M.Tech. (Computer Science) Dissertation Series

# Exploring Deep Learning for IR within the TREC DL Track

Submitted in partial fulfillment of

the requirements for the degree of

**Master of Technology in Computer Science**

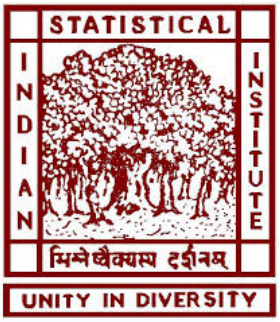
By

**Ritesh Kumar Tiwary**

Roll No: CS2224

Under the supervision of

**Dr. Mandar Mitra**




INDIAN STATISTICAL INSTITUTE

203, Barrackpore Trunk Road

Kolkata-700108

**Indian Statistical Institute**
203, B.T. Road, Kolkata : 700108

# CERTIFICATE

I certify that I have read the thesis titled **Exploring Deep Learning for IR within the TREC DL Track**, prepared under my guidance by **Ritesh Kumar Tiwary**, and in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of **Master of Technology in Computer Science** of the Indian Statistical Institute.

Mandar Mitra
Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute

Kolkata
June, 2024.

# Declaration

I, Ritesh Kumar Tiwary, declare that this dissertation titled, **"Exploring Deep Learning for IR within the TREC DL Track"**, which is submitted in fulfillment of the requirements for the Degree of Master of Technology in Computer Science, represents my own work except where due acknowledgement has been made. I further declare that it has not been previously included in a thesis, dissertation, or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed: Ritesh Tiwary

Date: 21 June 2024

# *Acknowledgements*

I would like to express my deepest gratitude to my supervisor, Dr. Mandar Mitra, whose expert guidance, infinite patience, and constant encouragement have endowed me with a strong foundation in Information Retrieval and made this dissertation possible. His insightful feedback and unwavering support throughout my dissertation journey have been instrumental in shaping this work. I feel truly blessed to have had the opportunity to work under his mentorship.

I would also like to extend my heartfelt thanks to Mr. Saurav Saha for his invaluable assistance and support throughout my dissertation work. His willingness to help and his constructive advice have greatly contributed to the completion of this thesis.

Ritesh Kumar Tiwary
Indian Statistical Institute
June 12, 2024

# Contents

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **IR** | **I**nformation **R**etrieval |
| **NLP** | **N**atural **L**anguage **P**rocessing |
| **BERT** | **B**idirectional **E**ncoder **R**epresentations from **T**ransformers |
| **PLM** | **P**re-trained **L**anguage **M**odel |
| **MS MARCO** | **M**icro**S**oft **MA**chine **R**eading **CO**mprehension |
| **TREC** | **T**ext **RE**trieval **C**onference |
| **NIST** | **N**ational **I**nstitute of **S**tandards and **T**echnology |
| **IARPA** | **I**ntelligence **A**dvanced **R**esearch **P**rojects **A**ctivity |
| **NDCG** | **N**ormalized **D**iscounted **C**umulative **G**ain |
| **MAP** | **M**ean **A**verage **P**recision |
| **MRR** | **M**ean **R**eciprocal **R**ank |
| **NLU** | **N**atural Language **U**nderstanding |
| **NCE** | **N**oise **C**ontrastive **E**stimation |
| **LCE** | **L**ocalized **C**ontrastive **E**stimation |
| **SPLADE** | **SP**arse **L**exical **A**nd **D**ense **E**mbeddings |
| **MLM** | **M**asked **L**anguage **M**odel |
| **HDCT** | **H**ierarchical **D**ocument **C**ontext **T**erm |

# 1 Introduction

Information Retrieval (IR) is crucial in today's world, underpinning many practical applications such as web searches, question-answering systems, personal assistants, chatbots, and digital libraries. The primary aim of IR is to find and retrieve information that matches a user's query. Since multiple records can be relevant, IR systems often rank the results based on their relevance to the query.

## 1.1 Neural IR

Traditional text retrieval systems primarily rely on matching terms between a query and the documents. However, these term-based retrieval systems face several challenges, such as polysemy (words with multiple meanings), synonymy (different words with the same meaning), and lexical gaps between the query and the documents [11].

In recent years, advancements in computing power and the availability of large labeled datasets have significantly influenced the field of Natural Language Processing (NLP). More specifically these advancements have enabled researchers to apply deep learning techniques, leading to numerous innovations. By leveraging deep learning, traditional text retrieval systems have also been improved to address the limitations of term-based retrieval systems.

However, implementing these advanced techniques demands substantial amounts of data and computing resources. Consequently, researchers are continuously developing more sophisticated deep learning algorithms to meet these requirements and achieve superior results in NLP tasks [26]. With these advanced algorithms, the performance of IR systems has markedly improved, resulting in more accurate and efficient information retrieval for end-users.

Several advancements in deep learning techniques have been applied to IR, including neural network architectures such as convolutional neural networks (CNNs) [25] and recurrent neural networks (RNNs) [27]. Additionally, transfer learning and pre-training techniques have been employed to enhance text data representation and improve IR systems' ability to understand natural language queries [15].

Moreover, attention-based mechanisms, like the Transformer architecture [36], have been utilized to improve IR systems' capability to focus on the most relevant parts of the query and documents. Pre-trained language models, such as BERT [15] and GPT-2 [34], have further demonstrated their effectiveness in enhancing IR systems by providing a deeper understanding of the semantics and context of natural language queries and documents. These advancements have collectively contributed to the enhanced performance and reliability of modern IR systems.

## 1.2 About TREC

TREC, co-sponsored by NIST and IARPA, began in 1992 as part of the TIPSTER Text program[1]. Its primary purpose is to support and foster research in the IR community by providing a platform for large-scale evaluation of text retrieval methodologies and accelerating the transfer of technology from research labs to commercial products[35].

Each TREC track features a challenge where NIST supplies data sets and test problems to participating groups [2]. These problems may involve questions, topics, or target extractable features. Uniform scoring ensures fair evaluation, and the results are discussed in a workshop that facilitates the exchange of ideas and future research directions.

Relevance judgments in TREC define relevance as the utility of a document's information for a report on the topic. Most tasks use binary relevance, while some employ graded relevance to capture varying degrees of relevance. The pooling method aggregates top-ranked documents from each run for evaluation, as complete relevance assessment is impractical for large collections[23].

### 1.2.1 TREC Deep Learning Track

The Deep Learning Track in TREC focuses on studying information retrieval within a large training data regime. This scenario involves having a substantial number of training queries, each with at least one positive label, potentially numbering in the tens or hundreds of thousands. Such a regime mirrors real-world applications, such as training models using click logs or labels from shallow pools[7], like those used in the TREC Million Query Track or evaluations based on early precision metrics [3].

Machine learning methods, particularly those based on deep learning, often require extensive datasets for effective training. The scarcity of large-scale datasets has historically impeded the development of these methods for common IR tasks, such as document ranking. The Deep Learning Track addresses this limitation by providing large-scale datasets and fostering a concentrated research effort.

The Deep Learning Track aims to advance the understanding and development of deep learning methods in information retrieval, ultimately enhancing the performance and applicability of these methods in real-world scenarios.

## 1.3 Problem Statement

The Deep Learning track, introduced at TREC 2019, focuses on exploring information retrieval with large training datasets. The TREC DL 2019 dataset is foundational, establishing baseline benchmarks for subsequent years, and is highly stable and consistent, reducing complications from later variations. It has been extensively studied and cited, offering a rich body of literature and comparative analyses. Using the 2019 dataset allows for historical comparisons, highlighting methodological advancements. This makes TREC DL 2019 a robust choice over datasets from 2020-2023.

This track includes two main tasks: passage ranking and document ranking, both of which use a substantial set of human-generated training labels from the MS MARCO* dataset. Each task has two associated subtasks: full ranking and top-k re-ranking. In the full ranking subtask, the goal is to rank passages or documents directly from the entire provided document collection. In the re-ranking subtask, the aim is to re-rank based on an initial set of top-k passages or documents.

For official evaluation, depth pooling is used, creating separate pools for passage ranking and document ranking tasks. NIST assessors then label passages and documents in these pools using multi-graded judgments, allowing for the measurement of the NDCG metric. Out of the 200 test queries, 43 ground truth queries are selected through the process described in Appendix B and used for evaluation.

In our study, we aim to advance the field of information retrieval by leveraging state-of-the-art transformer models to enhance document ranking performance. Our primary objective is to compare these advanced models—bert-base-uncased[15],

*https://msmarco.org/

deberta-v3[20], and Naver's *splade-cocondenser-ensembledistil* and *splade-cocondenser-selfdistil*[17]—by achieving competitive results. Additionally, we investigate the impact of hyperparameter tuning on their effectiveness. By pushing the boundaries of current methodologies, we seek to provide more accurate and efficient retrieval systems, contributing valuable insights and advancements to the research community.

This report is structured as follows: Chapter 2 provides a detailed introduction to the approach used in our experiments, Chapters 3 and 4 presents the experimental settings and results respectively. Chapter 5 concludes the report.

## 1.4 Datasets

Both tasks utilize large training sets based on human relevance assessments from MS MARCO. These assessments are sparse, featuring no negative labels and often only one positive label per query, similar to real-world training data such as click logs [10]. The document corpus, newly released for use in TREC, includes three fields for each document: (i) URL, (ii) title, and (iii) body text. Table 1.1 provides descriptive statistics for the datasets. More information about the datasets, including download instructions, is available on the TREC 2019 Deep Learning Track website[†].

**Table 1.1:** Summary of statistics on TREC 2019 Deep Learning Track datasets.

| **File description** | **Document retrieval dataset** | | **Passage retrieval dataset** | |
|---|---|---|---|---|
| | Number of records | File size | Number of records | File size |
| Collection | 3,213,835 | 22 GB | 8,841,823 | 2.9 GB |
| Train queries | 367,013 | 15 MB | 502,940 | 19.7 MB |
| Train qrels | 384,597 | 7.6 MB | 532,761 | 10.1 MB |
| Validation queries | 5,193 | 216 KB | 12,665 | 545 KB |
| Validation qrels | 519,300 | 27 MB | 59,273 | 1.1 MB |
| Test queries | 200 | 12 KB | 200 | 12 KB |

## 1.5 Applied Models

In this section, we will explain the various models used in our work.

[†]https://microsoft.github.io/msmarco/TREC-Deep-Learning-2019

### 1.5.1 BERT-Base-Uncased

BERT (Bidirectional Encoder Representations from Transformers) is a method for pre-training language representations. It involves training a general-purpose "language understanding" model on a large text corpus, such as Wikipedia, which is then used for downstream NLP tasks like question answering. BERT surpasses previous methods because it is the first unsupervised, deeply bidirectional system for pre-training NLP. Unsupervised training means BERT was trained using only plain text data, making use of the vast amount of publicly available text on the web in various languages.

In this work, we used the pre-trained BERT-Base-Uncased model for tokenization. The tokenization process followed three key steps:

1. **Text Normalization:** Convert all whitespace characters to spaces. Lowercase the input and strip out accent markers for the Uncased model. Example: "John Johanson's," becomes "john johanson's,".

2. **Punctuation Splitting:** Split all punctuation characters on both sides (i.e., add whitespace around all punctuation characters). Punctuation characters are defined as anything with a P* Unicode class. Any non-letter/number/space ASCII character (e.g., characters like $ which are technically not punctuation). Example: "john johanson's," becomes "john johanson ' s ,".

3. **WordPiece Tokenization:** Apply whitespace tokenization to the output of the above procedure. Apply WordPiece tokenization to each token separately. Example: " john johanson ' s ," becomes " john johan ##son ' s ," .

**Table 1.2:** Comparison of BERT models

| Model | Layers | Hidden Units | Heads | Parameters |
|---|---|---|---|---|
| BERT-Base, Uncased | 12 | 768 | 12 | 110M |
| BERT-Large, Uncased | 24 | 1024 | 16 | 340M |
| BERT-Base, Cased | 12 | 768 | 12 | 110M |
| BERT-Large, Cased | 24 | 1024 | 16 | 340M |

### 1.5.2 DeBERTa-V3

In our experiments, we also utilized the DeBERTa-V3 model. DeBERTa (Decoding-enhanced BERT with disentangled attention) is a Transformer-based neural language model that introduces two novel techniques to improve upon previous state-of-the-art pre-trained language models (PLMs): a disentangled attention mechanism

and an enhanced mask decoder.

**Disentangled Attention:** Unlike BERT, where each word in the input layer is represented by a single vector (the sum of its word (content) embedding and position embedding), DeBERTa represents each word with two separate vectors that encode its content and position. The attention weights among words are computed using disentangled matrices based on their content and relative positions. This approach recognizes that the attention weight between word pairs depends on both their contents and their relative positions. For instance, the dependency between "deep" and "learning" is stronger when they are adjacent than when they are in different sentences.

**Enhanced Mask Decoder:** Similar to BERT, DeBERTa[21] is pre-trained using masked language modeling (MLM), where the model predicts masked words based on the surrounding context. DeBERTa uses content and position information for MLM. While the disentangled attention mechanism considers the contents and relative positions of context words, it does not account for their absolute positions, which can be crucial for prediction. For example, in the sentence "a new store opened beside the new mall," understanding the syntactic roles of "store" and "mall" depends on their absolute positions. DeBERTa incorporates absolute word position embeddings before the softmax layer to enhance the prediction of masked words based on aggregated contextual embeddings.

In further development, DeBERTa-V3[20] was introduced, which improves the original DeBERTa by replacing MLM with replaced token detection (RTD), a more sample-efficient pre-training task. Analysis showed that vanilla embedding sharing in ELECTRA[9] reduces training efficiency and model performance due to conflicting training losses between the discriminator and generator. DeBERTa-V3 addresses this with a gradient-disentangled embedding sharing method, improving training efficiency and model quality. Pre-trained under the same settings as DeBERTa, DeBERTa-V3 demonstrates exceptional performance across a variety of downstream natural language understanding (NLU) tasks.

**Table 1.3:** Comparison of Different DeBERTa-V3 models

| Model | Vocabulary (K) | Parameters (M) | Hidden Size | Layers |
|---|---|---|---|---|
| DeBERTa-V3-Large | 128 | 304 | 1024 | 24 |
| DeBERTa-V3-Base | 128 | 86 | 768 | 12 |
| DeBERTa-V3-Small | 128 | 44 | 768 | 6 |
| DeBERTa-V3-XSmall | 128 | 22 | 384 | 12 |

# 2 Methodologies

## 2.1 Background

The separation of retrieval into stages arose naturally due to the efficiency-effectiveness trade-off among different ranking models: fast but less accurate models (e.g., BM25) retrieve from the entire corpus, while slower but more accurate models (e.g., BERT) refine the ranking in the top candidate list. Heuristic retrievers like BM25 rely solely on exact match signals, allowing the use of inverted list data structures for low-latency full corpus retrieval. However, their scoring is limited by document statistics. To address this, deep language models can be used to re-estimate term weights in the search index [12, 14]. Alternatively, probable query terms can be added to documents [29].

Pre-trained deep language models [33, 15] have shown strong supervised transfer performance on reranking tasks. Recent popular works [28, 13] fine-tune BERT [15] with a binary classification objective, demonstrating significant improvements over earlier models. However, we question whether this simple paradigm fully realizes BERT's potential, especially for high-performance deep retrievers that generate candidates with harder negatives.

An alternative to the binary classification objective is contrastive learning objectives, which directly consider negatives [19]. The popular Noise Contrastive Estimation (NCE) loss calculates scores for a positive instance and several negative instances, normalizes them into probabilities, and trains the model to assign a higher probability to the positive instance [38]. Incorporating negatives in the loss prevents the model from collapsing. While contrastive loss has been extensively studied in representation learning [38, 8], there are few prior works adopting it for training deep language model rerankers.

## 2.2 Preliminaries

We aim to train a BERT reranker to score a query-document pair using the following formula:

$$s = \text{score}(q, d) = \mathbf{v_p}^\top cls(\text{BERT}(\text{concat}(q, d))) \tag{2.1}$$

where *cls* extracts BERT's [CLS] vector and $\mathbf{v_p}$ is a projection vector. This approach, commonly referred to as the *Vanilla method* [13, 28], involves sampling query-document pairs independently and computing on each individual pair using binary cross-entropy (BCE) based on the query $q$, document $d$, and the corresponding label (positive/negative).

The loss function for the Vanilla method is defined as:

$$L_v := \begin{cases} \text{BCE}(\text{score}(q, d), +) & \text{if } d \text{ is positive} \\ \text{BCE}(\text{score}(q, d), -) & \text{if } d \text{ is negative} \end{cases} \tag{2.2}$$

The Vanilla method treats the reranker training as a general binary classification problem. However, rerankers have a unique nature; they handle the top portion of retriever results, each potentially containing many confounding features. Therefore, the reranker must exile at handling the top portion of retriever results and avoid collapsing by not matching with confounding features [18]. To address these challenges, we introduce Localized Contrastive Estimation (LCE) loss in this section. The contrastive loss prevents the model from collapsing, while localized negative samples focus the reranker on the top retriever results, ensuring it handles these effectively and avoids confounding features.

## 2.3 Localized Negatives from Target Retriever

Given a target initial-stage retriever and a set of training queries, we utilize the retriever to search the entire corpus, producing a set of document rankings for these queries. For each query $q$, we then sample $n$ non-relevant documents as negative examples from the set $R_q^m$ of the top $m$ ranked documents. These sampled documents collectively form the negative training set. Reconstructing the training set based on the specific target retriever is crucial for ensuring robust training [18].

## 2.4 Contrastive Loss

After aggregating all negatives sampled from the target retriever, we form a group $G_q$ for each query $q$. This group consists of a single relevant positive document $d_q^+$ and the sampled non-relevant negative documents from $R_q^m$. We treat the BERT scoring function as a deep distance function,

$$dist(q,d) = \text{score}(q,d) = \mathbf{v_p}^\top cls(\text{BERT}(\text{concat}(q,d))) \tag{2.3}$$

The contrastive loss for a query $q$ is then defined as,

$$L_q := -\log \frac{\exp(\text{dist}(q, d^+q))}{\sum d \in G_q \exp(\text{dist}(q,d))} \tag{2.4}$$

In this formulation, the loss and gradient are conditioned not only on the relevant pair but also on the retrieved negatives. This approach helps prevent the model from collapsing into simple confounding matchings[18].

## 2.5 LCE Batch Update

Bringing everything together, we can express the Localized Contrastive Estimation (LCE) loss for a training batch of a set of queries $Q$ as follows:

$$L_{\text{LCE}} := \frac{1}{|Q|} \sum_{q \in Q, G_q \sim R_m^q} -\log \frac{\exp(\text{dist}(q, d_q^+))}{\sum_{d \in G_q} \exp(\text{dist}(q,d))} \tag{2.5}$$

In contrast to the standard noise contrastive estimation (NCE) loss, LCE employs the target retriever to localize negative samples, thereby concentrating the learning process on the top portion rather than on randomly sampled noisy negatives.

## 2.6 SPLADE and its Methodology

### 2.6.1 SPLADE

SPLADE (Sparse Lexical and Dense Embeddings) is a sparse retrieval model that predicts term importance using the BERT WordPiece vocabulary. It leverages the Masked Language Modeling (MLM) layer from pre-training to perform implicit term

expansion[17]. For a given query or document $t$, let $w_{i,j}$ denote the importance of the $j$-th vocabulary token for the $i$-th input token. Text representations are obtained by pooling these importance predictors over the input sequence with a log-saturation effect. SPLADE originally used sum pooling, but experimental findings revealed that max pooling significantly improves performance [24]. Therefore, the formulation used is:

$$w_j = \max_{i \in t} \log(1 + \text{ReLU}(w_{i,j})) \tag{2.6}$$

The ranking score *s(q,d)* is computed as the dot product between the query $q$ and document $d$ representations.

**Training**

The model is trained using a query $q$, a positive document $d^+$, a negative document $d^-$ mined from BM25, and additional in-batch negatives $d_j^-$ [16]. The training objective combines a contrastive InfoNCE loss [30] and FLOPS regularization [31] to achieve the desired sparsity:

$$L = L_{\text{InfoNCE,BM25}} + \lambda_q L_q^{\text{FLOPS}} + \lambda_d L_d^{\text{FLOPS}} \tag{2.7}$$

### 2.6.2 Distillation, Hard Negative Mining, and PLM Initialization

To improve SPLADE, several techniques are applied, such as distillation, hard negative mining, and pre-trained language model (PLM) initialization[16].

**Distillation:** Using MarginMSE loss[22], SPLADE distillation optimizes the model by matching the positive-negative margins of a cross-encoder teacher and the student model.

$$L = L_{\text{MarginMSE,BM25}} + \lambda_q L_q^{\text{FLOPS}} + \lambda_d L_d^{\text{FLOPS}} \tag{2.8}$$

**Hard Negative Mining:** This involves generating higher quality negative samples to enhance training, replacing standard BM25 negatives with negatives mined from more sophisticated models [16].

**Pre-training:** Utilizing pre-trained checkpoints from retrieval-oriented tasks, such as CoCondenser, improves SPLADE's performance by initializing the model with embeddings that contain more informative knowledge for retrieval tasks[16].

### 2.6.3 SPLADE++ Models

The improved SPLADE strategies are collectively referred to as SPLADE++[16]. For our experiments, we used two configurations:

1. **CoCondenser-EnsembleDistil**
   CoCondenser-EnsembleDistil combines CoCondenser initialization with ensemble mining for hard negatives, followed by distillation[16]. This approach leverages the msmarco-hard-negatives dataset, which includes hard negatives mined from various dense retrievers and BM25. The ranking loss for this scenario is:

$$L = L_{\text{MarginMSE,ensemble}} + \lambda_q L_q^{\text{FLOPS}} + \lambda_d L_d^{\text{FLOPS}} \tag{2.9}$$

   By using multiple models to mine negatives, the ensemble approach aims to improve the quality of negative samples, leading to better training and performance.

2. **CoCondenser-SelfDistil**
   CoCondenser-SelfDistil combines CoCondenser initialization with self-mining for hard negatives, followed by distillation[16]. This involves a two-step training process:

   (a) Train a SPLADE model and a cross-encoder re-ranker using distillation.

   (b) Generate triplets using the trained SPLADE model and score them with the cross-encoder for another round of training.

   The ranking loss for this scenario is:

$$L = L_{\text{MarginMSE,self}} + \lambda_q L_q^{\text{FLOPS}} + \lambda_d L_d^{\text{FLOPS}} \tag{2.10}$$

   This self-distillation approach aims to improve the quality of negative samples by iteratively refining the training process.

# 3 Experimental Setup

## 3.1 Initial Stage Retriever

We experimented with the initial retriever HDCT [14]. HDCT (Hierarchical Document Context Term) is the method for augmenting document search indices with term weights re-estimated using BERT. We used the rankings provided by the authors [4]. The top 100 candidate lists from HDCT were input to our rerankers for further experimentation. This approach enhances the traditional bag-of-words (BoW) representation by incorporating context-aware term weighting. The framework operates in two main stages:

1. **Semantic Importance Estimation:** HDCT first estimates the semantic importance of each term within the context of individual passages of the document. This process involves understanding the role and relevance of a term relative to the surrounding text, thereby capturing more nuanced information than simple term frequency.

2. **Aggregation and Indexing:** The context-aware term weights derived in the first stage are then aggregated into a document-level representation. This enriched BoW representation is compatible with standard inverted indexes, ensuring that the enhanced document vectors can be stored and retrieved efficiently using existing search engine infrastructure.

## 3.2 Implementation

Our experiments are conducted on the TREC Deep Learning (DL) Track 2019 dataset using four models: two from HuggingFace (BERT-base and DeBERTa-V3) and two from Naver Labs Europe (NLE) (CoCondenser-selfDistil and CoCondenser-EnsembleDistil). We detail the implementation and training setup for these models in Appendix A.

We follow the setup inspired by [13], where the input to the rerankers consists of the concatenated document title, URL, and the first 512 tokens of the document body. This approach ensures that the model processes a rich context for each

document.

For our specific implementation, we used the top 100 candidate lists as input from HDCT. A Python script, `build_train_from_ranking.py`, generated JSON files for all 367,013 queries. Each JSON file contains the query ID (qid), tokenized query, and both positive and negative examples. The positive and negative examples include document IDs (did) and their truncated tokenized document bodies. These JSON files are used for positive and negative sampling as well as hard negative mining.

During training and evaluation, we randomly sampled 10 documents from the top 100 documents for each query. These 10 documents were divided into 8 for the training set and 2 for the evaluation set. We saved checkpoints in the checkpoint directory after every 2000 or 6000 steps. The maximum number of tokens per document was limited to 512.

The rerankers are implemented in mixed precision using PyTorch [32] and are based on HuggingFace's BERT implementation [37] as well as Naver's SPLADE implementation[5]. Negative sampling is performed from the top 100 documents ranked by the target retriever, similar to the reranking depth. This strategy ensures that the model is trained on challenging examples, improving its ranking performance.

Additionally, we preprocessed the data to create a `test.d100.tsv` file. This file's columns include qid, query, did, URL, title, body, and an unused field. This preprocessing step ensured that our test data was in a consistent format for evaluation purposes.

The output of our reranking process is a TSV file containing the query ID (qid), document ID (did), and the score value. These files are used to rank the documents for test queries, which are further used for evaluation purposes. All the score files along with the detailed processes are uploaded to GitHub repository*.

For the training process, we utilized the GitHub repository of Reranker by cloning to download requirement files as method mentioned in below Github Repo. This setup, detailed in Appendix A, ensures a comprehensive training regime, leveraging state-of-the-art techniques for efficient and effective training of the reranker models on the TREC DL Track 2019 dataset.

*https://github.com/RiteshKTiwary/Reranking-of-TREC-Deep-Learning-2019-Track-Datasets-using-neural-method/

# 4 Results

## 4.1 Document Ranking Performance

In this section, we compare the performance of four models on the TREC DL Track 2019: two models from HuggingFace (BERT-base and DeBERTa-V3) and two models from Naver Labs Europe (CoCondenser-selfDistil and CoCondenser-EnsembleDistil). We evaluated the models using three IR measures: nDCG@10, MRR, and MAP, with a primary focus on nDCG@10.

For the first three models (BERT-base, DeBERTa-V3, and CoCondenser-EnsembleDistil), we used the same hyperparameters as mentioned in Table A.1 in Appendix A. Among the first three models, DeBERTa-V3 outperformed the others in terms of nDCG@10. However, with the adjusted hyperparameters with slight modifications mentioned in Table A.2, the fourth model (CoCondenser-SelfDistil) achieved the best performance across all measures, demonstrating the effectiveness of proper hyperparameter tuning.

The following table summarizes the performance of the four models across the three IR measures:

**Table 4.1:** Performance comparison of different models on TREC DL Track 2019.

| Model | nDCG@10 | MRR | MAP |
|---|---|---|---|
| **BERT-base** | 0.5807 | 0.8682 | 0.2399 |
| **DeBERTa-V3** | 0.6197 | 0.9050 | 0.2497 |
| **CoCondenser-EnsembleDistil** | 0.5924 | 0.8709 | 0.2398 |
| **CoCondenser-SelfDistil** | 0.6207 | 0.8992 | 0.2498 |

## 4.2 Analysis

In this section, we analyze the performance of the four models: BERT-base, DeBERTa-V3, CoCondenser-SelfDistil, and CoCondenser-EnsembleDistil, in the context of the TREC DL Track 2019. Our focus is on understanding the impact of model architecture and hyperparameter settings on the evaluation metrics: nDCG@10, MRR, and MAP.

### 4.2.1 Model Performance Overview

**BERT-base:** As a well-established baseline, BERT-base achieved reasonable performance across all metrics. However, it was outperformed by the other models, indicating that more recent advancements in model architecture and training strategies offer significant improvements.

**DeBERTa-V3:** DeBERTa-V3 demonstrated superior performance among the models trained with the same hyperparameters in Table A.1. This model's advanced architecture, featuring disentangled attention and enhanced mask decoding, likely contributed to its higher nDCG@10, MRR, and MAP scores compared to BERT-base and CoCondenser-selfDistil.

**CoCondenser-EnsembleDistil:** This model also showed competitive performance, outperforming BERT-base but slightly trailing DeBERTa-V3 in the same hyperparameter setting. The ensemble approach, thorough distillation process and improved negative sampling contributed to its effectiveness, yet there was room for further optimization.

**CoCondenser-SelfDistil:** After adjusting the hyperparameters, CoCondenser-SelfDistil emerged as the best-performing model. The self-distillation technique and improved negative sampling, coupled with optimized training settings, led to significant gains in all metrics. This highlights the critical role of hyperparameter tuning in maximizing model performance.

### 4.2.2 Hyperparameter Influence

**Save Steps:** Increasing the `save_steps` to 6000 for the fourth model (CoCondenser-SelfDistil) allowed for less frequent saving of checkpoints. This adjustment potentially led to more stable and extended training intervals. The save steps for the first three models were set at 2000, ensuring a baseline for comparison.

**Batch Size:** The `per_device_train_batch_size` was increased to 8 for the fourth model helped achieve better gradient estimates per training step, thus enhancing the learning process. The first three models had a batch size of 1, providing a clear contrast in the impact of this parameter change.

**Gradient Accumulation Steps:** The `gradient_accumulation_steps` was increased to 2 for the fourth model, compared to 1 for the first three models. This change allowed for the accumulation of gradients over more steps, which can lead to improved stability and performance during training.

**Evaluation Batch Size:** Reducing the `per_device_eval_batch_size` to 8 for the fourth model, in contrast to 64 for the first three models, may have contributed to

more efficient evaluation, balancing computational load and performance accuracy.
**Single GPU Utilization:** Consistently training all models on a single NVIDIA RTX A5000 GPU ensured uniform use of computational resources, thereby avoiding discrepancies due to varying hardware. This setup facilitated a fair comparison of the models' performance under different hyperparameter settings.

### 4.2.3 Key Observations

**Architectural Improvements:** Advanced model architectures like DeBERTa-V3 and CoCondenser-SelfDistil offer substantial improvements over older models such as BERT-base. Innovations in attention mechanisms and distillation processes contribute significantly to these gains.
**Hyperparameter Optimization:** The performance of CoCondenser-SelfDistil underscores the importance of fine-tuning hyperparameters. By carefully adjusting parameters like batch size and checkpoint saving frequency, significant improvements in model performance can be achieved.
**Effectiveness of Distillation:** The use of distillation techniques, particularly in CoCondenser-SelfDistil, proves to be highly effective. This method helps in transferring knowledge from more complex teacher models, resulting in better performance for the student models.

# 5 Conclusion

In this study, we evaluated the performance of four neural IR models—BERT-base, DeBERTa-V3, CoCondenser-selfDistil, and CoCondenser-EnsembleDistil—on the TREC DL Track 2019. Our experiments demonstrated that modern architectures and advanced training techniques significantly enhance document ranking performance. DeBERTa-V3, with its disentangled attention and enhanced mask decoding, outperformed BERT-base and CoCondenser-EnsembleDistil under the same hyperparameters. However, CoCondenser-SelfDistil, after fine-tuning hyperparameters, achieved the best results across all evaluation metrics. This underscores the importance of both architectural innovations and hyperparameter optimization in developing effective IR models. Our findings highlight the potential for further improvements in neural IR models through continued advancements in model design and training methodologies.

# Bibliography


[1] In: (web). URL: https://en.wikipedia.org/wiki/Text_Retrieval_Conference.

[2] In: (web). URL: https://trec.nist.gov/tracks.html.

[3] In: (web). URL: https://microsoft.github.io/msmarco/TREC-Deep-Learning.

[4] In: (web). URL: http://boston.lti.cs.cmu.edu/appendices/TheWebConf2020-Zhuyun-Dai/.

[5] In: (web). URL: https://huggingface.co/naver.

[6] M. Abualsaud, N. Ghelani, H. Zhang, M. D. Smucker, G. V. Cormack, and M. R. Grossman. "A System for Efficient High-Recall Retrieval". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM. 2018, pp. 1317–1320.

[7] N. Arabzadeh, A. Vtyurina, X. Yan, and C. L. A. Clarke. *Shallow pooling for sparse labels*. 2022. arXiv: 2109.00062 [cs.IR].

[8] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. "A simple framework for contrastive learning of visual representations". In: *arXiv preprint arXiv:2002.05709* (2020).

[9] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. "Electra: Pre-training text encoders as discriminators rather than generators". In: *arXiv preprint arXiv:2003.10555* (2020).

[10] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees. "Overview of The TREC 2019 Deep Learning Track". In: *arXiv preprint arXiv:2003.07820* (2020).

[11] W. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*. Vol. 520. Addison-Wesley Reading, 2010.

[12] Z. Dai and J. Callan. "Context-aware term weighting for first stage passage retrieval". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020.

[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*. 2019.

[14] Z. Dai and J. P. Callan. "Context-aware document term weighting for ad-hoc search". In: *Proceedings of The Web Conference 2020*. 2020.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[16] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant. "From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 2353–2359. ISBN: 9781450387323. DOI: 10.1145/3477495.3531857. URL: https://doi.org/10.1145/3477495.3531857.

[17] T. Formal, B. Piwowarski, and S. Clinchant. *SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking*. 2021. arXiv: 2107.05720 [cs.IR].

[18] L. Gao, Z. Dai, and J. Callan. *Rethink Training of BERT Rerankers in Multi-Stage Retrieval Pipeline*. 2021. arXiv: 2101.08751 [cs.IR].

[19] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality reduction by learning an invariant mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[20] P. He, J. Gao, and W. Chen. *DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing*. 2021. arXiv: 2111.09543 [cs.CL].

[21] P. He, X. Liu, J. Gao, and W. Chen. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. 2021. arXiv: 2006.03654 [cs.CL].

[22] S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury. *Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation*. 2021. arXiv: 2010.02666 [cs.IR].

[23] K. Järvelin and J. Kekäläinen. "IR evaluation methods for retrieving highly relevant documents". In: New York, NY, USA: Association for Computing Machinery, 2000. ISBN: 1581132263. DOI: 10.1145/345508.345545. URL: https://doi.org/10.1145/345508.345545.

[24] O. Khattab and M. Zaharia. *ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT*. 2020. arXiv: 2004.12832 [cs.IR].

[25] Y. Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1746–1751.

[26] J. Lin, R. Nogueira, and A. Yates. *Pretrained transformers for text ranking: Bert and beyond*. Vol. 14. Synthesis Lectures on Human Language Technologies. 2021, pp. 1–325.

[27] Y. Liu. "Recurrent Convolutional Neural Networks for Text Classification". In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 2016, pp. 2397–2403.

[28] R. Nogueira and K. Cho. "Passage re-ranking with bert". In: *arXiv preprint arXiv:1901.04085* (2019).

[29] R. Nogueira, W. Yang, J. Lin, and K. Cho. "Document expansion by query prediction". In: *arXiv preprint arXiv:1904.08375* (2019).

[30] A. van den Oord, Y. Li, and O. Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019. arXiv: 1807.03748 [cs.LG].

[31] B. Paria, C.-K. Yeh, I. E. H. Yen, N. Xu, P. Ravikumar, and B. Póczos. *Minimizing FLOPs to Learn Efficient Sparse Representations*. 2020. arXiv: 2004.05665 [cs.LG].

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[33] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. "Deep contextualized word representations". In: *arXiv preprint arXiv:1802.05365* (2018).

[34] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. *Language Models are Unsupervised Multitask Learners*. OpenAI. 2019.

[35] B. R. R. D. W. W. A. N. L. D. A. Simoni. *Economic Impact Assessment of NIST's Text REtrieval Conference (TREC) Program*. Tech. rep. RTI International, 2010.

[36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6000–6010.

[37] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. "Huggingface's transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771* (2019).

[38] Z. Wu, Y. Xiong, S. Yu, and D. Lin. "Unsupervised feature learning via non-parametric instance discrimination". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE. 2018, pp. 3733–3742.

# A : Training Setup

This appendix details the training setup used for the experiments. We utilized a single NVIDIA RTX A5000 GPU with the following command line setup:

Command Line Setup

```
torchrun \
--nproc_per_node 1 run_marco.py \
--output_dir /path/to/checkpoints \
--model_name_or_path <model_used> \
--do_train \
--save_steps 2000 \
--train_dir /path/to/train_data \
--max_len 512 \
--fp16 \
--per_device_train_batch_size 1 \
--train_group_size 8 \
--gradient_accumulation_steps 1 \
--per_device_eval_batch_size 64 \
--warmup_ratio 0.1 \
--weight_decay 0.01 \
--learning_rate 1e-5 \
--num_train_epochs 2 \
--overwrite_output_dir \
--dataloader_num_workers 8
```

The training configuration includes the hyperparameters mentioned in Table A.1. For the fourth model (CoCondenser-SelfDistil), we made some slight modifications to the hyperparameters mentioned in Table A.2.

**Table A.1:** Hyperparameters for Training on TREC DL 2019 Track Datasets

| **Hyperparameter** | **Value** | **Explanation** |
|---|---|---|
| `nproc_per_node` | 1 | Number of processes to run per node. Set to 1 for single-GPU training. |
| `save_steps` | 2000 | Save a checkpoint every 2000 steps. |
| `max_len` | 512 | Maximum sequence length for inputs. |
| `fp16` | True | Use mixed precision training (fp16) to reduce memory usage and speed up training. |
| `per_device_train_batch_size` | 1 | Number of training samples per batch per device. |
| `train_group_size` | 8 | This parameter is specific to certain training scripts; often not used in standard BERT training. |
| `gradient_accumulation_steps` | 1 | Number of steps to accumulate gradients before updating the model parameters. |
| `per_device_eval_batch_size` | 64 | Number of evaluation samples per batch per device. |
| `warmup_ratio` | 0.1 | Ratio of steps to perform linear learning rate warmup to the total number of training steps. |
| `weight_decay` | 0.01 | Weight decay to apply for regularization. |
| `learning_rate` | 1e-5 | Initial learning rate for training. |
| `num_train_epochs` | 2 | Number of epochs to train the model. |
| `dataloader_num_workers` | 8 | Number of subprocesses to use for data loading. More workers can speed up data loading but use more CPU resources. |

**Table A.2:** Hyperparameter changes for the fourth model (CoCondenser-SelfDistil).

| **Hyperparameter** | **First Three Models** | **Fourth Model** |
|---|---|---|
| **save_steps** | 2000 | 6000 |
| **per_device_train_batch_size** | 1 | 8 |
| **gradient_accumulation_steps** | 1 | 2 |
| **per_device_eval_batch_size** | 64 | 8 |

# B : Judgement Process

In official evaluations, depth pooling is employed to create distinct pools for passage ranking and document ranking tasks. NIST assessors label the passages and documents in these pools using multi-graded judgments, facilitating the measurement of the NDCG metric. The results are based on 200 test queries, with NIST initially selecting 52 of these queries for both the passage ranking and document ranking tasks. These topics were chosen based on the behavior of submitted Document Ranking task runs on the entire test set, evaluated using the sparse MARCO judgments. Test questions with median MRR scores greater than 0.0 but no more than 0.5 were considered for judgment.

An additional 43 topics were selected through a four-step process:

1. For each question, a top-10 pool was created across all runs in the task, including any document with a judgment in the MARCO sparse judgments, resulting in a set of size $P$, which varies by topic. Assessors first judged these pool documents, followed by another 100 documents chosen using the University of Waterloo's HiCAL system [6]. HiCAL builds a relevance model from the current set of judgments and selects the next document most likely to be relevant for judgment. At the end of this stage, there are $R$ known relevant documents. If $2R < P$, judging is finished for the topic.

2. The difference between the number of judged documents and the desired number of $2R+100$ judgments is labeled $G$. An additional $G$ documents are judged using HiCAL, resulting in a total of $J = P + 100 + G$ judgments for the topic, and a new count of relevant documents, $R^*$. If $2R^* + 100 < J$, assessment is finished for the topic. If $R^* \approx J$, the topic is discarded due to the high cost of obtaining "sufficiently complete" judgments.

3. If the topic remains viable, a new increment proportional to the number of known relevant documents is added to the topic budget. This process iterates until the number of known relevant documents is less than half the number of judged documents.

4. The process terminates when assessors run out of time or have no documents left to judge.

The evaluation set included topics with at least three relevant documents and a ratio of $R^*/J < 0.6$. This method resulted in 43 topics for both the Document Ranking and Passage Ranking tasks, with a slightly different set of 43 topics for each task.