ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

OPTIMIZING LISTWISE RERANKING WITH MODERNBERT: A DATA-CENTRIC APPROACH WITH A NOVEL GRADED CORPUS

CANDIDATE	SUPERVISOR
Mattia Maranzana	Paolo Torroni
	CO-SUPERVISOR
	Anna Elisabetta Ziri

Academic year 2024-2025

Session III

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ability to effectively retrieve relevant information from vast collections of documents is a cornerstone of modern information systems. Search engines, recommendation systems, and question-answering platforms all rely on sophisticated Information Retrieval (IR) techniques to meet user needs. A prevalent and highly effective strategy in modern IR is the multi-stage retrieval pipeline. This process typically begins with a fast, recall-oriented retriever model that scans a large corpus to identify a narrowed pool of potentially relevant passages. Following this initial stage, a more computationally intensive but significantly more accurate *reranker* model is employed to re-evaluate and reorder this candidate set, pushing the most relevant passages to the top. This two-stage approach balances efficiency with precision, and the reranking phase is critical for enhancing the quality of the final results.

The advent of large-scale, pre-trained language models has revolutionized the field of Natural Language Processing (NLP) and, by extension, the domain of information retrieval. Transformer-based models have demonstrated an unprecedented ability to understand the nuances of human language, thanks to their deep contextualized representations of text. These properties make them exceptionally powerful tools for tasks that require a profound understanding of semantic relevance, such as passage reranking.

Among these models, ModernBERT represents a recent and highly optimized evolution of the BERT architecture. It preserves the strengths of the original bidirectional Transformer design while introducing significant improvements in efficiency, scalability, and robustness. ModernBERT operates within the cross-encoder paradigm, where a query and a passage are processed jointly. This joint encoding allows for deep, token-level attention to model the intricate interactions between the query and the passage, resulting in rich combined representations. These representations can then be mapped to a relevance score through a learned prediction head. While classification heads are common, ModernBERT's flexibility allows researchers to experiment with different output layers and loss functions to maximize performance in reranking scenarios.

Despite the progress enabled by these models, the quality and structure of available datasets remain a critical factor for advancing research. For English, several benchmark datasets exist; however, many of them adhere to a rigid structure. Typically, they are built with a single positive (relevant) passage and a set of one or more negative (non-relevant) passages for each query. Although effective for evaluation, this design does not fully capture the complexity of real-world search scenarios, where multiple passages may exhibit varying degrees of relevance. Such structural limitations may bias models toward identifying only the single best passage, rather than learning to distinguish nuanced levels of relevance.

To address these limitations, this thesis introduces a new English passage reranking dataset. Unlike conventional resources, it is designed with a more varied and realistic structure, moving beyond the "one positive, many negatives" paradigm. By providing a richer spectrum of relevance annotations, the dataset enables a more robust and comprehensive evaluation of reranking models.

The central aim of this thesis is therefore twofold. First, we present the creation and characteristics of this novel English passage reranking dataset.

Second, we conduct an in-depth empirical study of ModernBERT, analyzing how architectural choices and training objectives affect its performance in reranking.

Chapter 2

The Foundations of Text Ranking

2.1 Early years

The central task of Information Retrieval (IR) is to satisfy a user's information need by providing relevant information. The most common formulation of this task is *ad hoc retrieval*, where a system must rank a collection of texts in response to a user's query. The goal is not merely to find documents containing the query's keywords, but to generate an ordered list where the texts at the top are most likely to be relevant to the underlying information need. This foundational challenge has driven decades of research, evolving from simple keyword matching to the sophisticated, context-aware models that define the current state-of-the-art.

The evolution of information retrieval has been shaped by a shift from *document retrieval* to *document ranking*. Early systems of the 1950s, rooted in Boolean logic, treated relevance as a rigid binary-documents either matched a query or not. This "go or no-go affair" failed to account for the ambiguity of natural language, often retrieving irrelevant results while missing relevant ones.

A decisive turning point came with Maron and Kuhns' [27]. They re-framed retrieval as a probabilistic inference problem, proposing that the task of an information retrieval system is not simply to identify matches, but to estimate the probability that a document D_i is relevant to a user's query containing index term I_j. Drawing on Bayes' theorem, they expressed this as:

$$P(D_i \mid I_j) = \frac{P(I_j \mid D_i) \cdot P(D_i)}{P(I_j)} \tag{2.1}$$

Here, $P(D_i \mid I_j)$ is the probability that document D_i is relevant given the occurrence of index term I_j; $P(I_j \mid D_i)$ represents the likelihood that a user interested in D_i would use term I_j in a query and $P(D_i)$ is the prior probability of D_i's relevance, reflecting its general utility. The denominator $P(I_j)$ serves as a normalizing constant.

This probabilistic formulation introduced the concept of a *relevance number*, a score quantifying the likelihood of relevance, and established the principle that documents should be *ranked by degrees of probable relevance* rather than returned as an unordered set.

Building on this foundation, Salton, et al. [45] introduced the *Vector Space Model* (VSM), which provided a geometric and algebraic framework for ranking documents. In this model, each document D_i is represented as a vector in a high-dimensional space spanned by the vocabulary terms:

$$D_i = (d_{i1}, d_{i2}, \ldots, d_{it}) \tag{2.2}$$

where d_{ij} denotes the weight of term T_j in document D_i.

Beyond the basic algebraic formulation, Salton et al. explored the geometric properties of the document space. They showed that retrieval effectiveness is inversely related to space density: effective indexing methods tend to produce configurations where relevant documents cluster together, while irrelevant ones are well separated . This insight laid the groundwork for techniques such as clustering, discrimination-based indexing, and refined term weighting. In particular, the introduction of TF–IDF weighting was a major innovation, as it emphasizes terms that are frequent in a given document but rare across the corpus, thereby "spreading out" document vectors in the space and improving their discriminability.

Building upon the TF-IDF paradigm, a significant thread of research throughout the 1980s and 1990s focused on optimizing term weighting strategies within the vector space framework. These approaches remained grounded in the principle of *exact term matching*, wherein a term must appear in both the query and the document to influence the relevance score [24]. While stemming and token normalization allowed for some flexibility in matching, the fundamental constraint was that only overlapping terms contributed to document ranking.

Formally, scoring functions based on exact term matching typically take the form:

$$S(q, d) = \sum_{t \in q \cap d} f(t) \tag{2.3}$$

where $f(t)$ is a function of term-level statistics such as:

- **Term Frequency (TF):** The number of times term t appears in document d.
- **Document Frequency (DF):** The number of documents in the corpus that contain term t.
- **Document Length:** The total number of terms in document d.

These statistics serve as the foundation for the widely-used TF-IDF weighting scheme, where terms are given more importance if they occur frequently in a document but are rare in the overall corpus. However, TF–IDF does not directly model document length normalization or term saturation, leading to further refinements in retrieval models.

One of the most successful and enduring advancements is the BM25 ranking function [42], which remains a strong baseline in both academic and industrial search systems. BM25 retains the core idea of exact term matching but incorporates more sophisticated handling of term frequency saturation and document length normalization.

The BM25 scoring function is defined as:

$$\text{BM25}(q,d) = \sum_{t \in q \cap d} \log \frac{N - \text{df}(t) + 0.5}{\text{df}(t) + 0.5} \cdot \frac{\text{tf}(t,d) \cdot (k_1 + 1)}{\text{tf}(t,d) + k_1 \cdot (1 - b + b \cdot \frac{l_d}{L})} \tag{2.4}$$

where N is the total number of documents in the corpus, df(t) is the number of documents containing term t, tf(t,d) is the frequency of term t in document d, l_d is the length of document d; L is the average document length in the collection and k_1 and b are tunable hyperparameters that control term frequency saturation and length normalization, respectively.

BM25 models two core intuitions: (1) additional occurrences of a term in a document increase relevance, but with diminishing returns; and (2) longer documents are more likely to contain a term by chance, so their scores should be normalized accordingly. The first component of the summation—the inverse document frequency (IDF)—downweights ubiquitous terms, while the second component scales the contribution of term frequency with respect to document length.

Although BM25 is often viewed as a heuristic, its empirical effectiveness has led to widespread adoption and many variants, including those implemented in open-source IR libraries such as Lucene. Importantly, BM25 forms the conceptual and empirical foundation upon which many neural and hybrid retrieval models are evaluated today.

2.2 Learning to rank

The advent of Learning to Rank (LTR) marked a significant evolution in information retrieval, characterized by the integration of supervised machine learning techniques with explicitly defined ranking objectives. Unlike traditional probabilistic and vector space models, where relevance estimation relied on fixed statistical heuristics, LTR approaches learn ranking functions from labeled data, often in the form of queries associated with graded relevance judgments. However, it is important to note that the term ’learning to rank’ does not simply refer to any supervised learning applied to ranking. Instead, it defines a distinctive research period and methodology within the IR community, particularly in the 2000s, where models were driven by sparse, hand-crafted features and optimization of ranking-specific objectives [24].

2.2.1 RankNet

A seminal work in this lineage is RankNet [8], which introduced a *pairwise learning-to-rank paradigm* based on probabilistic cost functions. RankNet learns to predict which of two documents is more relevant to a given query by modeling the probability of preference using a sigmoid function over a scoring model $f(\cdot)$, often implemented as a feed-forward neural network.

For a given query q, let U_i and U_j denote two candidate documents (for example, two different URLs retrieved for q). Each document U_k is represented by a feature vector $\mathbf{x}_k \in \mathbb{R}^n$, which encodes query-document interaction features such as term overlap, document length. RankNet maps each feature vector to a real-valued score through the learned function $f(\cdot)$, producing $s_i = f(\mathbf{x}_i)$ and $s_j = f(\mathbf{x}_j)$.

If the relevance label indicates that U_i should be ranked higher than U_j (e.g., U_i labeled “excellent” and U_j labeled “bad”), we denote this preference relation as $U_i \succ U_j$. The model then estimates the probability of this event by applying a sigmoid to the score difference:

$$P_{ij} \equiv P(U_i > U_j) \equiv \frac{1}{1 + e^{-\sigma(s_i - s_j)}}$$

This probabilistic formulation allows RankNet to be trained using cross-entropy loss between the predicted probability and the ground-truth preference, enabling the model to learn from relative relevance judgments without requiring absolute scores.

2.2.2 LambdaRank

RankNet established the foundation for learning-to-rank by modeling pairwise preferences, but it required evaluating all document pairs for each query, which introduced computational overhead. LambdaRank [7] addresses this limitation by avoiding the explicit definition of a cost function. Instead, it formulates gradients directly in terms of how much a ranking metric, such as Normalized Discounted Cumulative Gain (NDCG), would change if the positions of two documents were swapped.

NDCG is one of the most widely used metrics in information retrieval because it emphasizes placing highly relevant documents near the top of the ranking. For a ranked list of length k, it is defined as

$$\mathrm{DCG@}k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{2.5}$$

where rel_i denotes the graded relevance of the document at position i.

In LambdaRank, consider two documents U_i and U_j associated with the same query. If swapping their positions leads to a change $\Delta\mathrm{NDCG}$, the gradient contribution is defined as

$$\lambda_{ij} = \frac{\partial C(s_i - s_j)}{\partial s_i} = \frac{-\sigma}{1 + e^{\sigma(s_i - s_j)}} \cdot |\Delta\mathrm{NDCG}_{ij}| \tag{2.6}$$

where s_i and s_j are the scores assigned by the model, and σ is a scaling parameter. This expression ensures that the score of the more relevant document is pushed upward, while the less relevant one is pushed downward, in proportion to the potential improvement in NDCG.

By replacing the explicit loss with lambda-based gradients, LambdaRank enables direct and efficient optimization of non-differentiable ranking metrics such as NDCG.

2.2.3 LambdaMART

While RankNet and LambdaRank were originally implemented with neural networks, the underlying ideas are not restricted to them. In practice, tree-based models often offer better scalability and interpretability for ranking tasks. This leads to the exploitation of MART (Multiple Additive Regression Trees) [17], a gradient boosting framework that builds an ensemble of decision trees in a stage-wise manner.

The central idea of gradient boosting is to construct a strong predictor by iteratively adding weak learners, where each new tree is trained to correct the mistakes of the ensemble so far. Instead of optimizing the original loss function directly, MART fits each tree to the *residual errors* (or more precisely, the gradients of the loss) of the current model. In this way, the model gradually improves its predictions through successive refinements.

Formally, MART represents the final model after M boosting rounds as an additive combination of regression trees:

$$F_M(\mathbf{x}) = \sum_{m=1}^{M} \eta\, f_m(\mathbf{x}), \tag{2.7}$$

where each f_m is a regression tree, and η is a learning rate that controls how much each tree contributes. By repeatedly fitting new trees to capture the remaining errors, MART performs a form of gradient descent in function space. This combination of trees and boosting has proven to be highly effective across many supervised learning tasks.

Building on this, LambdaMART [6] combines the metric-aware gradient signals of LambdaRank with the powerful boosting mechanism of MART.

Instead of computing residuals from a differentiable loss, LambdaMART directly uses the λ-gradients from LambdaRank as the training signal. Intuitively, the model learns to grow trees that reduce ranking errors most harmful to evaluation metrics such as NDCG.

For each query, the pairwise λ values are first computed as in Equation 2.6, and then aggregated into a single gradient target λ_i for each document. These aggregated signals serve as the pseudo-residuals that guide MART's boosting procedure. Each tree is thus trained to predict how document scores should be adjusted to improve the target ranking metric.

In summary, LambdaMART inherits the ability of LambdaRank to align training with evaluation metrics, while leveraging the efficiency and robustness of boosted regression trees.

2.3 Deep Neural Ranking Models

The next major development in ranking research came with the application of deep learning techniques. Unlike earlier learning-to-rank approaches, which relied on carefully engineered features to capture query-document relationships, neural ranking models typically operate directly on distributed representations of raw text. This shift enabled models to automatically extract relevance signals without the need for hand-crafted input features, a departure that marked a clear break from the feature-based learning-to-rank era.

Broadly, early neural ranking models can be grouped into two families: representation-based and interaction-based architectures [24]. Representation-based models encode queries and documents independently into dense vector embeddings, after which a similarity measure such as cosine similarity or inner product is used to estimate relevance. In contrast, interaction-based models emphasize fine-grained matching by first computing pairwise token-level interactions between the query and document, typically represented as a similarity matrix. This matrix is then processed by additional neural layers to infer a final ranking score.

In both paradigms, deep architectures such as convolutional and recurrent neural networks were employed to capture sequential and compositional patterns in text.

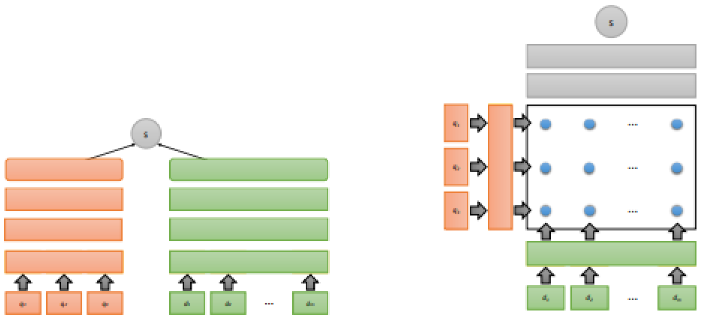


Figure 2.1: Representation-based and Interaction-based architectures [24].

2.3.1 Representation-Based Models

Representation-based models approach ranking by independently encoding queries and documents into dense vector embeddings within a shared semantic space. Relevance is then estimated by applying a similarity function directly to the resulting embeddings. In this setup, the interaction between query and document is not modeled explicitly at the term level; instead, all necessary information is assumed to be captured in the learned representations themselves.

The main appeal of this paradigm is its simplicity and efficiency. Once embeddings are computed, relevance scores can be obtained through lightweight similarity calculations, making such models highly scalable to large document collections. However, the independence of query and document encoding can limit expressiveness, as fine-grained term-level interactions are not directly captured.

Deep Structured Semantic Model

A representative example of the representation-based paradigm is the Deep Structured Semantic Model (DSSM) [20]. The central idea is to map both queries and documents into a common low-dimensional semantic space, such that their similarity can be directly measured. Relevance is then computed using cosine similarity between the query and document embeddings in this shared space.

Formally, both queries and documents are first represented as high-dimensional term vectors, typically based on raw counts of words. Passing these vectors directly to a neural network is impractical due to vocabulary size, which in web search can reach hundreds of thousands of unique terms. To address this, DSSM introduces a word hashing step: words are decomposed into letter n-grams (e.g., trigrams) and represented as fixed-dimensional vectors. This technique reduces input dimensionality while also providing robustness to rare and out-of-vocabulary terms.

Training leverages large-scale clickthrough data, where clicked documents are assumed to be more relevant than unclicked ones for the same query. Given a query q and its clicked document d^+ among a set of candidates $\{d^+, d_1^-, \ldots, d_n^-\}$, the model maximizes the conditional likelihood of d^+ via a softmax over similarity scores:

$$P(d^+|q) = \frac{\exp(\gamma\,\mathrm{rel}(q, d^+))}{\sum_{d' \in \{d^+, d_1^-, \ldots, d_n^-\}} \exp(\gamma\,\mathrm{rel}(q, d'))}, \tag{2.8}$$

where γ is a smoothing parameter and $rel(q, d)$ is the cosine similarity between the query vector and the document vector. Through this design, DSSM demonstrates how representation-based models can align query and document semantics in a dense vector space, removing the dependence on manual feature engineering and providing scalability to large vocabularies through hashing.

Convolutional Latent Semantic Model

While representation based approaches such as DSSM encode queries and documents independently, they fail to capture local contextual dependencies. To address this limitation, Convolutional Latent Semantic Models (CLSM) [49] introduced convolutional and pooling layers to explicitly model word-level interactions within a sequence.

The architecture of CLSM consists of several key components. First, each word in a query or document is represented using letter-trigram hashing, as in DSSM, which provides a compact and generalizable encoding of vocabulary items. These hashed word vectors are then concatenated within a sliding context window to form n-gram representations.

A convolutional layer applies shared linear transformations to these n-gram vectors, producing local contextual feature representations. This step ensures that semantically similar word n-grams, even if unseen during training, are mapped to nearby points in the feature space. To aggregate these local features into a fixed-length representation, CLSM employs max pooling, which selects the most salient activations across the sequence. The intuition is that only a subset of word n-grams are crucial for capturing the semantics of a query or document, and max pooling highlights these discriminative patterns.

The pooled sentence-level vector is then passed through a fully connected layer to obtain the final latent semantic embedding. Relevance between a query q and a document d is computed as the cosine similarity between their semantic vectors.

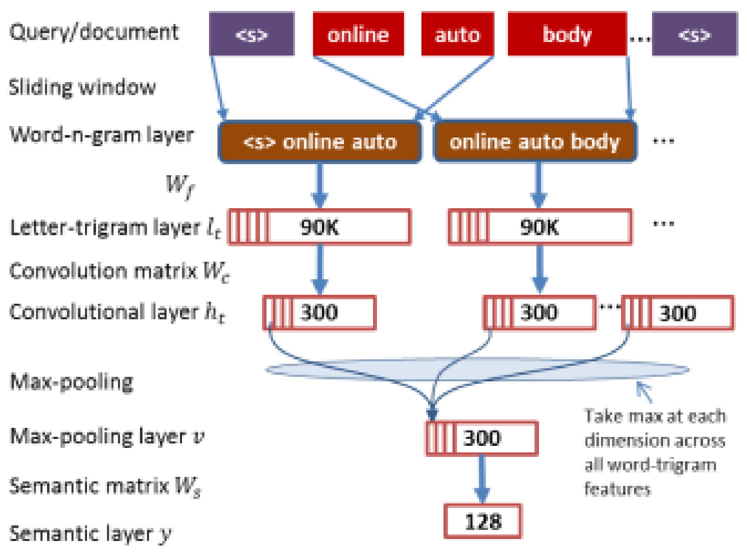


Figure 2.2: Convolutional Latent Semantic Model Architecture [49].

Dual Embedding Space Models

Another line of work explored leveraging distributional semantics for ranking. A prominent example is the Dual Embedding Space Model(DESM) [30], which builds on the Continuous Bag-of-Words (CBOW) variant of Word2Vec [28]. Unlike the standard usage of Word2Vec where only the input embedding matrix is retained after training, DESM explicitly utilizes both the input (IN) and output (OUT) embedding spaces.

The core motivation of DESM is to better capture whether a document is about a query term, rather than merely containing it. In CBOW, the IN space tends to encode functional similarity (words of the same type, e.g., “Harvard” and “Yale”), while the OUT space captures topical co-occurrence (e.g., “Yale” and “faculty”). By mapping query words into the IN space and document words into the OUT space, DESM exploits this complementary structure to measure topical alignment between queries and documents.

Formally, each document D is represented by the centroid of its normalized word embeddings:

$$\mathbf{D} = \frac{1}{|D|} \sum_{d_j \in D} \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|}, \tag{2.9}$$

where $\mathbf{d}_j$ denotes the embedding of word d_j. Relevance between a query Q and document D is then defined as the average cosine similarity between each query word embedding and the document centroid:

$$\mathrm{DESM}(Q, D) = \frac{1}{|Q|} \sum_{q_i \in Q} \frac{\mathbf{q}_i^{\top} \mathbf{D}}{\|\mathbf{q}_i\| \, \|\mathbf{D}\|}. \tag{2.10}$$

This formulation allows pre-computation of document centroids, making the model efficient for large-scale retrieval.

2.3.2 Interaction-Based Models

In contrast to representation-based approaches, which encode queries and documents independently, interaction-based models emphasize fine-grained matching patterns between their terms. As illustrated in Figure 2.1, these models typically construct a similarity matrix where rows correspond to query terms and columns to document terms. Each entry m_{ij} in this matrix encodes the similarity between the embedding of the i-th query term and the j-th document term.

At a high level, interaction-based architectures operate in two stages: feature extraction and relevance scoring. During feature extraction, the similarity matrix is processed to highlight relevance signals emerging from term-level alignments. In the relevance scoring stage, the extracted features are aggregated through pooling and non-linear transformations to produce a compact representation. This representation is then fed into a feed-forward network to compute the final query–document relevance score. Through this two-step design, interaction-based models explicitly operationalize the intuition that effective ranking depends not only on overall semantic similarity but also on the detailed structure of query–document interactions.

Deep Relevance Matching Model

Among interaction-based architectures, a well-known example is the Deep Relevance Matching Model (DRMM) [18]. The authors argue that ad-hoc retrieval differs fundamentally from other NLP matching tasks in three ways:

1. **Exact matching signals:** the presence of exact query terms in a document remains a dominant indicator of relevance.
2. **Query term importance:** since queries are typically short and keyword-based, different terms contribute unequally to relevance.
3. **Diverse matching requirements:** relevance may occur in specific passages of long documents, without requiring global semantic alignment.

DRMM does not encode queries and documents independently before comparison, Instead it explicitly constructs local interactions between each query term and all document terms, represented as similarity scores. These raw interactions are then aggregated into matching histograms, which capture the distribution of similarity strengths. Neural layers operate directly on these interaction features, rather than on global query/document embeddings. In this way, DRMM models the query–document relationship at the token-interaction level.

For each query term q_i, the model computes cosine similarities with every document term d_j:

$$\mathrm{sim}(q_i, d_j) = \frac{\mathbf{q}_i^\top \mathbf{d}_j}{\|\mathbf{q}_i\| \, \|\mathbf{d}_j\|}. \tag{2.11}$$

This yields a set of similarity scores between q_i and all document terms. Rather than preserving term positions, DRMM discretizes these scores into a fixed number of bins spanning $[-1, 1]$, with exact matches (i.e., similarity $= 1$)

treated as a dedicated bin. The histogram for q_i is then obtained by counting how many similarity values fall into each bin.

For example, consider the query term “virus” and document terms $\{$virus, infection, software, cure, data$\}$, with corresponding cosine similarities $(1.0, 0.73, 0.12, 0.55, -0.20)$. Using five bins $[-1, -0.5), [-0.5, 0), [0, 0.5), [0.5, 1), [1]$, the histogram is $[0, 1, 1, 2, 1]$. This fixed-length vector compactly represents the strength distribution of interactions, distinguishing exact matches from weaker semantic similarities.

Each query-term histogram is fed into a feed-forward matching network, which learns hierarchical patterns of interaction strengths and outputs a relevance score $z_i^{(L)}$. To incorporate term importance, a term gating network assigns a weight g_i to each query term. The final query–document relevance score is the weighted sum:

$$s(q, d) = \sum_{i=1}^{M} g_i \, z_i^{(L)}. \tag{2.12}$$

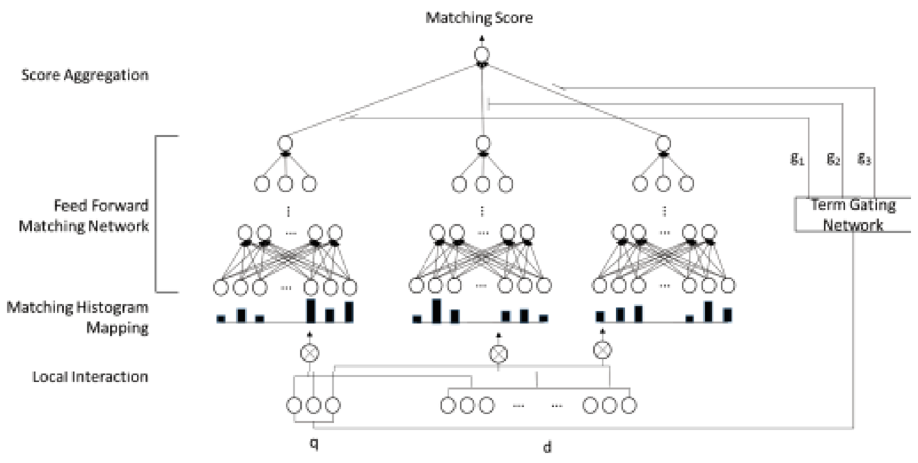


Figure 2.3: Deep Relevance Matching Model Architecture [18].

MatchPyramid

Another representative interaction-based architecture is the MatchPyramid model [38] which explicitly constructs a two-dimensional matching matrix that captures all pairwise similarities between query and document terms:

$$M_{ij} = w_i \otimes v_j, \tag{2.13}$$

where w_i and v_j are embeddings of the i-th query and j-th document terms, and $\otimes$ denotes a similarity function such as indicator, cosine, dot product, or Gaussian kernel. This interaction matrix can be interpreted as an "image" whose pixels encode token-level similarity signals.

To extract higher-level interaction patterns, the similarity matrix is passed through a series of convolutional and dynamic pooling layers. Convolutional kernels capture local structures such as n-gram matches and proximity patterns, while pooling layers reduce dimensionality and emphasize the most salient matching signals. For ad-hoc retrieval, pooling by paragraph length in the document was shown to be particularly effective at filtering out noisy background terms while retaining informative signals.

The high-level feature maps are then flattened and fed into fully connected layers to produce a final query-document matching score as shown in the following image.

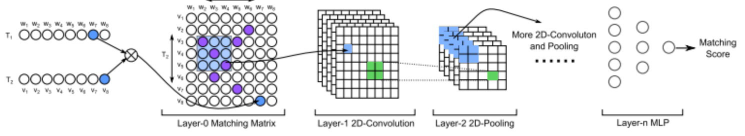


Figure 2.4: MatchPyramid Pipeline [38].

2.3.3 Hybrid Approaches

While interaction-based and representation-based models capture different aspects of query–document relevance, they are not mutually exclusive. Hybrid models attempt to combine the strengths of both paradigms, leveraging the fine-grained matching signals of interaction-based methods alongside the semantic generalization ability of representation-based methods.

The Duet Model

The central hypothesis of the Duet model [29] is that exact term matches (local representations) and semantic matches (distributed representations) provide complementary signals for ad-hoc retrieval. To operationalize this idea, the Duet architecture consists of two distinct deep neural subnetworks trained jointly:

The local component estimates relevance from patterns of exact term matches and their positions within the document. Each query and document term is represented as a one-hot vector, and an interaction matrix X is constructed:

$$X = D^{\top} Q, \tag{2.14}$$

where Q and D denote the one-hot matrices of query and document terms, respectively. This matrix encodes exact matches between terms and preserves positional information. Convolutional layers are then applied to capture proximity and clustering effects, followed by fully connected layers producing a local relevance score $f_{\ell}(Q, D)$.

The distributed component learns dense representations of text via character n-gram encodings. Queries and documents are projected into embedding spaces through convolution and pooling layers, resulting in continuous vector representations that capture semantic relatedness beyond surface term overlap. The query and document embeddings are combined via a Hadamard product, and the resulting features are processed by fully connected layers to output a distributed relevance score $f_d(Q, D)$.

The final Duet score is the sum of the two subnetworks:

$$f(Q,D) = f_\ell(Q,D) + f_d(Q,D). \tag{2.15}$$

Training is performed using a softmax loss over relevant and non-relevant documents, with human-judged labels as supervision. Importantly, the two subnetworks are optimized together, allowing them to complement one another rather than compete.

The Duet model demonstrates how hybrid architectures can effectively unify the strengths of interaction-based and representation-based approaches. Its distributed subnetwork excels at capturing semantic similarity for frequent queries, while its local subnetwork provides robustness on rare or tail queries where exact term matching is crucial. By combining these complementary capabilities, the Duet model consistently achieves strong performance across different query types.

2.4 The advent of Transformers

The introduction of the Transformer architecture [54] marked a decisive shift in information retrieval and text ranking. Unlike recurrent or convolutional models, which process sequences sequentially or locally, transformers rely on self-attention mechanisms to compute contextualized representations in parallel. This design enables modeling of long-range dependencies, efficient training on large corpora, and unprecedented scalability.

The release of BERT [15] demonstrated that pretrained transformers could be fine-tuned on downstream ranking tasks with relatively little supervised data, yet achieve dramatic gains over prior deep neural ranking models.

The advent of transformers did not merely improve ranking accuracy: it restructured the architecture of retrieval systems. Modern search pipelines increasingly rely on a retriever-reranker division of labor, where transformers dominate the reranking stage. This transition sets the stage for the next chapter, which examines multi-stage architectures in detail.

The next chapter will examine these multi-stage architectures in detail, beginning with early cross-encoder approaches such as BERT-based rerankers, and then tracing the evolution toward the T5 family of models (MonoT5, DuoT5, and ListT5) that reformulated reranking as a text-to-text task. We will then consider the more recent paradigm of using large language models (LLMs), including GPT-style architectures, as powerful listwise rerankers. Finally, the chapter will introduce ModernBERT, which combines the efficiency of compact architectures with the representational strength of transformers, and which will serve as the primary reranker investigated in this thesis.

Chapter 3

Transformer Architectures for Reranking

Building upon the foundations of classical retrieval models and the advent of transformers discussed in the previous chapter, this section focuses on the architectures specifically designed for passage reranking. While early ranking approaches relied on feature engineering or shallow neural networks, transformer-based models introduced a paradigm shift by jointly modeling queries and candidate passages through deep contextualized representations.

In this chapter, we first examine cross-encoder architectures, where queries and passages are concatenated and processed together, highlighting their strengths and computational trade-offs. We then move to sequence-to-sequence frameworks, such as the T5 family, which reformulate reranking as a text-to-text generation task. Finally, we consider the latest developments, including large language models (LLMs) and ModernBERT, the architecture at the core of this thesis.

3.1 BERT-based Rerankers

Before discussing BERT as a reranker, we first provide an overview of the model itself. BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) was introduced by Devlin et al. [15]. BERT leverages a deep bidirectional Transformer encoder pre-trained on large corpora using two self-supervised tasks:

- **Masked Language Modeling (MLM)**: predicting randomly masked tokens given their bidirectional context.
- **Next Sentence Prediction (NSP)**: predicting whether a sentence B follows sentence A in the corpus.

These objectives enable BERT to learn rich contextualized representations that capture both local and long-range dependencies. The pre-trained model can then be fine-tuned with minimal architectural changes, simply by adding a small output layer for the target task. This paradigm shift significantly reduced the need for heavily engineered task-specific models.

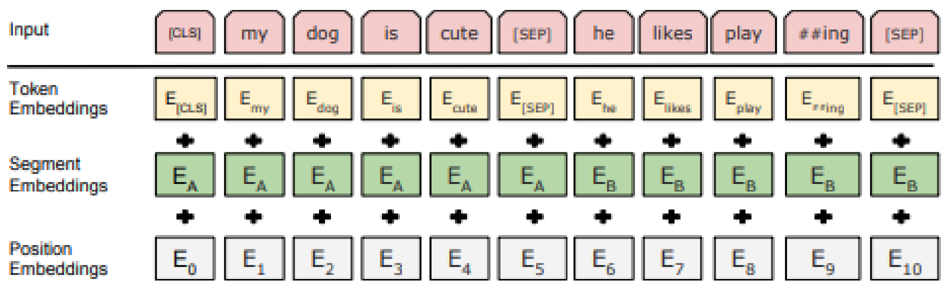

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3.1: BERT input representation: token embeddings are combined with segment and position embeddings [15]).

Figure 3.1 illustrates how BERT represents input sequences. Each token is represented as the sum of three embeddings:

- **Token embeddings**: obtained from a WordPiece [58] vocabulary of about 30,000 subword units, ensuring that rare words are decomposed into smaller, more frequent pieces.

- **Segment embeddings**: used to differentiate between sentence A and sentence B when the input consists of a pair (e.g., question and passage). All tokens from sentence A share one segment embedding, and those from sentence B share another.

- **Position embeddings**: encode the absolute position of each token in the sequence, allowing the otherwise order-invariant self-attention mechanism to take word order into account.

Special tokens further structure the input: `[CLS]` is placed at the beginning of every sequence and provides a holistic representation of the input (commonly used for classification tasks), while `[SEP]` marks the boundary between sentences or the end of the sequence.

BERT was released in two main configurations: BERT-Base (12 layers, hidden size 768, 12 attention heads, 110M parameters) and BERT-Large (24 layers, hidden size 1024, 16 heads, 340M parameters). Pre-trained on BooksCorpus (800M words) and English Wikipedia (2,500M words), these models achieved state-of-the-art results across a wide range of NLP benchmarks with only modest task-specific fine-tuning.

A detailed survey of BERT and its many variants is given in Rogers et al. [43]. The survey reviews over 150 studies, analyzing what linguistic and world knowledge BERT encodes, how this information is distributed across layers, and which probing methods can extract it. Probing studies suggest that BERT embeddings capture syntactic trees, semantic roles, and even certain types of factual knowledge. However, the model often relies on superficial cues, and its success does not necessarily imply deep linguistic understanding. For example, BERT has been shown to perform well on subject–verb agreement tasks, yet it may still produce plausible outputs for sentences with scrambled word order or semantically incoherent content.

The survey also highlights several important limitations. Although later large language models scale to billions of parameters, BERT itself was already considered over-parameterized: it contains many redundant components, and studies demonstrated that comparable performance can often be achieved after pruning attention heads, compressing layers, or distilling knowledge into smaller models. This redundancy makes the architecture resource-intensive to train and deploy relative to its effective capacity, motivating a wave of research into compression and efficiency. Beyond size, other limitations are methodological. Attention weights, for instance, are frequently used as an interpretability signal, but the connection between attention patterns and linguistic structure remains ambiguous, and visualizations can be misleading. Moreover, BERT demonstrates brittleness: it can achieve high benchmark scores yet fail under input perturbations such as shuffled word order, adversarial triggers, or logically inconsistent sentences. Finally, while BERT encodes a surprising amount of syntactic, semantic, and factual knowledge, it struggles with multi-step reasoning, robust handling of negation, and generalization beyond surface patterns.

3.1.1 BERT as a Passage Reranker

While BERT was originally introduced as a general-purpose language understanding model, its effectiveness as a reranker in information retrieval tasks was quickly established. Nogueira and Cho [32] were the first to demonstrate that a fine-tuned BERT model could substantially improve passage re-ranking performance on large-scale benchmarks. In their setup, the query was provided as sentence A and the candidate passage as sentence B. The `[CLS]` embedding was then used as input to a classification layer, yielding the probability that the passage is relevant to the query. Despite its simplicity, this approach surpassed previous state-of-the-art neural ranking methods by large margins, establishing BERT as a strong baseline for retrieval research.

A key reason the `[CLS]` token works in this setup is that, through BERT's self-attention mechanism, it learns to aggregate information from all tokens in the input sequence, both from the query and the passage. During fine-tuning, supervision from relevance judgments pushes the `[CLS]` representation to encode features predictive of query–document matching. Intuitively, the `[CLS]` token acts as a compressed interaction space: when strong semantic or lexical alignments exist between query and passage tokens, these signals propagate into `[CLS]`, allowing a shallow classification head to map this vector to a scalar relevance score. In this way, the `[CLS]` token serves as a global summary of query–passage interactions that can be directly transformed into ranking scores.

MonoBERT-DuoBERT

Building upon this work, Nogueira et al. [35] proposed a multi-stage document ranking architecture that integrates BERT at different levels of the pipeline. Their framework introduced two key variants: *monoBERT*, which applies BERT in a pointwise manner to score individual query–document pairs, and *duoBERT*, which adopts a pairwise classification strategy by jointly modeling two candidate passages with respect to the same query. By combining these models in a cascaded ranking pipeline, the system achieves both high effectiveness and controllable latency.

Figure 3.2 illustrates this multi-stage ranking architecture. An initial candidate set is retrieved using BM25 (H_0), after which monoBERT (H_1) scores each candidate independently. The top-k candidates are then passed to duoBERT (H_2), which refines the ranking by explicitly modeling pairwise preferences between documents. This integration of BERT into classical retrieval pipelines marked a turning point, showing that pretrained language models can be successfully adapted to large-scale ranking tasks with significant gains in retrieval quality.

A crucial aspect of this design is the way queries and candidate passages are encoded as BERT inputs. In monoBERT, the query q is placed in segment A and the passage d_i in segment B, with the total length truncated so that the

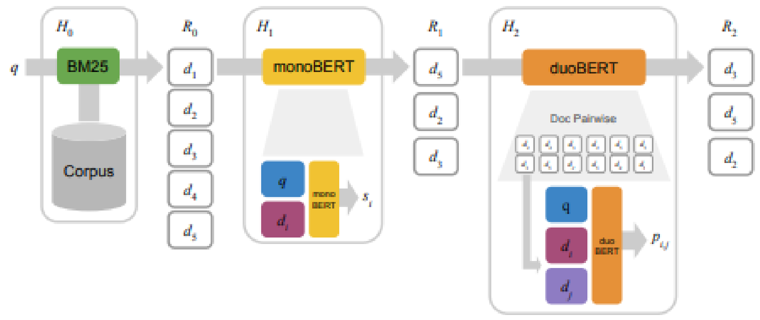


Figure 3.2: Multi-stage ranking with BM25, monoBERT, and duoBERT [35].

concatenated sequence (including special tokens) does not exceed 512 tokens. Typically, queries are capped at 64 tokens, while the remaining budget is allocated to the passage. The [CLS] token at the beginning serves as a summary representation and is fed into a classification layer to predict the probability of relevance. In duoBERT, the input is extended to three segments: the query as sentence A, candidate d_i as sentence B, and candidate d_j as sentence C. To fit within the same 512-token limit, the query is truncated to 62 tokens, and each candidate passage to 223 tokens. This careful segmentation ensures that BERT jointly models the interactions between queries and documents, or between document pairs, while respecting its architectural constraints.

Although BERT-based rerankers provide large gains in retrieval effectiveness, their computational cost is prohibitive if applied directly to an entire corpus. Each query–document interaction requires a full forward pass through a large transformer model, which becomes infeasible when millions of candidate documents are considered. For this reason, an initial stage based on a lightweight and efficient retrieval method, such as BM25, remains indispensable. BM25 serves as a high-recall filter that quickly selects a manageable set of candidates (e.g., the top 1000 documents) from the corpus. Subsequent neural rerankers can then be applied only to this reduced candidate set, balancing efficiency with effectiveness. In this way, the classical inverted-index retrieval stage and modern transformer-based rerankers complement each other:

BM25 ensures scalability, while BERT provides semantic precision in the final ranking.

3.1.2 Limitations of BERT for Long Passages

A notable limitation of monoBERT is that it does not offer an obvious solution to the input length restrictions of BERT. Nogueira and Cho [32] did not face this issue in their early experiments, since the benchmark collections they examined consisted primarily of short passages that easily fit within BERT's maximum sequence length of 512 tokens. However, in realistic ad hoc retrieval scenarios such as ranking news articles, reports, or web pages documents are often substantially longer. In such cases, truncation is unavoidable and directly affects effectiveness.

This constraint arises from two intertwined factors:

1. **Architectural and Pretraining Constraints.** BERT was pretrained with maximum sequences of 512 tokens, meaning that positional embeddings for indices beyond this length were never learned. Since position embeddings are the only source of linear order information in the model, inputs exceeding this limit lose structural cues.

2. **Computational Complexity.** Self-attention in BERT scales quadratically in both time and memory with respect to sequence length, making naïve extensions to longer inputs computationally prohibitive

While these limitations appear fundamental, research has demonstrated that BERT can still be applied effectively to longer inputs through segmentation and aggregation strategies. Instead of discarding overflow tokens, passages can be divided into smaller spans (e.g., sentences or sliding windows) that are each paired with the query and processed independently.

One line of work proposes aggregating sentence-level evidence. Akkalyoncu Yilmaz et al. [2] showed that applying BERT at the sentence level and then combining the top-scoring sentences yields strong retrieval effectiveness. This approach is motivated by the observation that even in long passages, relevance is often concentrated in a small portion of text; identifying and leveraging these "best" sentences provides a practical workaround to the length barrier.

Another strategy is to apply BERT to fixed-width passage segments and then aggregate their scores. Dai and Callan [13] explored three simple yet effective aggregation methods:

- **FirstP**: using the score of the first segment,
- **MaxP**: using the score of the most relevant segment,
- **SumP**: summing the scores of all segments.

Among these, MaxP often performs best, reflecting the intuition that a single highly relevant span can dominate the overall relevance judgment of a passage.

Together, these approaches show that even though monoBERT is naturally constrained to short inputs, segmentation combined with evidence aggregation allows BERT-based rerankers to handle longer passages effectively. This not only preserves semantic precision but also opens the door to more robust reranking in realistic retrieval scenarios where passages often exceed the 512-token window.

PARADE: Passage Representation Aggregation for Document Reranking

Building on these ideas, Li et al. [23] proposed PARADE (**P**assage **R**epresentation **A**ggregation for **D**ocument R**E**ranking), which moves beyond score aggregation by combining *representations* of query–passage pairs. Each passage is paired with the query and encoded by BERT, producing a `[CLS]` embedding. PARADE then applies aggregation mechanisms such as pooling, attention, CNNs, or transformers to integrate these embeddings into a holistic representation before predicting relevance. This enables the model to capture not only the strongest local relevance signals but also their distribution and dependencies across multiple spans.

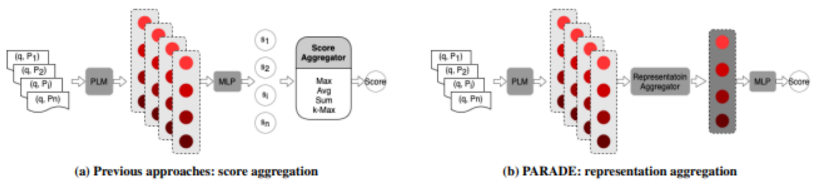

(q, P1)
(q, P2)
(q, Pi)
(q, Pn)
PLM
MLP
s1
s2
si
sn
Score Aggregator
Max
Avg
Sum
k-Max
Score
Representatoin Aggregator
(a) Previous approaches: score aggregation
(b) PARADE: representation aggregation

Figure 3.3: Comparison between score aggregation and PARADE's representation aggregation [23]

.

Although PARADE was originally proposed for document reranking, its central idea is equally relevant for long passage reranking: by aggregating multiple truncated segments at the representation level, it preserves more contextual information than simple truncation or max-score strategies. PARADE thus illustrates how segmentation and aggregation can evolve from simple heuristics to end-to-end trainable architectures that overcome BERT's context window limitations.

3.2 T5 and Text-to-Text Reranking

3.2.1 Introduction to T5

The Text-to-Text Transfer Transformer (T5) [39] was proposed as a unified framework for transfer learning in NLP. Unlike previous approaches that designed models and objectives around specific tasks, T5 casts every problem into a *text-to-text* format. This means that both the input and output are sequences of text: machine translation is framed as generating text in the target language given text in the source language, classification is framed as generating a class label and ranking can be cast as generating an ordering of candidates. This unifying principle allows the same architecture, loss function, and training pipeline to be applied across a wide variety of tasks.

T5 is built on the Transformer encoder–decoder architecture, in contrast to BERT, which is encoder-only. The encoder processes the input text, while the decoder autoregressively generates the output text, conditioned on both the encoder's representation and its own previously generated tokens. This architectural choice makes T5 naturally suitable for both understanding-oriented tasks (e.g., classification, ranking) and generation-oriented tasks (e.g., summarization, translation).

Whereas BERT is pretrained with a masked language modeling (MLM) objective, T5 uses a span corruption or denoising objective. Random spans of text in the input are replaced with sentinel tokens, and the model is trained to reconstruct the missing spans in order, generating them as output. This approach generalizes MLM while leveraging the encoder–decoder setup, encouraging the model to handle both missing-word prediction and free-form text generation.

The differences between T5 and BERT can be summarized as follows:

- **Architecture:** BERT is encoder-only, while T5 uses an encoder–decoder structure.
- **Objective:** BERT is pretrained with MLM, while T5 uses span corruption, a denoising objective that better aligns with generative tasks.
- **Task Formulation:** BERT typically requires task-specific output layers (e.g., classification heads), whereas T5 unifies all tasks into a text-to-text format, avoiding architecture modifications.
- **Generative Capabilities:** BERT is primarily designed for representation learning and classification, while T5 naturally supports both discriminative and generative tasks.

This unification and flexibility make T5 particularly attractive for information retrieval research, where reranking can be cast as a sequence-to-sequence task (e.g., generating labels, scores, or rankings). Building on this foundation, recent work has introduced retrieval-specific T5 variants which adapt the model to ranking scenarios more directly.

Nogueira et al. [33] were the first to adapt a pretrained sequence-to-sequence model, specifically T5, to the document reranking task. Their formulation differs from the classification-based approach commonly used with encoder-only models like BERT, which typically predict relevance by applying a classification head to the `[CLS]` representation. Instead, the T5 reranker casts ranking as a generation task.

In this setup, the query q and candidate document d are concatenated into an input sequence of the form:

`Query:` q `Document:` d `Relevant:`

The model is fine-tuned to generate either the token `true` or `false`, representing whether the document is relevant to the query. At inference time, the logits associated with these two tokens are normalized with a softmax, and the probability of the token `true` is used as the relevance score. Documents are then reranked according to these scores.

Finally, they conducted probing experiments by varying the target words (e.g., replacing “true/false” with unrelated or semantically inconsistent pairs). Results showed that the choice of target words significantly affected effectiveness, indicating that the model relies on latent semantic and linguistic knowledge learned during pretraining to map queries and documents to relevance judgments.

This work marks the first demonstration that pretrained sequence-to-sequence models can serve as effective and data-efficient rerankers, motivating the development of subsequent retrieval-specific T5 variants.

3.2.2 Advances in T5-based Reranking

Building on the initial monoT5 formulation [33], subsequent work has explored more direct and effective ways to adapt T5 for ranking. Two of the most notable developments are **RankT5** and **ListT5**, which extend the T5 framework to better align with learning-to-rank objectives and listwise inference.

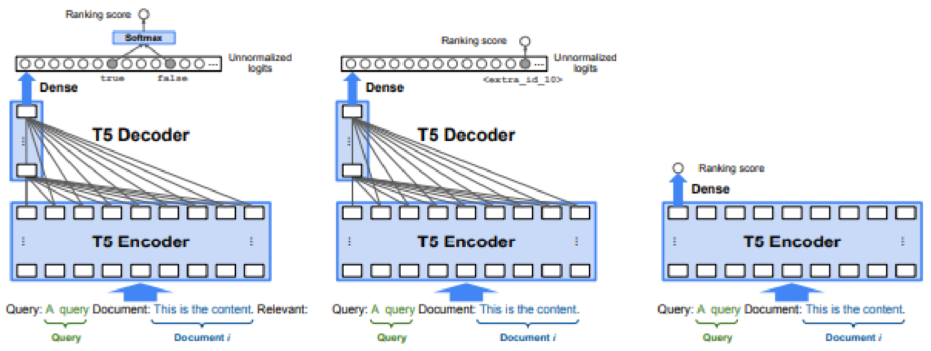
RankT5

The monoT5 model demonstrates that reranking can be formulated as a text generation task, where the model is trained to produce "true" or "false" tokens given a query-document pair. While effective, this formulation has two limitations: it reduces ranking to binary classification, and it does not directly optimize for ranking metrics that depend on the relative ordering of multiple documents.

To address these issues, Zhuang et al. [61] introduced RankT5, which adapts T5 to natively support ranking by predicting real-valued scores instead of tokens. RankT5 introduces two main variants (Figure 3.4):

- **Encoder-Decoder RankT5**: the standard T5 architecture is preserved, and the decoder outputs a reserved special token whose unnormalized logit is taken as the relevance score. This leverages the generative decoder but bypasses the need to interpret discrete tokens like "true/false".
- **Encoder-Only RankT5**: the T5 encoder is paired with a pooling layer followed by a dense layer to directly predict a scalar score. This variant discards the decoder entirely, making inference faster and closer in style to BERT rerankers.

A central contribution of RankT5 is its effort to align training objectives with evaluation metrics. While monoT5 is trained as a binary classifier, retrieval effectiveness is typically measured with ranking metrics such as MRR



(a) monoT5 (Nogueira et al., 2020) (b) RankT5 (Encoder-Decoder) (c) RankT5 (Encoder-Only)

Figure 3.4: Comparison between monoT5 [33] and RankT5 [61].

and NDCG, which are sensitive to the relative order of documents. To better capture this distinction, RankT5 explores three classes of training losses: pointwise, pairwise, and listwise. Pointwise cross-entropy treats each query–document pair independently and thus fails to optimize for relative order. Pairwise logistic loss improves alignment by encouraging relevant documents to be ranked higher than non-relevant ones, but it still neglects global list structure. In contrast, listwise objectives such as ListMLE and softmax cross-entropy directly model permutations of candidate documents, providing a closer match to ranking metrics like NDCG. Empirical results confirm that listwise RankT5 consistently outperforms pointwise and pairwise variants.

The encoder-only variant of RankT5 offered a compelling trade-off between effectiveness and efficiency, reducing inference cost while preserving competitive accuracy.

ListT5

Although RankT5 aligns training objectives with ranking metrics, it remains a pointwise approach at inference: documents are scored independently and later sorted. Yoon et al. [60] pushed this idea further by introducing ListT5, which performs listwise reranking by considering multiple candidate passages jointly.

As shown in Figure 3.5, each candidate passage is concatenated with the query and a unique index, and then encoded independently by the T5 encoder. The decoder then autoregressively generates a permutation of passage indices, which directly corresponds to a ranking of candidates.

This formulation has several advantages:

- It allows the model to directly output an ordering, eliminating the need for separate score sorting.
- It captures dependencies between passages during inference, enabling the model to reason jointly about relative relevance.

A challenge in listwise inference is computational scalability: naïve approaches can require quadratic $O(n^2)$ comparisons across documents. To overcome this, ListT5 employs an m-ary tournament sort algorithm with output caching, reducing inference to $O(n + k \log n)$ complexity. This makes the model practical even when reranking hundreds of candidates.

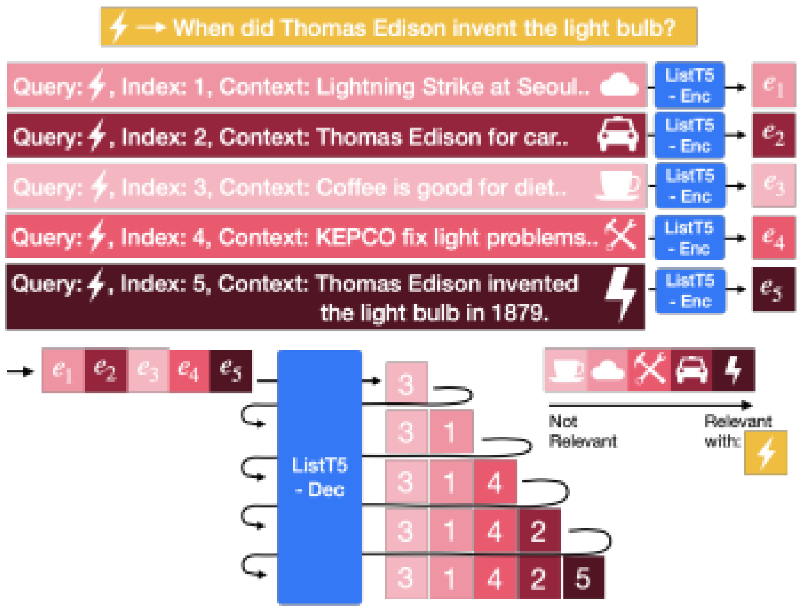

When did Thomas Edison invent the light bulb?
Query: , Index: 1, Context: Lightning Strike at Seoul..
Query: , Index: 2, Context: Thomas Edison for car..
Query: , Index: 3, Context: Coffee is good for diet..
Query: , Index: 4, Context: KEPCO fix light problems..
Query: , Index: 5, Context: Thomas Edison invented the light bulb in 1879.
ListT5 - Enc
e_1 e_2 e_3 e_4 e_5
ListT5 - Dec
3 1 4 2 5
Not Relevant
Relevant with:

Figure 3.5: Illustration of ListT5 [60].

In summary, RankT5 and ListT5 represent complementary advances in adapting T5 to reranking. RankT5 improves alignment between training objectives and ranking metrics, while ListT5 reformulates inference to output rankings directly. Together, they highlight the growing role of encoder–decoder architectures in state-of-the-art information retrieval.

3.3 Beyond BERT and T5: LLMs for Zero-Shot Reranking

The rapid progress of large language models (LLMs) such as GPT-4 has opened a new frontier in reranking research. Unlike earlier models such as BERT and T5, which typically rely on supervised fine-tuning with large amounts of labeled relevance data, LLMs bring a different advantage: strong zero-shot reasoning capabilities developed through large-scale pretraining and instruction tuning. This makes them especially appealing in scenarios where task-specific training data is scarce or entirely absent.

As a result, recent work has begun to explore how prompting strategies and structural adaptations can transform general-purpose LLMs into effective rerankers. Rather than requiring additional training, these approaches aim to exploit the models' existing reasoning abilities to directly order candidate documents by relevance.

RankGPT

One of the first systematic attempts to adapt LLMs for reranking is RankGPT, introduced by Sun et al. [51]. Their work investigates whether models such as ChatGPT and GPT-4 [36] can be instructed to perform ranking tasks effectively.

The LLM is prompted to output a complete permutation of the candidate set in descending order of relevance. This setup encourages the model to jointly reason over multiple candidates when determining their order.

To deal with the input length constraints of GPT-4, the authors employ a sliding window strategy: candidate passages are divided into overlapping groups, ranked within each window, and then merged into a final ordering. Experiments across standard benchmarks—including TREC DL [12],BEIR [52], and Mr.TyDi [44] show that GPT-4, when used in this way, consistently outperforms strong supervised baselines.

To further test generalization, Sun et al. introduced NovelEval, a small evaluation set collected after the release of GPT-4 to ensure that the model had not seen the relevant information during training. Even on this challenging test, GPT-4 achieved state of the art performance, suggesting that it can effectively rerank unfamiliar content in a zero-shot setting.

Finally, recognizing the computational expense of deploying GPT-4 in large-scale search systems, the authors explored permutation distillation. Here, the rankings generated by GPT-4 are used as supervision to train smaller student models (such as cross-encoders or compact GPT like architectures). These distilled models inherit much of GPT-4's performance while being far more efficient to deploy.

Taken together, the RankGPT results suggest that modern LLMs not only rival but often surpass specialized supervised rerankers in zero-shot conditions. Moreover, the use of distillation highlights a practical path toward deployment, where powerful but expensive LLMs can serve as teachers for lightweight and cost-effective retrieval models.

TourRank: Tournament-Inspired Zero-Shot Reranking

While RankGPT demonstrates that LLMs can serve as powerful zero-shot rerankers, it also highlights several practical challenges. Among these are the limited input length of current models, their sensitivity to the order in which candidates are presented, and the high computational cost of listwise or pairwise prompting. Addressing these issues requires more structured prompting strategies that can scale more effectively.

To this end, Chen et al. [10] introduced TourRank, a framework that reimagines reranking as a tournament. Inspired by sports competitions such as the FIFA World Cup, TourRank avoids asking the LLM to order a large candidate set all at once. Instead, documents are grouped into small "matches," with the LLM selecting the most relevant candidates to advance through successive rounds.

This tournament-style design offers several advantages:

- **Scalability:** By restricting comparisons to small groups, TourRank naturally sidesteps the input length constraints of LLMs, while also allowing parallel processing across matches.
- **Robustness:** Multiple independent tournaments are run, and their results are aggregated using a points based ensemble, reducing the impact of input ordering and model stochasticity.
- **Efficiency:** Compared with pairwise prompting (which scales quadratically) or listwise prompting (which requires long contexts), TourRank achieves a favorable balance between effectiveness and computational cost.

By reframing reranking as a structured tournament, TourRank turns the constraints of LLMs into design opportunities. It demonstrates that careful prompting strategies can bring both effectiveness and efficiency, making LLM based reranking more practical for real world retrieval systems.

DynRank: Dynamic Prompting with Question Classification

A complementary direction to listwise and tournament-style approaches is taken by Abdallah et al. [1], who introduced DynRank. Instead of applying a fixed prompt across all queries, as in RankGPT, DynRank adapts the instruction dynamically based on fine-grained question classification. In this way, the prompting strategy becomes sensitive to the specific intent of the query.

Formally, each candidate passage z_i is evaluated by estimating how likely the LLM can regenerate the original question q under a dynamic prompt p. The relevance score is given by:

$$s_i = \frac{1}{|q|} \sum_{t=1}^{|q|} \log P(q_t \mid q_{<t}, z_i, p; \Theta),$$

where $|q|$ is the number of tokens in the question, q_t is the t-th token, $q_{<t}$ denotes all preceding tokens, z_i is the candidate passage, p is the dynamic prompt, and Θ are the parameters of the pre-trained language model. Candidate passages are then reranked in descending order of s_i.

This formulation makes the reranking process more query aware, since the evaluation criterion adapts to the type of question. For instance, queries classified as HUM:individual emphasize biographical cues, while NUM:date queries highlight temporal information. By coupling classification with dynamic prompting, DynRank addresses a different limitation than TourRank: rather than restructuring how candidates are compared, it adapts how the model is instructed. This improves flexibility and generalization in zero-shot reranking.

3.4 ModernBERT for Reranking

Encoder-only Transformers have remained the workhorses of retrieval and classification since BERT, offering lower latency and simpler deployment than decoder style LLMs. Yet, despite numerous variants, much of the ecosystem still inherits core constraints from the original 2018 design: short effective context windows, absolute positional encodings, GeLU activations, and training/inference inefficiencies that become more pronounced at scale. These limitations motivate a more thorough modernization of the encoder architecture culminating in ModernBERT.

Over the years, successive BERT style models tackled isolated pain points without fully revising the backbone. RoBERTa [26] emphasized stronger pre-training; ALBERT [22] reduced parameters via sharing and factorized embeddings; DistilBERT [46] compressed models through distillation; ELECTRA [11] improved sample efficiency with replaced token detection. Other lines pushed longer inputs or better representations—Longformer [5] with sparse attention for extended contexts, and DeBERTa [19] with disentangled attention and relative positions.

Despite these advances, most variants retained architectural choices that now look dated: absolute position embeddings that do not scale gracefully to long sequences; post norm layouts and activation choices that limit training stability and expressivity; attention patterns that do not exploit hardware friendly kernels; and depth/width allocations not tuned for modern accelerators. As a result, context length, throughput, and cost have remained bottlenecks for high throughput reranking. ModernBERT addresses these issues as a cohesive, hardware-aware upgrade rather than a single point fix.

Architectural Upgrades

ModernBERT [57] revisits BERT's blueprint end to end, adopting contemporary design patterns and kernels that matter in practice:

- **Bias removal.** Bias terms are removed from linear layers and LayerNorms (except in the decoder head), reallocating capacity to more impactful components and simplifying kernels.

- **Rotary positions.** Rotary positional embeddings (RoPE) [50] replace learned absolute position embeddings by rotating query/key vectors in each attention head by a position dependent phase. This yields smoother length generalization than absolute embeddings, straightforward extension to long contexts and compatibility with fused attention kernels (e.g., FlashAttention).

- **Pre-normalization.** The model switches to pre-norm Transformer blocks for stability. An additional LayerNorm is applied after embeddings, and a redundant early attention LayerNorm is removed to streamline computation.
- **Structured attention.** Attention alternates between local sliding windows (e.g., 128 tokens) and periodic global layers (e.g., every third block), preserving accuracy while reducing quadratic overhead.
- **FlashAttention integration.** Local layers use FlashAttention-2 [14], while global layers adopt FlashAttention-3 [48], improving memory efficiency and kernel throughput on modern GPUs.

Together, these choices modernize the encoder stack, yielding a model that is both more scalable (longer contexts, better kernels) and more trainable (stable norms, stronger activations), without abandoning the simplicity and deployment advantages of encoder-only inference.

Pretraining and Efficiency

ModernBERT's gains also come from a training recipe tuned for utilization, stability, and long-context capacity:

- **Data mixture.** Broad pretraining over web, code, and scientific text with a modernized BPE tokenizer, deduplication and filtering improve coverage without overfitting to frequent sources.
- **Objective & signal density.** Masked language modeling with a higher masking rate (e.g. $\sim$30%) and dynamic masking increases per token learning signal at large batch sizes.
- **Token packing.** Pack multiple short sequences into each $L_{\max}$ window (bin-packing) and mask cross-sequence attention, converting padding

into useful tokens. Utilization $U = \frac{T_{\text{nonpad}}}{B \cdot L_{\max}}$ rises from $\ll 1$ to ≈ 1 on heterogeneous corpora. Correctness follows from per-token `attn_mask`, a loss mask that ignores padding/sentinels, and per-segment MLM corruption.

- **Long-context adaptation.** RoPE scaling with continued training extends context from 1,024 to up to 8,192 tokens, preserving short-context quality while enabling cross-document reasoning.

ModernBERT reframes the BERT style encoder for today's hardware and reranking workloads: rotary positions and structured attention unlock longer contexts, pre-norm and bias removal stabilize and streamline training, FlashAttention kernels and token packing drive near full utilization and continued training with RoPE scaling preserves quality while extending reach. In aggregate, these changes translate into higher throughput at lower latency for reranking, with the capacity to incorporate multi passage evidence.

Chapter 4

Dataset Creation and Analysis

A key contribution of this thesis is the construction of a new English passage reranking dataset specifically designed to overcome the limitations of existing benchmark collections. As discussed in Chapter 1, most publicly available datasets for reranking adhere to a rigid one positive, many negatives format, in which each query is associated with a single relevant passage and a set of explicitly non relevant distractors. While such datasets are effective for evaluation, they do not accurately reflect real world retrieval scenarios, where multiple passages may express different degrees of relevance to the same information need.

The goal of the dataset introduced in this work is therefore twofold:

1. To provide a more realistic supervision signal, capturing not only binary relevance but also graded semantic relatedness between passages.

2. To expose reranking models to richer intra query variability, encouraging them to reason beyond exact answer spans and instead learn contextual preference ordering among partially relevant candidates.

This chapter presents the dataset construction pipeline in detail. We begin by describing the source corpora and query selection strategy, followed by the methodology used to retrieve candidate passages. We then outline the annotation procedure employed to assign relevance labels, highlighting cases where multiple passages are considered partially or complementary relevant. Finally, we provide statistics on the resulting dataset, including query distribution, passage lengths and relevance label frequencies.

4.1 Overview of the Pipeline

The construction of the dataset follows a multi-stage pipeline designed to simulate realistic retrieval conditions while enabling fine-grained relevance supervision. The process begins with the selection of target Wikipedia pages, chosen from question-answering benchmarks such as SQuAD and Google QA. Each selected page is segmented into coherent passages, which serve as the initial pool of candidate documents.

To generate diverse information needs, synthetic queries are produced using a large language model for a subset of passages. In order to avoid overly narrow supervision restricted to a single source page, the candidate corpus is expanded by incorporating related Wikipedia pages obtained through category links, disambiguation pages, and internal references.

For each query, a retrieval step is performed to obtain the top-N most relevant passages from the expanded candidate set. These passages are then jointly evaluated by a language model, which assigns relevance labels that capture different degrees of semantic alignment with the query.

The following sections provide a detailed description of each stage of this pipeline.

4.2 Data Sources, Expansion, and Passage Embedding

As outlined in the pipeline overview, the first phase of dataset construction involves creating a large and diverse pool of candidate passages. This section details the three core steps of this process: selecting an initial set of articles from established benchmarks, expanding this collection to include topically related content, and embedding the resulting passages to enable efficient semantic retrieval.

4.2.1 Initial Article Selection

The construction process begins by extracting Wikipedia article titles from two widely used question answering benchmarks: SQuAD [40] and Google NQ [21]. These sources were selected for three main reasons. First, both datasets contain naturally phrased questions written by human annotators, ensuring that the underlying information needs are fluent and diverse. Second, each question is explicitly linked to its originating Wikipedia page, providing a stable contextual anchor. Third, the two datasets exhibit complementary content characteristics: while SQuAD mainly draws from expository articles, Google NQ includes longer and structurally varied pages. Combining both yields a broader and more heterogeneous set of initial pages.

While these benchmarks provide a valuable starting point, directly reusing their original passages proved infeasible due to the evolving nature of Wikipedia. In many cases, the content associated with a question had been restructured, relocated, or removed. To ensure consistency, we retain only the *page titles* from the benchmarks and re-download all pages in their most recent form. This forward-compatible strategy guarantees that supervision remains grounded in a temporally consistent snapshot of Wikipedia.

4.2.2 Corpus Expansion for Hard Negative Mining

After selecting the initial set of Wikipedia pages, each source article is expanded into a broader local corpus by following interlinking relations that capture semantic neighborhood structures. This expansion serves two main purposes: (i) to replicate the natural network of related content within Wikipedia, and (ii) to enable the retrieval of challenging distractor passages that are topically close yet ultimately non-relevant. For each initial page, we follow three categories of links—internal hyperlinks, category and disambiguation links, and “See also” references. To maintain topical breadth without uncontrolled expansion, we cap each neighborhood to 20 disambiguation pages, 30 category-linked pages, and 40 internal links.

This approach encourages the inclusion of *hard negatives*—passages that are semantically close to the query but not fully relevant. Such negatives provide stronger supervision for reranker training by forcing models to make fine-grained relevance distinctions [59]. Our construction intentionally favors settings where multiple partially relevant candidates coexist, aligning with the dataset’s primary goal.

4.2.3 Content Parsing and Segmentation

Each selected article and its expanded set of linked pages are downloaded and parsed. Boilerplate elements such as navigation menus, citation markers, and template artifacts are removed to obtain clean text. The resulting content is then segmented into overlapping passages of approximately 100 tokens using a sliding-window strategy. Metadata including section titles, source URLs, and outgoing links are retained for downstream provenance tracking. In total, 2,175 Wikipedia pages were processed, forming the base candidate pool for subsequent retrieval and reranking.

4.2.4 Passage Embedding for Retrieval

To enable semantic retrieval, each passage chunk is embedded into a dense vector representation. While traditional sparse approaches like BM25 rely on lexical overlap, dense embeddings from transformer-based encoders capture paraphrasing and implicit semantic relations. We evaluated a set of representative embedding models covering diverse architectures and objectives:

- **E5 (e5-small)** [56]: an instruction-tuned encoder optimized for retrieval tasks using contrastive learning with explicit role prefixes (e.g., `query:`, `passage:`).
- **MPNet (multi-qa-mpnet-base-dot-v1)** [41]: a cross-encoder distilled into a dual-encoder form, commonly used for question-answer retrieval.
- **Sentence-T5 (sentence-t5-base)** [31]: a sequence-to-sequence model repurposed for embedding via decoder pooling.
- **MiniLM (all-MiniLM-L6-v2)** [47]: a compact transformer model suited for resource-constrained retrieval.
- **BGE-M3 (BAAI/bge-m3)** [9]: a multilingual model supporting dense, sparse, and multi-vector retrieval.
- **BM25** [53]: a lexical baseline included for comparison.

For evaluation, a validation set of 100 Wikipedia pages was created. For each question sampled from SQuAD or Google NQ, the task was to retrieve the passage containing the correct answer from among all chunks derived from the same page. Retrieval was considered successful at rank-k if the target chunk appeared within the top-k results (Table 4.1).

Model	**Top-1**	**Top-3**	**Top-5**	**Top-10**
E5-small	76%	89%	95%	96%
MPNet	69%	87%	92%	98%
Sentence-T5	61%	83%	88%	93%
MiniLM	64%	83%	89%	95%
BGE-M3	81%	93%	95%	98%
BM25	56%	88%	92%	97%

Table 4.1: Retrieval success rates across different embedding models (single-page setting).

The choice between the two top-performing models, BGE-M3 and E5, was ultimately determined by scalability. BGE-M3's superior accuracy was offset by significant computational costs in GPU memory and throughput, making it impractical for this project's scale. E5 (e5-small) presented a more balanced profile, delivering competitive accuracy with high efficiency. This strong performance-to-cost ratio, combined with its helpful instruction-prefix format, made E5 the clear selection for the default encoder.

4.3 Query Generation, Retrieval, and Relevance Annotation

With the passage corpus established, the next phase of the pipeline focuses on creating supervised training instances. This involves a three-stage process: (1) generating synthetic queries grounded in the passage content, (2) retrieving a set of candidate passages for each query using a dense retriever, and (3) assigning fine-grained relevance labels to these candidates using an LLM annotator. This section details each of these stages in sequence.

4.3.1 Synthetic Query Generation

While the source benchmarks contain high-quality questions, they are bound to historical passages that no longer align with our reconstructed Wikipedia chunks. To create supervision for the updated corpus, we adopted a controlled large language model (LLM) pipeline to synthesize realistic, user-style queries directly from the current content.

After experimenting with several models, GPT-4o Mini [37] was chosen for its favorable balance of linguistic accuracy, cost-effectiveness, and format compliance, making large-scale generation practical. To ensure reliability, the model was instructed to return `"NA"` if a valid question could not be derived from the input text, thereby minimizing hallucinations.

Two distinct prompting strategies were employed to generate a diverse range of information needs.

- **Single-Chunk Queries:** For isolated passages, the model generated six factoid-style questions, one for each primary interrogative category (What, Where, When, Who, Why, How). This created supervision similar to traditional QA datasets.
- **Multi-Chunk Relational Queries:** To encourage more complex reasoning, multiple thematically related chunks were provided to the model. It was tasked with generating queries that required integrating information from *all* provided chunks to be answered. These queries targeted four relational categories: *Comparison*, *Cause-and-Effect*, *Temporal Sequence*, and *Elaboration*.

To create varied inputs for the multi-chunk strategy, passages were grouped using either sequential (adjacent chunks from the same section) or random (non-contiguous chunks) sampling patterns. The specific prompts used for single-chunk and multi-chunk generation are detailed in the listings below.

Single-Chunk Prompt

```
You are an expert question generator. Your task is to generate
six distinct questions
from the provided text, one for each of the fundamental
interrogative categories
(What, Where, When, Why, Who, How), and provide a direct answer
from the text.

**Primary Goal:**
For each category, create a single, insightful question and its
corresponding answer,
both derived *only* from the provided text passage.

**Mandatory Categories:**
```

```
You must generate a question-answer pair for each of the
following:
What, Where, When, Why, Who, How.

**Quality Standards:**
- Each question must be answerable using *only* the provided
text.
- NA Fallback Rule: If a question or answer for a specific
category cannot be formed
  from the text, you must use "NA" as the value.
- All questions must be self-contained and must not reference
"the text" or "the passage".

**Context from Wikipedia:**
---
{context}
---

Generate your response now. Prioritize creating high-quality,
answerable questions above anything else.
```

Multi-Chunk Relational Prompt

```
You are an expert query synthesizer for a search engine training
pipeline. Your task is
to analyze multiple related text chunks and generate a set of
insightful queries, one for
each of the four relational categories.

**Primary Goal:**
For each category below, create a query that a user might ask to
understand the specific
```

```
relationship between the chunks. Each query must be answerable
*only* by combining
information from ALL provided chunks.

**Mandatory Categories:**
You must generate a query-reasoning pair for each of the
following:
- Comparison: The chunks describe different entities that can be
compared or contrasted.
- Cause-and-Effect: One chunk describes an event or cause, and
another describes its effect or consequence.
- Temporal-Sequence: The chunks describe events that occur in a
sequence.
- Elaboration: One chunk introduces a topic, and the others
provide specific details or examples.

**Quality Standards:**
- NA Fallback Rule: If a meaningful query for a specific
category cannot be formed,
  you must use "NA" for both query and reasoning.
- Reasoning: Briefly explain *how* the chunks connect to answer
the query.

**Context from Wikipedia:**
--- Chunk 1 ---
{chunk 1}

--- Chunk 2 ---
{chunk 2}

(Additional chunks if available)
```

```
Your output must be a single JSON object containing all four
categories.
```

4.3.2 Candidate Retrieval via Dense Embedding Search

With the synthetic queries generated, the next step was to retrieve a set of candidate passages for each one, simulating the first stage of a search pipeline. For each query q, its embedding was computed using the E5 model (prefixed with ``query:'') and a dense similarity search was performed against all chunk embeddings from the corresponding source page and its expanded neighborhood:

$$\text{score}(q, p_i) = \cos(\text{E5}(q), \text{E5}(p_i)).$$

The top-$N = 30$ highest-scoring passages were selected as the initial candidate set for annotation.

Two filtering steps were applied to ensure the quality of these candidate sets. First, a ground truth validation step confirmed that all of the original passages used to generate the query were present within the top-30 retrieved candidates. Queries failing this check were discarded. Second, a semantic deduplication process was applied to remove near-duplicate passages. If two candidates exceeded a cosine similarity of 0.90, only the higher-scoring one was retained, unless one was a ground-truth positive. This ensured that each query was paired with genuinely competing candidates rather than trivially repetitive ones.

4.3.3 LLM-Based Relevance Annotation

While dense retrieval provides a ranked list, training a reranker requires explicit, graded relevance scores. To obtain this supervision at scale, we employed GPT-4o Mini as an automatic annotator. For each query, the full set of up to 30 retrieved candidate passages was presented to the LLM for scoring. To minimize positional bias, the passages were randomly shuffled before being included in the prompt.

The model was instructed to assign each passage a relevance score from 1 (No Relevance) to 10 (Complete Direct Answer), along with a brief justification, based on a detailed scoring rubric. The prompt used for this task is shown below.

```
You are a highly intelligent and precise relevance scoring AI.

TASK:
Your task is to evaluate how relevant each of the following N
text
chunks is to the given query. For each chunk, assign a score
from 1 to 10
and provide a brief reasoning.

SCORING CRITERIA (1-10):
10 - Complete Direct Answer: Fully and precisely answers the
entire query.
9  - Nearly Complete: Strong answer missing only minor details.
8  - Strong Partial Answer: Directly addresses the core question
but incomplete.
7  - Clearly Relevant: Provides useful information that relates
to the query.
6  - Supportive Information: Background or secondary context
that helps indirectly.
5  - Topic Match: Discusses the same subject but does not answer
the question.
4  - Tangentially Related: Mentions related concepts but focuses
elsewhere.
3  - Weak Connection: Contains loose keyword or thematic
overlap.
```

```
2  - Minimal Relevance: Very distant relationship to the
question.
1  - No Relevance: Completely unrelated to the query.

IMPORTANT:
- Use the full range of scores when appropriate.
- Do NOT omit any chunk. Every chunk must receive exactly one
score.

QUERY:
{query}

CHUNKS TO SCORE:
Chunk 1:
{text_1}
---
...
```

Although the LLM annotated all 30 passages, we employed a structured subsampling strategy to construct the final training instances, capping the number of candidates per query at 10. This was a deliberate design choice to improve the quality of the learning signal. The final set for each query was constructed by:

1. Preserving all known positive passages (those used to generate the query).
2. Selecting the top-6 highest-scoring non-positive passages to retain strong hard negatives.
3. Randomly sampling from the remainder to reach the cap of 10.

This approach creates a compact yet diverse supervision signal that balances clear positives against competitive distractors, encouraging the model to learn fine-grained distinctions without being overwhelmed by low-value negatives.

4.4 Dataset Statistics

The final dataset contains 10,000 annotated query-candidate groups. Each data point consists of a synthetic user query paired with exactly 10 passages, each assigned a relevance score from 1 to 10 by the LLM annotation procedure. The following subsections provide a detailed breakdown of the dataset's composition, analyzing the distribution of query types, their linguistic forms, and the resulting relevance scores.

4.4.1 Query Type Distribution

The dataset contains two primary query categories: single-chunk factual questions and multi-chunk relational questions. As shown in Table 4.2, the majority of queries are factoid-style, providing a strong foundation for standard retrieval tasks, while a significant subset of relational queries is included to facilitate research on cross-passage reasoning.

Table 4.2: Distribution of query types in the dataset.

Query Type	**Count**	**Percentage**
Single-Chunk Factual	7,723	77.23%
Multi-Chunk Relational	2,277	22.77%

4.4.2 Analysis of Query Phrasing

A finer-grained analysis of the initial query tokens reveals broad coverage of common interrogative forms (Table 4.3). The near-uniform spread across ”What,” ”How,” ”When,” ”Where,” ”Why,” and ”Who” indicates that the dataset reflects a diverse range of user intent types, moving beyond simple fact-finding to include procedural and explanatory information needs.

Table 4.3: Distribution of initial query tokens.

First Word	Count	Percentage
What	2,461	24.61%
How	2,405	24.05%
When	1,306	13.06%
Where	1,272	12.72%
Why	1,274	12.74%
Who	1,276	12.76%
Other	6	0.06%

4.4.3 Relevance Score Distribution

Each of the 100,000 passages in the dataset (10 passages for each of the 10,000 queries) was scored on a 1–10 relevance scale. The overall distribution of these scores is presented in Table 4.4.

Table 4.4: Overall distribution of relevance scores across all passages.

Score	Frequency	Percentage
1	15,544	15.54%
2	9,750	9.75%
3	8,234	8.23%
4	11,533	11.53%
5	11,029	11.03%
6	11,473	11.47%
7	6,257	6.26%
8	7,847	7.85%
9	8,011	8.01%
10	10,321	10.32%

The distribution spans the full range, with substantial representation in both the low-relevance (scores 1–3) and high-relevance (scores 8–10) regions. This diversity is crucial for training robust reranking models, as it provides the necessary signal to distinguish not only between relevant and irrelevant passages but also between partially and fully correct answers. The prevalence of mid-range scores (4–7) further supports the dataset's goal of modeling graded relevance.

4.5 Integration of the TWOLAR Dataset

In addition to the dataset constructed in this work, a complementary portion of training data was incorporated from the TWOLAR collection [4]. The goal of this integration was to further diversify the supervision signal by including queries and passages derived from independently designed retrieval pipelines. This section summarizes the TWOLAR dataset construction methodology and details how a curated subset of it was adapted for compatibility with our reranking framework.

4.5.1 Overview of the TWOLAR Dataset

The TWOLAR dataset was introduced by Baldelli et al. [4] as part of a two-step LLM-augmented distillation framework for passage reranking. The approach leverages large language models (LLMs) as high-quality teachers to automatically produce labeled query–document pairs without requiring human annotations. The resulting collection was designed to capture the fine-grained relevance judgments of an LLM while maintaining diversity across different retrieval paradigms.

The TWOLAR pipeline begins by generating two parallel sets of 10,000 synthetic queries from the MS MARCO corpus [3]:

- **Cropped-sentence queries**, obtained by sampling sentences directly from MS MARCO passages to form concise pseudo-queries.
- **docT5query-generated queries**, produced using the `docT5query` model [34], which reformulates passages into natural-language questions.

Each query subset (cropped or generated) was divided into four groups of queries, corresponding to four independent first-stage retrieval systems: BM25, SPLADE [16], DRAGON [25], and BM25+monoT5. For each query–retriever pair, the top 30 passages were collected, resulting in a large pool of candidate documents exhibiting substantial lexical and semantic diversity.

In the second stage, these retrieved passages were reranked using an LLM-based teacher model (gpt-3.5-turbo-16k-0613). Following the listwise prompting design of RankGPT, the model was provided with all 30 documents per query and asked to output a relevance-ordered permutation of the candidates. This LLM-driven reranking step effectively distilled ChatGPT's implicit ranking knowledge into labeled data suitable for fine-tuning compact rerankers such as `Flan-T5`. In total, the TWOLAR dataset contains 20,000 annotated query–document groups, each consisting of 30 candidate passages and an associated rank order derived from the LLM annotations.

4.5.2 Sampling and Adaptation for the Present Work

From the publicly released TWOLAR dataset,[1] we sampled a subset of 10,000 query–passage pairs to complement our in-house collection. To ensure both retrieval diversity and topical variety, we focused on examples originating from the DRAGON, SPLADE, and BM25+T5 retrieval sources. These systems collectively represent dense, sparse, and hybrid retrieval paradigms that closely align with the models analyzed in this thesis.

Unlike the original TWOLAR annotations, which provide only an implicit ordinal ranking of the top 30 passages per query, we employed an explicit grading procedure to obtain fine-grained relevance scores. Specifically, we used the `gpt-4o-mini` model to assign each passage a relevance grade on a 1–10 scale, following the guidelines described in Section 4.3.3. This approach produces semantically calibrated relevance scores that are directly comparable to those used in our internal datasets.

To maintain structural consistency across all training instances, we then subsampled each TWOLAR query group to retain only the top 10 passages according to the GPT-based relevance scores. This step mirrors the candidate size used for our synthetic queries and ensures a uniform query–candidate format suitable for joint training and evaluation across all data sources.

[1] Available at `https://huggingface.co/datasets/Dundalia/TWOLAR_ds`.

4.5.3 Resulting Combined Dataset

After integration, the expanded dataset comprises a total of 20,000 annotated query–candidate groups, evenly split between the newly constructed collection and the adapted TWOLAR subset. The TWOLAR-derived samples contribute a complementary dimension to the data:

- They introduce natural linguistic diversity through docT5query-style and cropped pseudo-queries.
- They provide challenging negatives retrieved via distinct systems (DRAGON, SPLADE, and BM25+T5), enhancing robustness against retriever-specific bias.
- They include relevance annotations derived from a high-capacity LLM teacher (`gpt-4o-mini`), yielding nuanced, fine-grained supervision signals on a 1–10 scale.

By combining local supervision with LLM-based relevance annotations, this hybrid dataset forms a rich and balanced training resource that reflects real-world semantic diversity and supports learning rerankers sensitive to nuanced relevance gradations.

Chapter 5

Fine-tuning Strategies and Experimental Methodology for Passage Reranking

This chapter presents the fine-tuning strategies and experimental methodology adopted to train ModernBERT for supervised passage reranking. Building on the graded relevance corpus introduced in Chapter 4, the central goal of this study is to investigate whether cross-encoder–based relevance modeling remains advantageous when passage relevance is expressed on a fine-grained, multi-level scale rather than through binary or coarse labels.

To address this question, two complementary encoding strategies are explored. The first approach, Independent Passage Encoding, follows the classical cross-encoder formulation in which each query–passage pair is processed independently, and relevance scores are optimized using listwise or pairwise learning-to-rank objectives. The second approach, All-Passage Encoding, extends the model to jointly encode all candidate passages associated with a query, enabling direct cross-passage interactions and potentially richer contextual signals.

Both strategies are fine-tuned under a unified experimental framework that includes consistent preprocessing, loss formulations, optimization routines, and evaluation metrics. By comparing their behavior across a range of ranking and correlation measures, this chapter aims to quantify the extent to which cross-attention–based modeling benefits passage reranking when trained on graded relevance supervision.

The remainder of the chapter details the two fine-tuning approaches, the experimental setup, and the results obtained from empirical evaluation.

5.1 Independent Passage Encoding

The first fine-tuning strategy adopts a classical cross-encoder formulation in which each query–passage pair is encoded independently. For a given query, the model processes its ten associated passages separately, producing a scalar relevance score for each candidate. The final ranking is obtained by jointly optimizing these scores under a listwise learning-to-rank objective. This approach serves as a strong and widely used baseline, enabling the model to capture fine-grained semantic interactions between the query and each passage through full cross-attention.

Sequence Construction. Each training instance consists of a single query paired with its ten candidate passages. The two texts are concatenated into the standard cross-encoder format:

$$\text{[CLS] Query [SEP] Passage [SEP]},$$

and tokenized using the ModernBERT tokenizer. Only the representation of the `[CLS]` token is used for relevance prediction, obtained through a linear layer applied to the encoder output.

Before defining the preprocessing pipeline, the distribution of actual token lengths for all query–passage pairs in the corpus was inspected. As shown in Figure 5.1, more than 99.8% of pairs fall below 256 tokens, with only 192 out of approximately 160,000 pairs exceeding this threshold. This allowed the use of a maximum sequence length of 256 tokens without incurring systematic truncation. Pairs exceeding the limit were discarded, representing only 0.12% of the data.

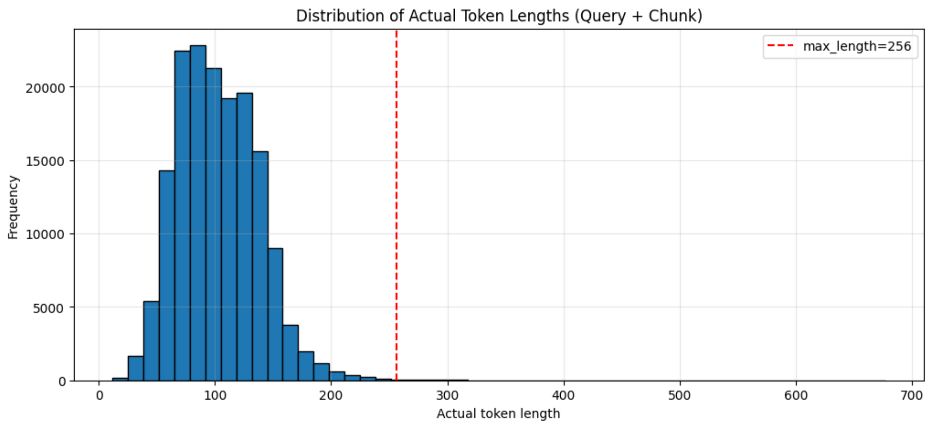


Figure 5.1: Distribution of query–passage token lengths

To ensure that each training instance provides a meaningful ranking signal, several consistency checks were applied. Instances containing incomplete or invalid labels (e.g., missing relevance scores) were removed, as were samples in which all passages shared the same relevance value. These uniform-label cases do not contribute to ranking learning and accounted for 23 removals from the training set, 5 from validation, and 6 from the test set. After filtering by label quality and token length, the final dataset contained:

$$\text{Train: } 15{,}826, \quad \text{Validation: } 1{,}974, \quad \text{Test: } 1{,}973.$$

For each query, the graded relevance annotations were scaled to the interval $[0, 1]$ using min–max normalization while preserving their relative ordering. Tokenized query–passage pairs were padded to the maximum length of 256 tokens, and batching was implemented at the level of entire query groups, yielding tensors of shape $(B, 10, L)$ for input IDs and attention masks.

All passages belonging to the same query are processed in parallel. For a batch of size B, the $B \times 10$ pairs are flattened along the passage dimension, encoded by ModernBERT, and reshaped back into a matrix of predicted relevance scores of size $(B, 10)$.

5.1.1 Evaluation Metrics

To assess reranking performance under the independent encoding strategy, a broad set of ranking metrics was employed. These metrics quantify both the quality of the highest-ranked passages and the overall agreement between predicted and true relevance orderings. Although several measures were considered, the primary evaluation criterion selected for all experiments is the **Exponential NDCG@3**. Its choice is motivated by the graded nature of the dataset and by the distributional properties of the relevance labels.

Standard NDCG

Normalized Discounted Cumulative Gain (NDCG) measures ranking quality by comparing the predicted ordering to an ideal ranking of the same items. It is computed as

$$\mathrm{NDCG}@k = \frac{\mathrm{DCG}@k}{\mathrm{IDCG}@k},$$

where the Discounted Cumulative Gain (DCG) at cutoff k is defined as

$$\mathrm{DCG}@k = \sum_{i=1}^{k} \frac{r_i}{\log_2(i+1)},$$

and r_i is the true relevance of the passage ranked at position i in the model's predicted ordering. The ideal DCG (IDCG) is computed by sorting passages by decreasing relevance and applying the same formula.

In principle, NDCG provides a normalized measure of ranking quality between 0 and 1. However, in datasets where multiple passages per query have high relevance scores and the relevance scale spans a wide numeric range, the resulting IDCG values can be large. Under such conditions, even weak or random scoring functions produce DCG values not far from IDCG, causing standard NDCG to fall within a narrow, high-valued band. This reduces its discriminative power for model comparison.

This effect is clearly observed in Table 5.1, which reports several NDCG variants computed for an untrained model and a random baseline. The standard NDCG metrics show almost no difference between the two conditions, indicating that they are not sufficiently sensitive for this task.

Table 5.1: Comparison of NDCG variants on untrained and random scoring functions.

Metric	**Untrained**	**Random**	**Gap**
Standard NDCG@10	0.8378	0.8409	-0.0031
Standard NDCG@5	0.6703	0.6748	-0.0045
Standard NDCG@3	0.6095	0.6208	-0.0113
Exponential NDCG@10	0.6025	0.6031	-0.0006
Exponential NDCG@5	0.4030	0.4009	0.0020
Exponential NDCG@3	0.3076	0.3162	-0.0086

Exponential NDCG

To increase the metric's sensitivity to ranking differences among highly relevant items, an exponential gain transformation was considered:

$$G(i) = 2^{r_i} - 1.$$

This formulation amplifies differences between high relevance values and enlarges the penalty for placing top-ranked passages in lower positions. When applied at a small cutoff (e.g., $k = 3$), the metric becomes particularly responsive to the correct ordering of the most informative passages.

This property aligns well with the structure of the dataset. Many queries contain multiple passages with substantive relevance scores; for example, 74.9% of training queries include at least three passages with $r \geq 7$. As a result, top-three ranking performance is especially meaningful, and Exponential NDCG@3 provides a clearer and more discriminative signal than its standard counterpart.

Additional Metrics

To provide a more complete evaluation, several complementary metrics were computed:

- **Exponential NDCG@5 and NDCG@10**, assessing ranking quality over longer lists.
- **MRR**, measuring the rank of the most relevant passage.
- **Top-k accuracy and recall**, evaluating whether the most relevant passages appear among the top predictions.
- **Spearman's** ρ and **Kendall's** τ, capturing global agreement between predicted and true orderings.

Together, these measures provide a detailed characterization of both local (top-k) and global ranking behavior. They are used consistently throughout the experiments on the independent encoding approach and later form the basis for comparison with the all-passage encoding strategy.

5.1.2 Learning-to-Rank Objectives

The independently encoded query–passage pairs are optimized using supervised learning-to-rank objectives that compare the predicted scores with the graded relevance annotations. Three losses were explored in this work: ListNet, LambdaRank, and an approximate variant of NDCG. These losses capture different aspects of the ranking problem and provide complementary approaches to modeling graded relevance.

ListNet

ListNet is a listwise learning-to-rank loss based on comparing probability distributions over candidate passages. Given predicted scores $\mathrm{s} \in \mathbb{R}^n$ and ground-truth relevance labels $\mathbf{r} \in \mathbb{R}^n$, both vectors are transformed into probability distributions using a temperature-scaled softmax:

$$P_{\text{pred}}(i) = \frac{\exp(s_i/\tau)}{\sum_{j=1}^{n} \exp(s_j/\tau)}, \qquad P_{\text{true}}(i) = \frac{\exp(r_i/\tau)}{\sum_{j=1}^{n} \exp(r_j/\tau)}.$$

The loss is the cross-entropy between the two distributions:

$$\mathcal{L}_{\text{ListNet}} = -\sum_{i=1}^{n} P_{\text{true}}(i) \log \left(P_{\text{pred}}(i) + \epsilon\right),$$

where ϵ ensures numerical stability. This loss encourages the entire predicted ranking distribution to match the distribution implied by the relevance labels, making it well suited for graded supervision.

LambdaRank

LambdaRank is a pairwise loss that optimizes the relative ordering of passage pairs. For each pair (i, j) with unequal relevance labels, the model is penalized when it predicts an incorrect ordering. Let s_i and s_j denote the predicted scores, and let r_i and r_j denote the corresponding true relevance values. A pair contributes to the loss only if $r_i > r_j$. The loss for a single query is

$$\mathcal{L}_{\text{LambdaRank}} = \frac{1}{M} \sum_{i<j} \log\left(1 + \exp\left[-\sigma(s_i - s_j)\right]\right) \mathbb{I}[r_i > r_j],$$

where M is the number of valid pairs and σ controls the steepness of the sigmoid. Unlike ListNet, LambdaRank directly models the preference relationships, making it effective in scenarios where relative ordering is the most important aspect of the ranking task.

Approximate NDCG

The third objective is a differentiable approximation of NDCG, designed to directly target the evaluation metric. Exact NDCG is non-differentiable because it depends on discrete ranks. The approximation replaces hard ranking with soft, expectation-based rank estimates. Let the gain of passage i be $G(i) = 2^{r_i} - 1$. For predicted scores $\mathbf{s}$, soft pairwise comparisons are computed using

$$P_{ij} = \sigma\left(\frac{s_i - s_j}{\tau}\right),$$

which approximates the probability that passage i should be ranked above passage j. The expected rank of each passage is then

$$\hat{\text{rank}}(i) = 1 + \sum_{j \neq i} (1 - P_{ij}).$$

An approximate DCG is computed by substituting the expected ranks into the DCG formula:

$$\text{DCG}_{\text{approx}} = \sum_{i=1}^{n} \frac{G(i)}{\log_2\left(\hat{\text{rank}}(i) + 1\right)}.$$

The loss is

$$\mathcal{L}_{\text{ApproxNDCG}} = 1 - \frac{\text{DCG}_{\text{approx}}}{\text{IDCG}},$$

where IDCG is computed from the ideal ranking based on the true labels.

Approximate NDCG provides a direct and differentiable surrogate for the evaluation metric used in this work. While computationally more expensive than ListNet or LambdaRank, it aligns the training objective closely with the final ranking metric and is therefore particularly suitable for graded relevance settings.

5.1.3 Training and Evaluation Procedure

The independent-encoding model is trained using mini-batch stochastic gradient descent with AdamW optimization, gradient accumulation, warmup-based learning rate scheduling, and early stopping. All experiments were conducted using ModernBERT-base, and all hyperparameters were kept constant across runs unless specified otherwise.

Training Loop

Model parameters are updated using the AdamW optimizer with learning rate 2×10^{-5} and weight decay implicitly handled through the optimizer's decoupled regularization. Due to memory constraints and the cost of encoding multiple passages per query, gradient accumulation is used to reach an effective batch size of

$$B_{\text{eff}} = B \times K,$$

where $B = 8$ is the per-step batch size and $K = 4$ the number of accumulation steps. Gradients are clipped to a maximum norm of 1.0 before each parameter update. Training proceeds for at most six epochs, with early stopping triggered if validation Exponential NDCG@3 does not improve by at least 0.01 for two consecutive epochs.

Learning Rate Scheduling

Preliminary experiments using a fixed learning rate resulted in poor convergence: the model failed to learn stable ranking behavior and validation metrics remained close to random. Introducing a cosine decay schedule with warmup proved essential for stable optimization. The learning rate follows

$$\eta(t) = \eta_0 \cdot \frac{1}{2}\left(1 + \cos\left(\pi \cdot \frac{t - t_{\text{warm}}}{T - t_{\text{warm}}}\right)\right),$$

after a linear warmup over 10% of the total training steps. Empirically, the cosine schedule produced smoother loss curves, more stable gradients, and substantially better ranking performance across all metrics. All results reported for this approach use the cosine scheduler.

Evaluation Protocol

Model evaluation is performed at the end of each epoch over the full validation set. For each query, the predicted scores for its ten passages are compared with their corresponding raw relevance labels. A comprehensive suite of ranking metrics is computed:

- **Exponential NDCG@1, @2, @3, @5**,
- **MRR**, **Top-1 accuracy**, and **Top-3 recall**,
- **Spearman's** ρ and **Kendall's** τ (rank correlations)

For each metric, values are averaged across all queries in the dataset. The validation Exponential NDCG@3 serves as the early stopping criterion and model selection metric.

Final Evaluation and Reproducibility

Several configurations of learning rates, loss functions, and scheduling parameters were explored to identify the most effective setup. Although all three ranking objectives (ListNet, LambdaRank, and ApproxNDCG) were evaluated under identical training conditions, their impact on performance proved to be minor. Across metrics such as Exponential NDCG@3 and MRR, the difference between the strongest and weakest loss amounted to less than one point, indicating that the model's behavior is largely insensitive to the specific choice of learning-to-rank objective. LambdaRank was therefore selected for the final experiments due to its slightly more consistent performance and stable convergence.

The chosen configuration was then retrained and evaluated using two additional random seeds to ensure that its performance was not an artifact of a favorable initialization. In total, the final evaluation is based on three seeds, all using the same hyperparameter settings and training procedure. For each run, the checkpoint achieving the highest validation Exponential NDCG@3 was selected and evaluated on the held-out test set. The following table summarizes the hyperparameters used in all three experiments.

Table 5.2: Final hyperparameter configuration used for the independent-encoding experiments.

Component	**Value**
Base model	`ModernBERT-base`
Loss function	LambdaRank ($\sigma = 1$)
Learning rate	2×10^{-5}
Optimizer	AdamW
Batch size	8
Gradient accumulation steps	4 (effective batch size 32)
Max sequence length	256 tokens
Scheduler	Cosine decay with warmup
Number of cosine cycles	0.5
Warmup proportion	10% of total steps
Epochs	8
Early stopping patience	2 epochs
Hardware	NVIDIA A100 (40GB, Google Colab)

The next section reports the test-set performance obtained from the three independent training runs, along with the corresponding seed-wise variability.

Test-Set Performance and Comparison with an Untrained Model

The selected hyperparameter configuration was evaluated using three random seeds (42, 123, 357). Seed 42, which achieved the highest validation Exponential NDCG@3, also produced the strongest overall performance on the test set, with its best checkpoint occurring at epoch 2. For seeds 123 and 357, the best checkpoints were obtained at epoch 1. Despite these differences, the overall variation across seeds is modest: all three runs exhibit closely aligned rankings, with differences typically within one to two points across the main metrics. This indicates that the selected training setup is stable and not strongly dependent on initialization.

Table 5.3 reports the complete set of test metrics for the three seeds.

To contextualize these results, the same evaluation procedure was applied

Table 5.3: Test-set results for the three seeds using the selected hyperparameter configuration.

Metric	**Seed 42**	**Seed 123**	**Seed 357**
MRR	0.6935	0.6806	0.6848
Exponential NDCG@3	0.7981	0.7829	0.7814
Exponential NDCG@5	0.8369	0.8231	0.8255
Top-1 Accuracy	0.5418	0.5291	0.5352
Top-3 Recall	0.6491	0.6307	0.6298
Pairwise Accuracy	0.7785	0.7666	0.7675
Spearman's ρ	0.6437	0.6179	0.6214
Kendall's τ	0.5276	0.5048	0.5064
Best Epoch	2	1	1

to an untrained ModernBERT cross-encoder, using random weights. As expected, the uninitialized model performs poorly on all ranking metrics that emphasize the top of the ranking (MRR, Top-1 Accuracy, Exponential NDCG), while achieving an artificially high score on Standard NDCG@10. This behavior confirms the earlier observation that Standard NDCG@10 is insufficiently discriminative for this dataset, as it produces values close to those of fully trained models even in the absence of meaningful ranking signals.

The comparison between the untrained model and the best-performing trained run (Seed 42) is shown in Table 5.4.

Table 5.4: Performance of the untrained model compared with the best trained model (Seed 42).

Metric	**Untrained Model**	**Trained Model (Seed 42)**
Standard NDCG@10	0.8456	0.8773
Exponential NDCG@3	0.3076	0.7981
Exponential NDCG@5	0.4030	0.8369
MRR	0.1824	0.6935
Top-1 Accuracy	0.1042	0.5418
Top-3 Recall	0.2116	0.6491
Spearman's ρ	0.0923	0.6437
Kendall's τ	0.0711	0.5276

Overall, the trained model yields substantial improvements across all meaningful metrics, especially those emphasizing correct identification of the most relevant passages. The large gap between the untrained and trained models demonstrates the effectiveness of the cross-encoder architecture and validates the fine-tuning strategy adopted in this work.

5.2 Joint Passage Encoding (All-Passage Model)

The second fine-tuning strategy is inspired by the architecture of JinaReranker-v3 [55]. Instead of encoding each query–passage pair independently, all passages are concatenated together with the query into one long sequence enriched with special marker tokens. The transformer thus has full visibility over cross-document context, potentially enabling more nuanced relevance estimation.

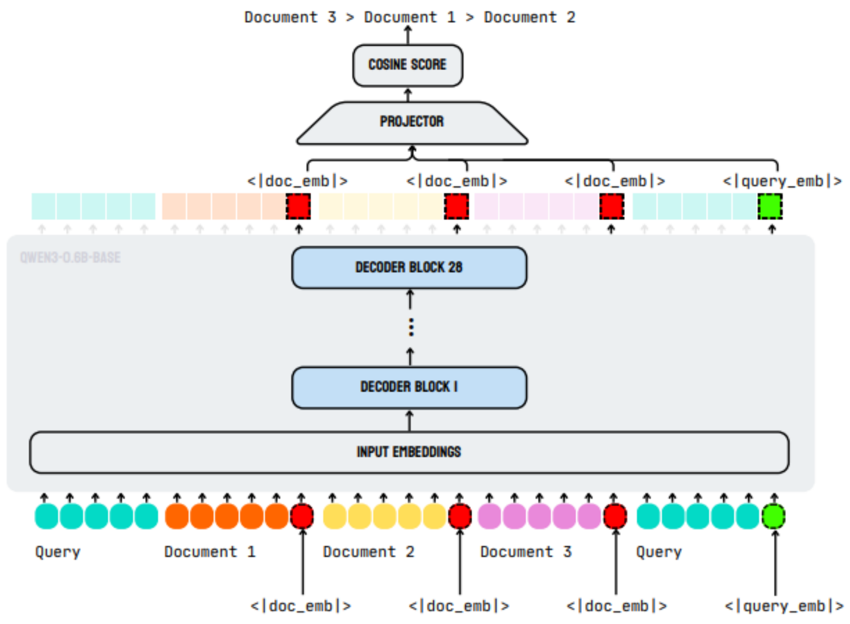


Figure 5.2: Architecture of the JinaReranker-v3 model (adapted from [55]).

In this formulation, a single sequence embeds:

$$\text{Query} + \langle\texttt{query\_emb}\rangle + \text{Document}_1 + \langle\texttt{doc\_emb}\rangle + \cdots + \text{Document}_{10} + \langle\texttt{doc\_emb}\rangle.$$

During the forward pass, the transformer processes the entire sequence holistically. The hidden states corresponding to each `<doc_emb>` token serve as document-level representations, while `<query_emb>` provides a comparable query embedding. These vectors are projected through a two-layer MLP and scored using cosine similarity.

This design offers two advantages over the independent-encoding baseline: (1) cross-document interactions become possible, as the model sees all passages at once, and (2) only a single transformer pass is required per query, making the approach more computationally efficient at inference time.

Figure 5.3 shows the distribution of total sequence lengths after concatenating all passages. The average length is approximately 890 tokens, well within the token limit of ModernBERT. No truncation was required for any query, unlike the independent-encoding strategy.

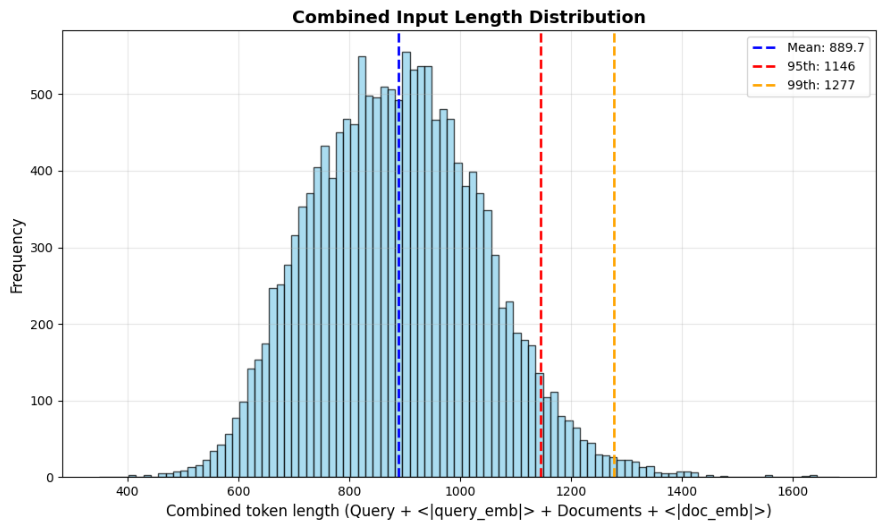


Figure 5.3: Distribution of combined token lengths (query + all passages + marker tokens).

Training Procedure and Evaluation

To ensure a fair comparison with the independent-encoding baseline, the all-passage model was trained using the same hyperparameter configuration that yielded the best results in the first strategy (learning rate, batch size, cosine scheduler with warmup, gradient accumulation, and early-stopping criteria). Only the architectural components differ, namely the joint query–document input format and the use of special marker tokens `<|query_emb|>` and `<|doc_emb|>` to extract embedding representations.

Two training variants were explored:

1. **Standard Joint Encoding:** Passages are encoded in their canonical order, as they appear in the dataset.

2. **Shuffled-Passage Variant:** For each epoch, the ten passages associated with each query are randomly permuted before concatenation. All relevance labels and document-embedding marker positions are shuffled accordingly. The goal of this variant is to assess whether the model

relies on positional biases and whether randomizing document order improves generalization.

Table 5.5 reports the full set of test metrics for the two all-passage variants, alongside the best-performing model from the first approach (Seed 42).

Table 5.5: Test-set comparison between the best independent-encoding model (Seed 42) and the two variants of the all-passage joint encoder.

Metric	**First Approach (Seed 42)**	**All-Passage Standard**	**All-Passage Shuffled**
Exponential NDCG@3	0.7981	0.7278	0.7158
Exponential NDCG@5	0.8369	0.7803	0.7671
MRR	0.6935	0.6527	0.6391
Top-1 Accuracy	0.5418	0.5095	0.5021
Top-3 Recall	0.6491	0.5695	0.5402
Spearman's ρ	0.6437	0.5496	0.5020
Kendall's τ	0.5276	0.4422	0.4016

Across all evaluation metrics, the independent-encoding approach remains the strongest performer. The all-passage joint encoder—despite its architectural advantages, full-context integration, and the ability to model interactions between passages—does not surpass the simpler first strategy. The gap is consistent and sizeable, with decreases of approximately 6–10 percentage points in the primary ranking metrics (MRR, Exponential NDCG@3/5).

Between the two joint-encoding variants, the shuffled-passage model yields slightly lower scores than the standard ordering. This suggests that the model does not rely heavily on positional biases; however, randomizing the passage order does not provide a regularization benefit and may mildly disrupt the model's ability to learn stable passage–query relationships.

These findings indicate that the independent-encoding configuration is better aligned with the structure of the dataset. Encoding passages separately likely reduces interference between documents, preserves finer semantic distinctions, and produces more reliable ranking behavior. In contrast, the all-passage model may struggle with long-sequence attention dynamics and with disentangling multiple relevance signals within a single forward pass.

Finally, it is important to emphasize that the loss function adopted here constitutes a simplified version of the training objective proposed in the Jina Reranker v3 work. The original model employs a significantly more sophisticated training objective, combining multiple auxiliary losses and regularization terms, which were omitted in this thesis to ensure experimental clarity and reproducibility. As a result, the joint encoder evaluated here may not fully exploit the potential demonstrated in the original architecture, and it remains plausible that more expressive loss formulations could narrow or even reverse the performance gap observed between the two approaches.

5.3 Zero-Shot Evaluation on the BEIR Benchmark

To further assess the generalization capabilities of the trained rankers, we perform a zero-shot evaluation on a subset of the BEIR benchmark [52]. BEIR provides a diverse collection of retrieval tasks covering multiple domains, query styles, and relevance distributions. Evaluating on BEIR therefore offers an external validation step and allows us to determine whether the ranking models learn transferable behaviour beyond the supervised training distribution.

For this purpose, we use the best checkpoint obtained from the independent-encoding approach, which demonstrated the strongest ranking performance. No additional fine-tuning is performed on any BEIR dataset: all evaluations are strictly zero-shot.

The next chapter presents the BEIR evaluation protocol, the retrieval pipeline used to adapt the models to the benchmark, and the resulting performance across multiple BEIR tasks.

5.3.1 BEIR Dataset Analysis and Truncation Feasibility

Before conducting zero-shot evaluation on the BEIR benchmark, all candidate datasets were analysed to assess their suitability for the ModernBERT reranker. Although ModernBERT supports substantially larger context windows, the reranker used in this study was fine-tuned with a maximum input length of 256 tokens. For methodological consistency and computational efficiency during evaluation, the same input constraint was applied in all zero-shot experiments. Consequently, it was necessary to examine the extent to which BEIR documents can be processed without severe truncation.

For each dataset, the number of queries and documents, query and document length distributions, approximate token counts (using a conversion of 1.3 words per token), and the proportion of documents exceeding the 256-token limit were computed.

Dataset	**Docs**	**Queries**	**Mean**$_{\text{tok}}$	**Trunc.256**
scifact	5,183	300	279	53.9%
arguana	8,674	1,406	217	28.3%
scidocs	25,657	1,000	230	29.5%
nfcorpus	3,633	323	304	71.3%
trec-covid	171,332	50	210	42.0%
fiqa	57,638	648	173	18.1%
webis-touche2020	382,545	49	382	39.2%
climate-fever	5,416,593	1,535	127	11.6%
fever	5,416,568	6,666	127	11.6%
dbpedia-entity	4,635,922	400	65	0.0%

Based on the proportion of documents exceeding the 256-token limit, the datasets were grouped into three categories:

- **Good fit (<30% truncation):** scidocs, fiqa, climate-fever, fever, dbpedia-entity, hotpotqa.
- **Penalized (≥30% truncation):** scifact, trec-covid, webis-touche2020, nfcorpus.

This analysis informed the subsequent evaluation strategy. Datasets with minimal truncation provide reliable insight into the zero-shot generalisation ability of the cross-encoder, while datasets with heavy truncation primarily reveal limitations imposed by the restricted 256-token context window inherited from training.

5.3.2 Experimental Setup and Zero-Shot Evaluation

This section outlines the experimental framework used to evaluate the ModernBERT reranker developed in this thesis. The aim is to determine whether the improvements observed during supervised fine-tuning translate to zero-shot performance on out-of-domain retrieval tasks. We describe the retrieval pipeline, the evaluation protocol, and the metrics used, followed by a summary of the results.

Retrieval Pipeline

We use a standard two-stage retrieval pipeline:

1. **Dense retrieval**: Queries and passages are encoded independently using the `intfloat/e5-small` model. Cosine similarity is used to retrieve the top-k candidates for each query.

2. **ModernBERT reranking**: The top-k candidates are re-scored using a ModernBERT cross-encoder fine-tuned on the training data introduced in this thesis. Each query–document pair is jointly encoded and assigned a relevance score through a prediction head.

Throughout all experiments, we set $k = 30$, which balances computational cost with the potential for meaningful reranking.

5.3.3 Results Discussion

The results in Table 5.6 demonstrate that the ModernBERT reranker yields consistent improvements over the E5-small dense retriever across various BEIR datasets. The gains are most evident in NDCG@10 and MRR@10, which reflect the model's ability to promote relevant passages into the highest-ranked positions. Averaged across datasets, ModernBERT increases NDCG@10 from 0.397 to 0.484 and MRR@10 from 0.530 to 0.640, confirming the reranker's effectiveness in improving top-ranked retrieval quality.

While these improvements are broad, their magnitude varies substantially across datasets, and this variation aligns closely with the truncation characteristics reported in Table 5.3.1. Datasets with *low truncation rates*, such as FiQA (18.1%), FEVER (11.6%), and DBPedia-Entity (0.0%), consistently benefit from substantial reranking gains. For example, ModernBERT improves MRR@10 by +0.255 on FEVER and NDCG@10 by +0.067 on DBPedia-Entity. These datasets typically contain short passages well within the 256-token context window, allowing the cross-encoder to fully exploit its joint query–document modelling capability without discarding relevant context.

The largest improvements are observed in semantically complex or evidence-oriented datasets, such as TREC-COVID and HotpotQA. Despite moderate truncation rates, these datasets require deeper reasoning over technical or multi-hop content, and ModernBERT's richer interaction mechanism clearly outperforms the bi-encoder baseline. For TREC-COVID, the reranker raises NDCG@10 from 0.508 to 0.734, one of the strongest relative gains in the benchmark.

In contrast, datasets with *severe truncation*, such as NFCorpus (71.3%) and SciFact (53.9%), exhibit smaller reranking gains. In these cases, a substantial portion of the document text must be discarded to meet the 256-token constraint. This constraint limits the reranker's ability to fully model the underlying evidence, narrowing the gap between the dense retriever and the cross-encoder. Nevertheless, even under such conditions, ModernBERT still provides measurable improvements, reflecting its robustness to partial-context inputs.

Overall, these results show that the ranking behaviour learned during fine-tuning generalises reliably across domains, with especially strong gains on datasets whose documents fit within the training-time context window. Performance degrades gracefully as truncation increases, highlighting the interaction between model capacity, input length constraints, and retrieval task structure.

Table 5.6: Zero-shot BEIR performance of the E5-small dense retriever and the ModernBERT reranker.

Dataset	**NDCG@10$_{\text{E5}}$**	**NDCG@10$_{\text{MB}}$**	**MRR@10$_{\text{E5}}$**	**MRR@10$_{\text{MB}}$**
dbpedia-entity	0.3543	**0.4218**	0.6777	**0.7407**
fever	0.4987	**0.7060**	0.4600	**0.7154**
fiqa	0.3274	**0.3757**	0.3927	**0.4517**
hotpotqa	0.5461	**0.6339**	0.7124	**0.8149**
nfcorpus	0.3353	0.3388	0.5491	0.5518
scidocs	0.1662	0.1816	0.2912	0.3184
scifact	0.6824	0.7070	0.6512	0.6791
trec-covid	0.5081	**0.7343**	0.7282	**0.9500**
webis-touche2020	0.1561	**0.2607**	0.3030	**0.5360**
Average	**0.3972**	**0.4844**	**0.5295**	**0.6398**

Chapter 6

Conclusion

This thesis investigated how modern transformer architectures, specifically ModernBERT, can be effectively leveraged for passage reranking within multi-stage information retrieval pipelines. The work addressed two major limitations in current research: the scarcity of datasets that capture graded, fine-grained relevance, and the lack of systematic analyses on how contemporary encoder models behave when trained under listwise objectives on such data.

The first contribution of this thesis was the construction of a novel English passage-reranking dataset designed to move beyond the traditional "one positive, many negatives" structure that dominates existing benchmarks. Through a carefully engineered pipeline incorporating synthetic query generation, dense-retrieval candidate selection, LLM-based scoring on a 1–10 relevance scale, and structured subsampling, the resulting corpus provides richer semantic diversity and more realistic supervision signals. The integration of a curated subset of the TWOLAR dataset further expanded linguistic variability and introduced challenging negatives, yielding a balanced and comprehensive resource for studying fine-grained ranking behaviour.

Building upon this corpus, the second major contribution consisted of an extensive empirical investigation of ModernBERT under two encoding strategies: *Independent Passage Encoding* and *All-Passage Joint Encoding*. Despite the theoretical advantages of joint modelling, the empirical results showed that independently encoding each query–passage pair produced more stable learning dynamics and superior overall performance. This finding aligns with the structure of the dataset, where nuanced relevance distinctions are more effectively captured when passages are processed independently, without cross-passage interference. The listwise learning-to-rank objective proved effective in leveraging the graded labels, enabling the model to learn relative ordering patterns rather than relying on coarse binary signals.

Finally, the zero-shot evaluation on selected BEIR datasets provided an external validation of the model's generalisation capability. The fine-tuned ModernBERT reranker yielded consistent improvements over the dense retrieval baseline across a variety of domains. The magnitude of these improvements correlated strongly with truncation characteristics, confirming that the 256-token training constraint plays a central role in downstream performance. Datasets characterised by shorter passages benefited the most, while tasks requiring long-context reasoning showed more moderate gains. Nonetheless, even in severely truncated settings, the model remained robust and delivered measurable improvements.

Overall, this thesis demonstrates that ModernBERT, when trained on a carefully curated graded-relevance corpus, is an effective and generalisable reranker, capable of outperforming strong baselines and transferring its learned ranking behaviour to out-of-domain retrieval tasks. The work additionally provides new data resources and methodological insights that can facilitate future research in neural reranking and information retrieval.

Future directions include extending the context window during training, refining joint-encoding architectures with more expressive objectives, incorporating retrieval-aware pretraining, and exploring larger-scale relevance annotation pipelines. The dataset and analyses presented in this thesis form a solid foundation for these developments and contribute meaningfully to the evolution of transformer-based reranking systems.

Bibliography

[1] A. Abdallah, J. Mozafari, B. Piryani, M. M. Abdelgwad, and A. Jatowt. DynRank: improve passage retrieval with dynamic zero-shot prompting based on question classification. In O. Rambow, L. Wanner, M. Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4768–4778, Abu Dhabi, UAE. Association for Computational Linguistics, January 2025. url: `https://aclanthology.org/2025.coling-main.319/`.

[2] Z. Akkalyoncu Yilmaz, W. Yang, H. Zhang, and J. Lin. Cross-domain modeling of sentence-level evidence for document retrieval. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3490–3496, Hong Kong, China. Association for Computational Linguistics, November 2019. doi: `10.18653/v1/D19-1352`. url: `https://aclanthology.org/D19-1352/`.

[3] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. Ms marco: a human generated machine reading comprehension dataset, 2018. arXiv: `1611.09268 [cs.CL]`. url: `https://arxiv.org/abs/1611.09268`.

[4] D. Baldelli, J. Jiang, A. Aizawa, and P. Torroni. Twolar: a two-step llm-augmented distillation method for passage reranking, 2024. arXiv: 2403.17759 [cs.IR]. url: https://arxiv.org/abs/2403.17759.

[5] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: the long-document transformer, 2020. arXiv: 2004.05150 [cs.CL]. url: https://arxiv.org/abs/2004.05150.

[6] C. Burges. From ranknet to lambdarank to lambdamart: an overview. *Learning*, 11, January 2010.

[7] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. url: https://proceedings.neurips.cc/paper_files/paper/2006/file/af44c4c56f385c43f2529f9b1b018f6a-Paper.pdf.

[8] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In pages 89–96, January 2005. doi: 10.1145/1102351.1102363.

[9] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu. Bge m3-embedding: multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024. arXiv: 2402.03216 [cs.CL].

[10] Y. Chen, Q. Liu, Y. Zhang, W. Sun, X. Ma, W. Yang, D. Shi, J. Mao, and D. Yin. Tourrank: utilizing large language models for documents ranking with a tournament-inspired strategy, 2025. arXiv: 2406.11678 [cs.IR]. url: https://arxiv.org/abs/2406.11678.

[11] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: pre-training text encoders as discriminators rather than generators, 2020. arXiv: 2003.10555 [cs.CL]. url: https://arxiv.org/abs/2003.10555.

[12] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos. Overview of the trec 2020 deep learning track, 2021. arXiv: 2102.07662 [cs.IR]. url: https://arxiv.org/abs/2102.07662.

[13] Z. Dai and J. Callan. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 985–988. ACM, July 2019. doi: 10.1145/3331184.3331303. url: http://dx.doi.org/10.1145/3331184.3331303.

[14] T. Dao. Flashattention-2: faster attention with better parallelism and work partitioning, 2023. arXiv: 2307.08691 [cs.LG]. url: https://arxiv.org/abs/2307.08691.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding, 2019. arXiv: 1810.04805 [cs.CL]. url: https://arxiv.org/abs/1810.04805.

[16] T. Formal, B. Piwowarski, and S. Clinchant. Splade: sparse lexical and expansion model for first stage ranking, 2021. arXiv: 2107.05720 [cs.IR]. url: https://arxiv.org/abs/2107.05720.

[17] J. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29, November 2000. doi: 10.1214/aos/1013203451.

[18] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM'16, pages 55–64. ACM, October 2016. doi: 10.1145/2983323.2983769. url: http://dx.doi.org/10.1145/2983323.2983769.

[19] P. He, X. Liu, J. Gao, and W. Chen. Deberta: decoding-enhanced bert with disentangled attention, 2021. arXiv: 2006.03654 [cs.CL]. url: https://arxiv.org/abs/2006.03654.

[20] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, San Francisco, California, USA. Association for Computing Machinery, 2013. isbn: 9781450322638. doi: 10.1145/2505515.2505665. url: https://doi.org/10.1145/2505515.2505665.

[21] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. L. Lee, M. Johnson, B. Roark, and A. Nenkova, editors. doi: 10.1162/tacl_a_00276. url: https://aclanthology.org/Q19-1026/.

[22] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: a lite bert for self-supervised learning of language representations, 2020. arXiv: 1909.11942 [cs.CL]. url: https://arxiv.org/abs/1909.11942.

[23] C. Li, A. Yates, S. MacAvaney, B. He, and Y. Sun. Parade: passage representation aggregation for document reranking, 2021. arXiv: 2008.09093 [cs.IR]. url: https://arxiv.org/abs/2008.09093.

[24] J. Lin, R. Nogueira, and A. Yates. Pretrained transformers for text ranking: bert and beyond, 2021. arXiv: 2010.06467 [cs.IR]. url: https://arxiv.org/abs/2010.06467.

[25] S.-C. Lin, A. Asai, M. Li, B. Oguz, J. Lin, Y. Mehdad, W.-t. Yih, and X. Chen. How to train your dragon: diverse augmentation towards generalizable dense retrieval, 2023. arXiv: 2302.07452 [cs.IR]. url: https://arxiv.org/abs/2302.07452.

[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: a robustly optimized bert pretraining approach, 2019. arXiv: 1907.11692 [cs.CL]. url: https://arxiv.org/abs/1907.11692.

[27] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, July 1960. issn: 0004-5411. doi: 10.1145/321033.321035. url: https://doi.org/10.1145/321033.321035.

[28] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality, 2013. arXiv: 1310.4546 [cs.CL]. url: https://arxiv.org/abs/1310.4546.

[29] B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search, 2016. arXiv: 1610.08136 [cs.IR]. url: https://arxiv.org/abs/1610.08136.

[30] B. Mitra, E. Nalisnick, N. Craswell, and R. Caruana. A dual embedding space model for document ranking, 2016. arXiv: 1602.01137 [cs.IR]. url: https://arxiv.org/abs/1602.01137.

[31] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang. Sentence-t5: scalable sentence encoders from pre-trained text-to-text models, 2021. arXiv: 2108.08877 [cs.CL]. url: https://arxiv.org/abs/2108.08877.

[32] R. Nogueira and K. Cho. Passage re-ranking with bert, 2020. arXiv: 1901.04085 [cs.IR]. url: https://arxiv.org/abs/1901.04085.

[33] R. Nogueira, Z. Jiang, and J. Lin. Document ranking with a pretrained sequence-to-sequence model, 2020. arXiv: 2003.06713 [cs.IR]. url: https://arxiv.org/abs/2003.06713.

[34] R. Nogueira and J. Lin. From doc2query to docTTTTTquery. Technical report, Epistemic AI / David R. Cheriton School of Computer Science, University of Waterloo, 2019. url: https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf. Online preprint, 6(2).

[35] R. Nogueira, W. Yang, K. Cho, and J. Lin. Multi-stage document ranking with bert, 2019. arXiv: 1910.14424 [cs.IR]. url: https://arxiv.org/abs/1910.14424.

[36] OpenAI. Gpt-4 technical report, 2024. arXiv: 2303.08774 [cs.CL]. url: https://arxiv.org/abs/2303.08774.

[37] OpenAI. Gpt-4o mini model documentation. https://platform.openai.com/docs/models/gpt-4o-mini. Accessed: 2025-10-17.

[38] L. Pang, Y. Lan, J. Guo, J. Xu, and X. Cheng. A study of matchpyramid models on ad-hoc retrieval, 2016. arXiv: 1606.04648 [cs.IR]. url: https://arxiv.org/abs/1606.04648.

[39] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. arXiv: 1910.10683 [cs.LG]. url: https://arxiv.org/abs/1910.10683.

[40] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text, 2016. arXiv: 1606.05250 [cs.CL]. url: https://arxiv.org/abs/1606.05250.

[41] N. Reimers and I. Gurevych. Sentence-transformers: multi-qa-mpnet-base-dot-v1. https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1. Accessed: 2025-02-10.

[42] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. K. Harman, editor, *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST), 1994. url: `http://trec.nist.gov/pubs/trec3/papers/city.ps.gz`.

[43] A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in bertology: what we know about how bert works, 2020. arXiv: `2002.12327 [cs.CL]`. url: `https://arxiv.org/abs/2002.12327`.

[44] D. S. Sachan, M. Lewis, M. Joshi, A. Aghajanyan, W.-t. Yih, J. Pineau, and L. Zettlemoyer. Improving passage retrieval with zero-shot question generation, 2023. arXiv: `2204.07496 [cs.CL]`. url: `https://arxiv.org/abs/2204.07496`.

[45] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, November 1975. issn: 0001-0782. doi: `10.1145/361219.361220`. url: `https://doi.org/10.1145/361219.361220`.

[46] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. arXiv: `1910.01108 [cs.CL]`. url: `https://arxiv.org/abs/1910.01108`.

[47] Sentence-transformers/all-minilm-l6-v2. `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`. Accessed: 2025-10-17.

[48] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani, and T. Dao. Flashattention-3: fast and accurate attention with asynchrony and low-precision, 2024. arXiv: `2407.08608 [cs.LG]`. url: `https://arxiv.org/abs/2407.08608`.

[49] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 101–110, Shanghai, China. Association for Computing Machinery, 2014. isbn: 9781450325981. doi: 10.1145/2661829.2661935. url: https://doi.org/10.1145/2661829.2661935.

[50] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu. Roformer: enhanced transformer with rotary position embedding, 2023. arXiv: 2104.09864 [cs.CL]. url: https://arxiv.org/abs/2104.09864.

[51] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren. Is chatgpt good at search? investigating large language models as reranking agents, 2024. arXiv: 2304.09542 [cs.CL]. url: https://arxiv.org/abs/2304.09542.

[52] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. Beir: a heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021. arXiv: 2104.08663 [cs.IR]. url: https://arxiv.org/abs/2104.08663.

[53] T. A. Tran. Rank_bm25: okapi bm25 implementation in python. https://github.com/dorianbrown/rank_bm25. Accessed: 2025-10-17.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. arXiv: 1706.03762 [cs.CL]. url: https://arxiv.org/abs/1706.03762.

[55] F. Wang, Y. Li, and H. Xiao. Jina-reranker-v3: last but not late interaction for listwise document reranking, 2025. arXiv: 2509.25085 [cs.CL]. url: https://arxiv.org/abs/2509.25085.

[56] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024. arXiv: 2212.03533 [cs.CL]. url: https://arxiv.org/abs/2212.03533.

[57] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, et al. Smarter, better, faster, longer: a modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*, 2024.

[58] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: bridging the gap between human and machine translation, 2016. arXiv: 1609.08144 [cs.CL]. url: https://arxiv.org/abs/1609.08144.

[59] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020. arXiv: 2007.00808 [cs.IR]. url: https://arxiv.org/abs/2007.00808.

[60] S. Yoon, E. Choi, J. Kim, H. Yun, Y. Kim, and S.-w. Hwang. Listt5: listwise reranking with fusion-in-decoder improves zero-shot retrieval, 2024. arXiv: 2402.15838 [cs.IR]. url: https://arxiv.org/abs/2402.15838.

[61] H. Zhuang, Z. Qin, R. Jagerman, K. Hui, J. Ma, J. Lu, J. Ni, X. Wang, and M. Bendersky. Rankt5: fine-tuning t5 for text ranking with ranking losses, 2022. arXiv: 2210.10634 [cs.IR]. url: https://arxiv.org/abs/2210.10634.