45.4%, 63.4%, 68.4% at the same ranks. BM25 lags behind both, with 35.8% @1 and up to 53.6% @10. DPR again shows the lowest scores (only 26.8% @1, improving to 48.0% by top-10), indicating significant difficulty with this dataset as well. Notably, ColBERT's advantage is most pronounced at rank 1: it finds a relevant passage as the very first result for nearly 64% of the questions, compared to 45% for Contriever and 36% for BM25. By top-10, ColBERT still maintains an 11–32 point lead in accuracy over the others (retrieving the correct story for about 80% of questions, vs. 68% for Contriever and 54% for BM25). These numbers clearly suggest that semantic matching is especially beneficial for CoQA queries, and that ColBERT's richer retrieval

mechanism yields a significant improvement in this conversational setting.

Retriever	Top-1	Top-5	Top-10
BM25	35.8%	48.8%	53.6%
Contriever	45.4%	63.4%	68.4%
DPR	26.8%	39.2%	48.0%
ColBERT	**63.8%**	**77.2%**	**79.8%**

Table 3.9: Retrieval accuracy on **CoQA** (conversational QA). The accuracy@k is the fraction of questions for which the correct story passage was retrieved in the top k.

Analysis: CoQA further illustrates how certain textual characteristics can hinder retrieval, especially for DPR. The conversational passages are easy to read (Flesch = 65), but the queries span an unusually wide vocabulary for their length (highest query TTR = 0.35). Each question often introduces new words or referents – including pronouns and implied context – which means there is often little direct lexical overlap with the answer text. Consider a scenario where earlier in the story a character named Mary was introduced, and later a question asks, "What did she do next?" The query itself has almost no lexical clue (the pronoun "she" is ambiguous and the word "next" is too generic). A purely lexical retriever like BM25 is likely to fail here: it will treat common words like "she" and "next" as keywords, which do not uniquely identify the relevant story, and thus it may retrieve irrelevant passages that coincidentally contain those words. In contrast, dense semantic retrievers shine in these scenarios. Both Contriever and ColBERT can leverage the contextual embeddings of the query to infer what it is asking for. Even without seeing the prior dialogue, the model's learned language patterns can associate "What did she do next?" with the concept of a sequence of actions by a female protagonist. ColBERT's encoder, in particular, can produce a query representation that captures the notion of "the next thing she did" and match it to a passage in the story where a female character's actions are described. Thus, semantic models implicitly use world knowledge and context understanding to handle the coreference and omission of explicit terms, whereas BM25 has no mechanism to bridge that gap. The results reflect this: the dense retrievers retrieve the correct story far more often in such cases (as evidenced by their much higher top-1 accuracy), indicating they understand the queries' intent better than BM25 for conversational questions.

Another factor in CoQA is that questions, while conversationally phrased, often paraphrase the story content. The wording of a question might not exactly match the text of the answer in the passage. For instance, a question might ask, "Was it raining?" when the story says "It started to drizzle." Here there is a clear vocabulary mismatch: the story never

uses the word "raining," so a lexical match would fail to find that passage. Contriever's dense embedding can map the concept of "raining" to "drizzle" in vector space, increasing the chance of retrieving the correct passage. ColBERT similarly can handle this paraphrase – during retrieval, the query token "raining" will still score highly with passage tokens like "drizzle" or "rain" due to semantic similarity in the BERT embedding space, even if not an exact string match. In essence, the dense models help mitigate vocabulary mismatch and synonyms, which are common in natural, conversational language. CoQA's multi-domain, multi-style nature (spanning childish story narratives, informal dialogue, etc.) means there is a lot of linguistic variation between queries and texts. An unsupervised model like Contriever, having been trained on a wide range of internet text, is relatively robust to such variation – and a model like ColBERT, built on a pretrained language model and fine-tuned on large QA data, is also able to generalize across domains and rephrasings. By contrast, BM25 has no ability to bridge vocabulary gaps beyond direct term overlap; if the question uses a different phrasing than the passage, BM25 will simply miss the connection. This explains why BM25's accuracy is so much lower on CoQA: many questions are phrased differently from the sentences that contain the answer, so without a direct overlap, BM25 often cannot retrieve the correct story at all.

The weaker performance of DPR on CoQA is likely due to two compounding reasons: domain mismatch and the conversational format. First, DPR's question encoder was trained on single-turn QA datasets (such as Natural Questions, based on Wikipedia). CoQA's questions, especially follow-ups, violate that assumption. The result is that DPR produces a query embedding that may not be close to the correct story at all, because the query by itself is ambiguous. The second issue is domain/style mismatch: CoQA includes various genres of text (fictional stories, spoken dialogue transcripts, etc.), which are very different in style from Wikipedia. DPR's original training on Wikipedia passages means its embeddings are tuned to that formal, encyclopedic style. Faced with children's story narratives or informal conversational text, DPR likely struggles to represent the query and passages in a compatible way. This exacerbates its errors. Only by increasing k does chance improve that the correct story might appear in the results; indeed, even at top-10 DPR finds the right passage for just 48% of the questions, far behind ColBERT's nearly 80%. This starkly underscores DPR's limitation in zero-shot transfer to conversational QA without additional training. In practical terms, DPR would likely need specialized fine-tuning or architectural changes to handle dialogue-style queries as well as the other models do.

Examining some concrete retrieval examples provides further insight. In cases where BM25 failed but the dense models succeeded, I often find that the question's critical clue was

expressed differently than in the text. For example, consider a CoQA conversation about a lamb. The story might say, "Its fleece was white as snow," and a question asks, "Did the little lamb's fleece stay clean?" Here the question uses the word "clean," whereas the story used "white" to imply cleanliness. BM25, looking for the word "clean" (or "cleaned"), would not find a match and thus could miss the relevant passage. Contriever or ColBERT, on the other hand, could infer that white as snow is an indication of cleanliness and retrieve that passage because the semantic meaning aligns with the question, even though the exact word "clean" is never used. Conversely, when BM25 succeeded and a dense model did not, the question usually contained an explicit keyword or name that uniquely matches a specific story. For instance, if a question mentions a character name or a distinctive entity that appears only in the correct story, BM25 will immediately zero in on that story by literal term matching. A dense retriever might occasionally overlook that exact match if it relies on general semantic similarity. I observed cases where Contriever retrieved a story that was topically similar but featured a different character with a similar role, essentially because it did not prioritize the rare proper noun in the query as heavily. ColBERT tends to be more robust in these scenarios. Since ColBERT scores matches at the token level, a unique name in the query will demand a matching token in the passage to achieve a high relevance score. In other words, if the question contains a specific name that is only in one story, ColBERT's MaxSim mechanism will give a large boost to that story (for having the name token overlap) which helps ensure it is retrieved. Thus, ColBERT recovers many of the exact-match successes of BM25, while still handling the paraphrased cases that BM25 misses.

Overall, for CoQA's conversational QA task, semantic retrieval methods dramatically outperform lexical matching. The dense retrievers (especially ColBERT, and to a strong extent Contriever as well) have a clear advantage in dealing with the linguistic phenomena inherent to dialogue: coreferences, ellipsis, and paraphrasing. BM25, lacking any ability to use context beyond literal tokens, struggles when queries are ambiguous or use different words than the text. My comparative analysis shows that a model like ColBERT, which combines fine-grained token interactions with learned semantic embeddings, can retrieve relevant story passages in a far more robust way for conversational questions – it retrieved correct contexts at nearly double the rate of BM25 at rank 1. Contriever also surpassed BM25 across the board, confirming that even without supervision, neural embeddings better capture the meaning of conversational questions. DPR's poor performance highlights the importance of architecture and training domain for dense models. In summary, the CoQA results emphasize that retrieving answers in a conversational setting requires models that handle context and synonyms; lexical keyword matching alone is insufficient. The gains from

ColBERT illustrate how much leveraging semantic context and rich interactions can improve retrieval in multi-turn QA. Looking forward, one might consider incorporating the conversation history explicitly or using hybrid retrieval techniques to further enhance performance on such tasks, but even with the question alone, it is evident that sophisticated dense retrievers are the key to high accuracy in conversational question answering retrieval.

3.2.5 HotpotQA: Retrieval Results and Analysis

Finally, I evaluate retrieval on HotpotQA. Table 3.10 shows the retrieval accuracy of the retriever models on HotpotQA. Overall, all methods (except DPR) perform quite well, reflecting that HotpotQA questions, while complex, do supply clear lexical clues. BM25 achieves about 66.4% accuracy@1 and 72.9% @10, meaning it retrieves a relevant paragraph as the top result for two-thirds of the questions and finds something relevant within the top-10 for roughly 73% of the queries. Contriever is on par with BM25: 66.9% @1 and 73.9% @10, essentially indistinguishable in terms of overall accuracy. ColBERT leads slightly, with about 73.8% @1 and 75.8% @10, the best among the models. Meanwhile, DPR significantly underperforms the others on this dataset (around 46.7% @1, improving to 61.9% @10). Notably, the gap between ColBERT/Contriever/BM25 and DPR is large here, whereas the gap amongst ColBERT, Contriever, and BM25 is relatively small compared to other datasets.

Retriever	**Top-1**	**Top-5**	**Top-10**
BM25	66.4%	71.7%	72.9%
DPR	46.7%	58.3%	61.9%
Contriever	66.9%	72.9%	73.9%
ColBERT	**73.8%**	**75.6%**	**75.8%**

Table 3.10: Retrieval accuracy on HotpotQA. Values indicate accuracy@k.

Analysis: HotpotQA's results reflect its mix of explicit clues and diverse content. Each question explicitly includes multiple entities or facts, so despite the passages being lexically rich (passage TTR 0.12), the query usually provides clear overlap terms for retrieval. For example, a question might ask, "Did the author of [Book X] ever collaborate with [Person Y]?" Here, the query itself includes "Book X" and "Person Y" as keywords. This explains why BM25 performs nearly on par with the neural models here – it will likely retrieve some paragraph about Book X or Person Y (or both). Indeed, BM25's solid performance (72.9% top-10 accuracy) indicates that in most cases at least one of the two relevant articles is retrieved via straightforward term matching. Contriever's equivalently strong performance suggests that semantic embedding does not confer a huge advantage in this scenario – probably because

there is not a severe vocabulary mismatch problem. The terms in the question ("author," the book title, the person's name) usually appear verbatim in the relevant pages, so BM25 is already doing a good job. Contriever can capture those as well, and perhaps in a few cases it does better when synonyms or indirect references occur, but those are not very frequent in HotpotQA questions (which were written to be answerable from specific Wikipedia sentences). ColBERT's slight edge (about +7 percentage points in top-1 over BM25) suggests that it is better at ranking the truly relevant paragraph to position 1. This could be because ColBERT can match on multiple tokens: for a bridging question, the relevant paragraph might be the one that mentions both clues (both Book X and Person Y). BM25 might retrieve one paragraph about Book X and another about Person Y and place them high, without knowing which is more directly useful. ColBERT, on the other hand, can give a higher score to a paragraph that contains both the book and the person, by accumulating evidence from multiple token matches via its late interaction mechanism. This would explain its higher accuracy at rank 1.

By top-5 or top-10, however, BM25 and Contriever catch up in accuracy, because between them they will have retrieved the needed pieces (even if scattered across results). The fact that all three methods reach around 73–76% top-10 accuracy implies that roughly a quarter of HotpotQA questions still have neither of their gold supporting passages in the top-10. These challenging cases might involve more subtle reasoning or names that have many distractors. For instance, if a question asks for a comparison ("Who is older, X or Y?"), the relevant passages each mention the birth date of X and Y. BM25 might retrieve lots of pages that mention X or Y without those specific facts. A dense model could theoretically use context ("age," "older") to prefer biographical passages, but my results show even ColBERT does not dramatically exceed BM25 in accuracy here. The very poor performance of DPR on HotpotQA is likely due to domain and task mismatch: DPR was trained on single-hop questions (NQ) which usually involve one Wikipedia page. It may not handle questions that are longer and have two distinct parts; it could be embedding the query in a way that focuses on one clue and ignores the other. In contrast, Contriever (though unsupervised) and ColBERT (supervised on MS MARCO) both seem capable of capturing multiple aspects of the query effectively.

In summary, HotpotQA demonstrates that straightforward lexical retrieval is already quite effective when queries explicitly contain the needed terms (making it more similar to a structured search). Dense retrieval provides a small benefit in prioritizing the most relevant combined-evidence passages (as seen by ColBERT's better rank-1 performance). However, the fact that no model exceeds 76% top-10 accuracy underlines the inherent difficulty of

multi-hop retrieval: a system might find one relevant piece easily but still miss the second piece. This suggests that additional techniques (like reranking or multi-step retrieval) might be needed to further improve performance on multi-hop questions. Nonetheless, among single-step retrievers, ColBERT performs best for HotpotQA, with Contriever and BM25 not far behind, and DPR significantly trailing. This trend reinforces the earlier observations: robustly-trained dense models (ColBERT) generally offer the best of both worlds, lexical methods (BM25) remain very competitive when query phrasing matches content, and dense models not tuned to the task (DPR) can struggle with complex queries.

Strict vs Non-Strict Comparative Analysis

Retriever	Top-5 Strict	Top-5 Non-strict	Top-10 Strict	Top-10 Non-strict
BM25	58.1%	71.7%	58.9%	73.0%
DPR	41.4%	58.3%	43.2%	61.9%
Contriever	59.3%	72.9%	60.1%	74.0%
ColBERT	**64.4%**	**75.6%**	**64.7%**	**75.8%**

Table 3.11: Top-5 and Top-10 retrieval accuracy on HotpotQA under strict and non-strict evaluation.

For HotpotQA, a similar pattern as the other multi-hop dataset (ObliQA) emerges: performance drops across all retrievers under the strict metric, though the magnitude of the penalty varies by model. In HotpotQA's two-hop questions, strict success requires retrieving both of the relevant Wikipedia paragraphs (supporting facts), rather than just one. Accordingly, DPR's top-10 accuracy declines from 61.9% (non-strict) to 43.2% (strict), a loss of nearly 19 points – the largest absolute drop among the retrievers. BM25 and Contriever each lose on the order of 14 percentage points on strict top-10 accuracy (falling from about 73% down to 59–60%), while ColBERT's performance decreases by only $\tilde{1}1$ points (75.8% to 64.7%). This indicates that DPR is the most penalized when requiring complete evidence, whereas ColBERT maintains the highest strict-score, suggesting it more often succeeds in retrieving both needed passages within the top ranks.

These differences reflect how each model handles HotpotQA's semantic bridges and multi-hop reasoning. DPR's heavy strict penalty implies it frequently retrieves one relevant piece (often the passage most directly related to the query) but misses the second, likely due to its single-vector query representation that may not preserve multiple distinct clues. BM25's moderate drop indicates that its keyword-based approach can retrieve both supporting documents when the question explicitly contains terms linking to each answer piece; however, it

struggles if a hop involves an implicit connection or paraphrased clue (e.g., a bridge entity or descriptor not directly mentioned in the query). Contriever, another dense semantic retriever, shows a strict-vs-non-strict gap similar to BM25's, suggesting that while it captures overall meaning well, it too can overlook one of the required passages if that passage's relevance is not strongly signaled in the query embedding. By contrast, ColBERT's relatively smaller gap points to an advantage in multi-hop retrieval: by matching query terms at a token level, it can more reliably surface both a bridging document and the final answer-bearing document within its top results. In summary, retrievers that rely purely on holistic semantic matching (DPR and to some extent Contriever) are more sensitive to the strict requirement of complete evidence, whereas models with robust lexical matching components (BM25 and ColBERT) cope better with queries that span disjoint pieces of information. Nonetheless, all models exhibit some strict-versus-non-strict disparity on HotpotQA, underlining that questions requiring the aggregation of multiple facts remain challenging for current retrieval methods.

Chapter 4

Conclusion

This study investigated how the linguistic profile of different datasets influence the performance of retrieval models by performing an empirical comparison. The findings clearly demonstrate that a retriever's performance is strongly conditioned by the nature of the dataset: no single model excelled universally, as each approach had advantages in certain contexts and weaknesses in others. In particular, dense neural retrievers proved highly effective for broad, open-domain queries by retrieving semantically relevant information beyond exact keyword matches, whereas the sparse BM25 method remained superior for specialized domains rich in unique terminology where exact term overlap was crucial. An unsupervised model (Contriever) exhibited notably robust cross-domain performance, surpassing a supervised dense retriever (DPR) that struggled without domain-specific training. The late-interaction model ColBERT offered the best of both worlds by combining semantic understanding with token-level precision, achieving the highest overall accuracy (albeit with increased computational cost). Collectively, these results confirm the central hypothesis that the linguistic properties of different textual domains significantly influence which retrieval strategy will be most effective.

From a practical perspective, these insights have direct implications for the design and deployment of retrieval-augmented generation systems. They suggest that practitioners should carefully consider the properties of their target domain when selecting or configuring a retriever. For instance, in domain-specific applications (such as legal or biomedical RAG systems), incorporating a lexical matcher like BM25 or fine-tuning a dense retriever on domain data may substantially boost retrieval accuracy, ensuring the language model receives the most relevant context. Conversely, for open-domain or heterogeneous knowledge bases,

neural semantic retrievers can provide broader semantic coverage. By improving the relevance of retrieved content, these practices ultimately enhance the factual correctness and reliability of the RAG system's outputs.

Finally, this work opens several avenues for future research. One direction is to explore retriever adaptation techniques – for example, further fine-tuning or continual pre-training of neural retrievers on target-domain corpora – which could help close the performance gap on specialized datasets. Another promising area is the development of more advanced hybrid retrieval frameworks, including dynamic ensembles or multi-stage pipelines that integrate lexical and semantic retrieval (and possibly a learned re-ranking stage) to better handle complex queries and multi-hop information needs. A third direction is to pursue deeper integration between retrieval and generative models by designing architectures where large language models themselves can perform context retrieval in a more integrated manner. Such approaches might enable the system to reason more effectively about what to retrieve and how to use it during generation. By building on the insights provided by this comparative study, future work can further enhance retrieval effectiveness and push the boundaries of what retrieval-augmented generation systems can achieve.

Bibliography

[1] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. *Proceedings of the Third Text REtrieval Conference (TREC-3)* (NIST Special Publication 500-226), 109–126. NIST. https://trec.nist.gov/pubs/trec3/t3_proceedings.html

[2] Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389. Now Publishers. https://www.researchgate.net/publication/220613776_The_Probabilistic_Relevance_Framework_BM25_and_Beyond

[3] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. https://www.cambridge.org/highereducation/books/introduction-to-information-retrieval/669D108D20F556C5C30957D63B5AB65C

[4] Connelly, S. (2018). Practical BM25—Part 2: The BM25 Algorithm and its variables. *Elastic Blog*. https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables.

[5] MyScale. (2024). Exploring Sparse Vectors vs BM25: Essential Contrasts. *MyScale Blog*. https://myscale.com/blog/key-differences-sparse-vectors-bm25/ (link does not work anymore, myscale.com domain is not referenced anymore, error 404).

[6] Luigi's Box. (2024). What Is BM25 (Best Match 25): Full Breakdown. https://www.luigisbox.com/search-glossary/bm25/.

[7] Mehta, S. (2024). Understanding Okapi BM25: A Guide to Modern Information Retrieval. *ADaSci*. https://adasci.org/understanding-okapi-bm25-a-guide-to-modern-information-retrieval/.

[8] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems (NeurIPS 2020)*, 33, 9459–9474. https://arxiv.org/pdf/2005.11401

[9] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-t. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of EMNLP 2020*, 6769–6781. Association for Computational Linguistics. https://arxiv.org/pdf/2004.04906

[10] Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P. N., Ahmed, J., & Overwijk, A. (2021). Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *International Conference on Learning Representations (ICLR 2021)*. https://arxiv.org/pdf/2007.00808

[11] Jégou, H., Douze, M., & Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1), 117–128. https://inria.hal.science/inria-00514462v2/document

[12] Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., & Grave, E. (2022). Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research (TMLR)*, 2022. https://arxiv.org/pdf/2112.09118

[13] He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum Contrast for Unsupervised Visual Representation Learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2020)*, 9729–9738. https://arxiv.org/pdf/1911.05722

[14] Khattab, O., & Zaharia, M. (2020). ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proceedings of SIGIR 2020*, 39–48. ACM. https://arxiv.org/pdf/2004.12832

[15] Santhanam, K., Khattab, O., Potts, C., & Zaharia, M. (2022). ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *Proceedings of NAACL 2022*, 3715–3734. Association for Computational Linguistics. https://arxiv.org/pdf/2112.01488

[16] Thakur, N. (2022). beir-ColBERT: Efficient Late Interaction for BEIR Evaluation. GitHub repository. https://github.com/thakur-nandan/beir-ColBERT.

[17] Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., Rosenberg, M., Shen, X., Soni, S., Wang, S., Wu, Y., Xiong, C., and Zhang, M. (2016). MS MARCO: A Human Generated MAchine Reading COmprehension Dataset.*arXiv:1611.09268*. https://arxiv.org/pdf/1611.09268

[18] Microsoft. (2025). MS MARCO Official Website. https://microsoft.github.io/msmarco/. [Accessed May 2025].

[19] Zhou Y. (2024). MSMARCOV2. GitHub repository. https://github.com/zhouyonglong/MSMARCOV2 [Accessed May 2025].

[20] Hugging Face. (2024). mteb/msmarco-v2 Dataset. https://huggingface.co/datasets/mteb/msmarco-v2 [Accessed May 2025].

[21] Rajpurkar, P., Jia, R., & Liang, P. (2018). Know What You Don't Know: Unanswerable Questions for SQuAD. *arXiv:1806.03822*. https://arxiv.org/pdf/1806.03822

[22] Balasubramanian, S. (2023). SQuAD 2.0: A Comprehensive Overview of the Dataset and Its Significance in Question Answering Research. *IJAIRD*, 1(1), 1–14. https://iaeme.com/MasterAdmin/Journal_uploads/IJAIRD/VOLUME_1_ISSUE_1/IJAIRD_01_01_001.pdf.

[23] Lee, G., Hwang, S., & Cho, H. (2020). SQuAD2-CR: Semi-supervised Annotation for Cause and Rationales for Unanswerability in SQuAD 2.0. *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC 2020)*, 5425–5432. European Language Resources Association. https://aclanthology.org/volumes/2020.lrec-1/

[24] Hugging Face. (2024). SQuAD v2 Evaluation Metric. https://huggingface.co/spaces/evaluate-metric/squad_v2.

[25] Hugging Face. (2024). rajpurkar/squad_v2 Dataset. https://huggingface.co/datasets/rajpurkar/squad_v2.

[26] RegNLP/ObliQADataset. (2024). GitHub Repository. https://github.com/RegNLP/ObliQADataset

[27] Gokhan, T., Wang, K., Gurevych, I., & Briscoe, T. (2024). RIRAG: Regulatory Information Retrieval and Answer Generation. *arXiv:2409.05677*. https://arxiv.org/pdf/2409.05677

[28] Reddy, S., Chen, D., & Manning, C. D. (2019). CoQA: A Conversational Question Answering Challenge. *Transactions of the Association for Computational Linguistics*, 7, 249–266. MIT PRESS. https://aclanthology.org/Q19-1016.pdf

[29] Stanford NLP Group. (2018). CoQA: A Conversational Question Answering Challenge. https://stanfordnlp.github.io/coqa/.

[30] Mandya, A., Bollegala, D., & Coenen, F. (2019). Evaluating Co-reference Chains based Conversation History in Conversational Question Answering. *PACLIC 31 (2019)*. https://livnlp.github.io/papers/Mandya_PACLING_2019.pdf

[31] Hugging Face Datasets. (2024). stanfordnlp/coqa. https://huggingface.co/datasets/stanfordnlp/coqa

[32] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., & Manning, C. D. (2018). HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *Proceedings of EMNLP 2018*, 2369–2380. Association for Computational Linguistics. https://nlp.stanford.edu/pubs/yang2018hotpotqa.pdf

[33] HotpotQA GitHub Repository. https://github.com/hotpotqa/hotpot

[34] Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019. https://arxiv.org/pdf/1901.04085

[35] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL), pages 6086–6096, 2019. https://aclanthology.org/P19-1000.pdf

[36] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., & Petrov, S. (2019). Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7, 452–466. MIT Press. https://aclanthology.org/Q19-1026.pdf

[37] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*. https://arxiv.org/pdf/1301.3781