BROWN

# Improving Information Retrieval through Contextual Ranking with Large Language Models

PhD Thesis

By

**George Zerveas**

Advisor:
Prof. Carsten Eickhoff

Committee:
Prof. Ellie Pavlick, Prof. Stephen Bach

Department of Computer Science
Brown University

Providence, Rhode Island
February 2024

This dissertation by Georgios Zerveas is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date ______________ ____________________________________

Dr. Carsten Eickhoff, Advisor

Recommended to the Graduate Council

Date ______________ ____________________________________

Dr. Ellie Pavlick, Reader

Date ______________ ____________________________________

Dr. Stephen Bach, Reader

Approved by the Graduate Council

Date ______________ ____________________________________

Dr. Thomas A. Lewis, Dean of the Graduate School

# Abstract

Large pre-trained transformer-based language models (PTLMs) have recently dominated the state-of-the-art in Information Retrieval tasks such as web search and question answering. Despite the advantages such models offer with respect to utilizing *term context* for building better query and document representations, their large number of parameters and computational complexity introduce important constraints and challenges. Thus, training such models for retrieval typically relies on point-wise similarity learning or pair-wise contrastive learning. However, such training settings discard inter-document information and deviate from the actual target objective of search, i.e., comparing a query against a very large collection of documents. Essentially, they overlook the effect of *ranking context*: that is, the benefit derived from jointly assessing the relevance of a large enough set of candidates in meaningful relationship to the same query and therefore to one-another. Ranking context has been found to be beneficial in earlier Leaning-to-Rank approaches that employ non-PTLM neural networks on top of handcrafted features.

In this work, we first investigate the effect of ranking context and its constituent parts: (1) jointly scoring a large number of candidates, (2) using retrieved (query-specific) instead of random negatives, and (3) a fully list-wise loss. To this end, we introduce *COntextual Document Embedding Reranking (CODER)*, a highly efficient and generic fine-tuning framework that for the first time enables incorporating context into transformer-based language models used in state-of-the-art dense retrieval. CODER acts as a lightweight performance enhancing framework that can be applied to virtually any existing dual-encoder model, and used both for standalone single-stage retrieval, as well as for reranking.

We next explore the potential that CODER offers in directly optimizing retrieval for essentially context-dependent, list-wise properties, such as ranking fairness. We find that, compared to the existing alternatives for deep neural retrieval architectures, our end-to-end differentiable and efficient approach based on CODER can attain much stronger bias mitigation (fairness). At the same time, for the same amount of bias mitigation, it offers significantly better relevance performance (utility). Crucially, our method allows for a more finely controllable and predictable intensity of bias mitigation.

Lastly, we seek to enhance the ranking context itself by addressing the problem of sparse relevance annotation in modern large-scale retrieval datasets. To mitigate penalizing the model in case of false negatives during training, we propose evidence-based label smoothing, i.e., propagating relevance from the ground-truth documents to unlabeled documents that are intimately related to them. To that end, we leverage the concept of reciprocal neighbors, moving beyond geometric similarity and exploiting local connectivity in the shared representation space. We find that using the CODER framework to fine-tune retrievers based on the recomputed "smooth" labels substantially improves ranking effectiveness.

# Acknowledgments

The pursuit of a doctorate is a journey of exploration into the land of the unknown. Often, as these journeys tend to be, it is arduous; replete of difficulties, frustration, and self-doubt. Sometimes, like rare, brilliant days of sunshine that transiently dispel overcast New England winter skies, one also gets to experience the thrill of discovery; the excitement of having ventured where no one has gone before.

Thankfully, I was far from alone on this journey. There have been companions ready to help me chart my course, fill my sails with wind, and be my anchor and haven when the nights were too cloudy to discern the North star. I would be remiss if I didn't acknowledge their support with deep gratitude.

I will start by thanking the one who trusted me with a ship and let me take hold of the helm: my advisor, Carsten Eickhoff. His calmness and belief in my abilities have been a constant, reassuring influence; together with his multifaceted support, they provided a steady frame for my work. We both started out as novices in our respective roles, and it has been a fun and rewarding experience exploring uncharted waters together.

Next, I would like to sincerely thank my close collaborators: Navid Rekabsaz, who generously contributed ideas, time and effort, and whose help was instrumental to my work. Dan Cohen, who likewise significantly contributed to shaping this work.

I am very grateful to Ellie Pavlick and Stephen Bach, my committee members, who over the years asked thought-provoking questions, and who carved out time in their busy schedules to meet and discuss ideas and experiments. Several analyses contained in this work originated from their suggestions.

I also want to thank my dear friend Ali Bahranian – it has been a real pleasure working with him, and an even bigger pleasure hanging out together. Also, Aaron Eisman, Ilkay Yildiz, Isaac Sears Adeel Abbasi and Xinrui Lyu, who offered me collaborations that were both interesting and fruitful in terms of knowledge and experience.

Special thanks go to Lauren Clarke, who, with her promptness and enduring willingness to help, has made navigating the treacherous waters of university bureaucracy a breeze.

A university department is only as good as its students. I would like to thank my fellow deckhands and good friends, Reza Esfandiarpoor and Ruochen Zhang, who kindly tolerated my humor and various quirks when working and hanging out together. Along with Wasiwasi Mgonzo, Catherine Chen, Michal Golovanevsky, Amina Abdullahi, William Rudman, Jack Merullo, Yong Zheng Xi, Peilin Yu, Nihal Nayak, Abdelrahman Hosny, Aaron Traylor, Saket Tiwari, Apoorv Khandelwal, Alessio Mazzetto, Jason Liu, Charlie Lovering, Calvin Luo and many others, they offered me a trove of fond memories, and stimulating and silly discussions in equal measure, which made my PhD experience so much more rewarding, exciting and fun.

Words fail me when trying to express how deeply grateful I am to my loving wife, Heeju. She has been an endless source of strength and joy, and an important pillar in my life. And last, but certainly not least, my profound gratitude goes to my parents and family in Greece, whose warm, loving presence I have always felt embracing me, despite thousands of miles of ocean separating us.

*Dedicated to the struggling people of Palestine, and all people around the world who, regardless of how much they have lost, refuse to lose their humanity.*

# Contents

**1 Introduction** **2**

1.1 Transformer-based Language Models for Information Retrieval . . . . . . . . 2

1.1.1 A new paradigm . . . . . . . . . . . . . . . . . . . . . . . . . 2

1.1.2 Limitations of current methods and the problem of ranking context . . 3

1.2 The contributions of this work . . . . . . . . . . . . . . . . . . . . . . . 4

1.2.1 A method for training PTLMs through contextual similarity learning . 4

1.2.2 Leveraging contextual similarity learning for bias mitigation . . . . . 4

1.2.3 Enhancing the ranking context of dense retrieval through Reciprocal Nearest Neighbors . . . . . . . . . . . . . . . . . . . . . . . . . 4

**2 An efficient framework for improving retrieval through contextual reranking of document embeddings** **9**

2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.1.1 Limitations of contemporary training methods for retrieval models . . 9

2.1.2 Our contribution . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.1.3 Motivation for simultaneously scoring a large number of negative documents . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Related Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.3 Method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.3.1 Architecture . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

2.3.2 Training through CODER . . . . . . . . . . . . . . . . . . . . . . 15

2.3.3 Loss function . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.4 Experimental Setup . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

2.5 Results and discussion . . . . . . . . . . . . . . . . . . . . . . . . . . 19

2.5.1 Results on MS MARCO and TREC DL . . . . . . . . . . . . . . . 19

2.5.2 Results on TripClick . . . . . . . . . . . . . . . . . . . . . . . . . 20

2.5.3 Efficiency . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

2.5.4 Analysis of key factors . . . . . . . . . . . . . . . . . . . . . . . 21

2.5.5 How context can help even without parametric modeling of document relationships . . . . . . . . . . . . . . . . . . . . . . . . . . 23

2.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24

**3 Mitigating bias in search results through contextual document reranking and neutrality regularization 32**
3.1 Introduction . . . 32
3.2 Method . . . 34
3.3 Experiments . . . 36
3.3.1 Resources . . . 36
3.3.2 Retrieval Models . . . 36
3.3.3 Evaluation of Bias Mitigation and Utility . . . 37
3.3.4 Training, Model Selection and Hyperparameters . . . 37
3.4 Results . . . 38
3.5 Conclusion . . . 39

**4 Enhancing the ranking context of dense retrieval through Reciprocal Nearest Neighbors 47**
4.1 Introduction . . . 47
4.1.1 Addressing the problem of sparse annotation . . . 47
4.1.2 Limitations of contemporary dense retrieval . . . 48
4.1.3 The contribution of this work . . . 49
4.2 Related work . . . 50
4.3 Methods . . . 52
4.3.1 Similarity metric based on Reciprocal Nearest Neighbors . . . 52
4.3.2 Evidence-based label smoothing . . . 54
4.3.3 Computational efficiency . . . 57
4.4 Experimental setting . . . 58
4.5 Results and Discussion . . . 59
4.5.1 Inference-time reranking with reciprocal nearest neighbors . . . 59
4.5.2 Evidence-based label smoothing . . . 62
4.6 Limitations and applicability to other frameworks . . . 68
4.7 Conclusion . . . 69

**5 Conclusion and outlook 76**
5.1 A new framework for contextual ranking and similarity learning for Large Language Models . . . 76
5.2 (Very) Large Language Models for in-context ranking . . . 77
5.3 Remaining challenges and future directions . . . 79
5.3.1 Expanding the generality of the contextual similarity learning framework 79
5.3.2 Extending the versatility of the bias mitigation approach . . . 80

**A General 83**
A.1 Data . . . 83
A.1.1 MS MARCO and TREC Deep Learning . . . 83
A.1.2 TripClick . . . 83
A.2 Evaluation principles . . . 84
A.3 Social impact considerations . . . 84

**B Supplementary material** **85**
B.1 Using a transformer as a contextual document scoring function . . . . . . . . 85
B.2 Inference-time reranking with reciprocal nearest neighbors . . . . . . . . . . 89
B.3 Evidence-based label smoothing . . . . . . . . . . . . . . . . . . . . . . . . 93

# List of Figures

2.1 Schematic diagram: effect of multiple negatives and dimensionality on loss . 11
2.2 Schematic diagram of the CODER method . . . . . . . . . . . . . . . . . . . 14
2.3 Histogram of ranks of relevant documents when training with CODER . . . . 19
2.4 Performance of BM25→CODER(RepBERT) on MS MARCO validation set over training steps . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25

3.1 Schematic diagram of CODER (2nd version) . . . . . . . . . . . . . . . . . 33
3.2 Utility (MRR@10) versus fairness (NFaiRR@10) . . . . . . . . . . . . . . . 36
3.3 Modulation of utility and fairness . . . . . . . . . . . . . . . . . . . . . . . 38
3.4 Evolution of utility and fairness during training . . . . . . . . . . . . . . . . 40

4.1 Schematic diagram: effect of reciprocal nearest neighbors . . . . . . . . . . . 48
4.2 Similarity scores per rank across a large number of queries. . . . . . . . . . . 55
4.3 Similarity scores of the top 1000 candidates for a single query . . . . . . . . 56
4.4 Latency per query for reranking with Recip. NN . . . . . . . . . . . . . . . . 58
4.5 Performance of rNN reranking of CODER(TAS-B) on MS MARCO dev.small as a function of the number of in-context candidates . . . . . . . . . . . . . . 61
4.6 Evolution of performance when training CODER(TAS-B) with label smoothing on MS MARCO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
4.7 Evolution of performance when training on TripClick with EB label smoothing 68
4.8 Disparity between MRR and KL-divergence . . . . . . . . . . . . . . . . . . 71

B.1 Schematic diagram of CODER employing a transformer “decoder” as a document set scoring module. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
B.2 Evolution of the loss on the training set, for a different number and type of candidates, as training of BM25→CODER(RepBERT) progresses . . . . . . 87
B.3 Evolution of the loss on the training set, for different number of candidates, as training of BM25→CODER(RepBERT) progresses . . . . . . . . . . . . . . 88
B.4 Performance of rNN reranking of CODER(TAS-B) on MS MARCO dev as a function of the number of in-context candidates . . . . . . . . . . . . . . . . 90
B.5 Performance of rNN reranking of CODER(TAS-B) on TREC DL 2020 as a function of the number of in-context candidates . . . . . . . . . . . . . . . . 90
B.6 Performance of rNN reranking of CODER(TAS-B) on TREC DL 2019 as a function of the number of in-context candidates . . . . . . . . . . . . . . . . 91
B.7 Performance of rNN reranking of TAS-B on MS MARCO dev as a function of the number of in-context candidates . . . . . . . . . . . . . . . . . . . . . 92

B.8 Performance of rNN reranking of CODER(RepBERT) on TripClick as a function of the number of in-context candidates . . . . . . . . . . . . . . . . . 92

# List of Tables

2.1 Main configuration parameters of CODER (without transformation of document representations). . . . . 17
2.2 Performance for passage ranking when applying CODER training to RepBERT and TAS-B . . . . . 18
2.3 Performance when applying CODER to RepBERT on the TripClick HEAD dataset . . . . . 20
2.4 Performance when applying CODER to RepBERT on the TripClick dataset . 23
2.5 Zero-shot results: all models are trained on MS MARCO but evaluated on TripClick . . . . . 23

3.1 Comparison of ranking performance based on top 4 F1 scores within the NFaiRR range [0.9 - 0.93]. . . . . . 39

4.1 Hyperparameters for reranking with Reciprocal Nearest Neighbors, MS MARCO 59
4.2 Recip. NN reranking, MS MARCO collection . . . . . 60
4.3 Recip. NN reranking, TripClick Test . . . . . 61
4.4 Label smoothing applied to training CODER(TAS-B) on MS MARCO . . . . 62
4.5 Example: documents receiving relevance probability through EB label smoothing 63
4.6 Hyperparameters for training with evidence-based label smoothing, MS MARCO 65
4.7 Example: ranking with model trained through label smoothing . . . . . . . . 66
4.8 Evaluation of evidence-based label smoothing (mixed rNN - geom. similarity) on TripClick . . . . . 67
4.9 EB label smoothing applied to training on TripClick . . . . . 69

5.1 Ranking effectiveness on TREC DL 2019, 2020 . . . . . 78

B.1 Performance of models trained on MS MARCO but zeroshot-evaluated on TripClick Test . . . . . 93
B.2 Performance of models trained on MS MARCO but zeroshot-evaluated on TripClick Val . . . . . 93
B.3 Example: documents receiving relevance probability through EB label smoothing 94
B.4 Example: documents receiving relevance probability through EB label smoothing 95
B.5 Example: ranking with model trained through label smoothing . . . . . . . . 96

# Chapter 1

# Introduction

## 1.1 Transformer-based Language Models for Information Retrieval

### 1.1.1 A new paradigm

Deep learning has already been successfully used for about a decade now in information retrieval (IR) [1]. However, the pivotal breakthrough that has resulted in unprecedented progress in the field was the advent of transformers, and especially of large pre-trained transformer-based language models (PTLMs). In fact, such models currently dominate the state-of-the-art in IR tasks such as web search and question answering [2, 3, 4, 5, 6], far outperforming established algorithmic methods based on lexical overlap features [7], as well as various machine learning approaches leveraging lexical features and their interactions (e.g. [8, 9]).

By extracting contextualized term representations both from the query as well as the target documents, the transformer architecture offers unique advantages with respect to capturing the user's intent from the query, and the salient topics in target documents. It thus enables building informative vector representations of queries and documents, optimally embedded in a common vector space. This is of critical importance for the effectiveness of *dense retrieval* methods, whose function relies on computing a simple similarity/distance metric between query and document vectors, and then retrieving the nearest neighbors with respect to this metric (e.g. [10]). Given the vector representations, the latter function can be efficiently performed during inference time using optimized libraries for retrieving Approximate Nearest Neighbors (e.g. [11]) and GPUs.

As a second approach, in so-called *cross-encoder* models (e.g. [12]), the tokens of the query and candidate documents, separated by special tokens, are concatenated within the same input context window, and the self-attention mechanism of the transformer encoder enables rich interactions between query and document term representations on multiple abstraction levels. As such, these models can typically achieve higher ranking effectiveness, but due to the $O(N^2)$ computational cost of self-attention with respect to the length of the context window $N$, they are forbiddingly computationally expensive: the need to concatenate queries with candidate documents leads both to slower processing times per candidate document, as well as separate evaluation of all query-candidate combinations. They are therefore inevitably used only for reranking the top-$k$ results retrieved for each query by first-stage methods, such as dense retrieval methods, with $k$ separate (but potentially concurrent, to an extent depending on batch size) evaluations *per query*.

### 1.1.2 Limitations of current methods and the problem of ranking context

Despite the opportunities PTLM models offer with respect to utilizing *term context*, their large number of parameters and computational complexity introduce important constraints and challenges. In particular, when training such models for information retrieval, current approaches typically rely either on (a) point-wise similarity learning - used especially for cross-encoders models, e.g. [12] - where each document is scored with respect to its similarity to a query in isolation, as described above, or (b) pair-wise contrastive learning - used practically in all contemporary dense retrieval methods, e.g. [10] - where the model is asked to score the similarity between the query and a ground truth relevant document (termed *positive*) more highly than the similarity between the query and a *negative* document. However, such training settings deviate from the actual end-target objective of search, i.e., simultaneously comparing a query against a large collection of documents, and additionally discard inter-document information. Essentially, they overlook the effect of *ranking context*, which has been previously found to be very beneficial in Leaning-to-Rank approaches that employ non-PTLM neural networks on top of handcrafted features [13, 14]. By ranking context we mean a large enough set of documents that are in meaningful relationship to the same query (and by extension to one-another) and are *jointly* evaluated with respect to their relevance to this query.

Interestingly, recent works have partially (and unwittingly) rediscovered aspects of ranking context as individual techniques and best practices, such as mining “hard negatives” (e.g. [10, 15, 16]) or including a large number of in-batch random negatives through the use of huge batch sizes and multiple GPU (e.g. [17]). The most effective state-of-the-art pipelines come with a very high computational cost, integrating the above techniques alongside others such as heavy-weight cross-encoder PTLMs for filtering negatives, distillation and and pseudo-labeling of additional collection documents (e.g. [2]).

Such techniques employed in contemporary dense retrieval literature can in fact be be seen as individual steps towards procuring and sanitizing a ranking context for training dense retrieval models, albeit never recognized as such. The fact that current PTLM-based retrieval research (which traces its origins in generic Natural Language Processing rather than Learning-to-Rank Information Retrieval) is oblivious to the concept of ranking context is exemplified by the fact that no method uses more than a handful of mined hard negatives, with some of the most influential works in the field advocating that no more than 4 hard negatives should be used based on empirical findings [10], and other works arguing that this negative impact of a larger number of hard negatives may be attributed to false negatives and thus proposing the filtering of negatives as essential [17].

## 1.2 The contributions of this work

### 1.2.1 A method for training PTLMs through contextual similarity learning

After identifying *ranking context* as an important component of training dense retrieval models, in Chapter 2 we expressly investigate its effect and what we consider its constituent parts: (1) jointly scoring a large number of candidates, (2) using retrieved (query-specific) instead of random negatives, and (3) a fully list-wise loss. To this end, we introduce *COntextual Document Embedding Reranking (CODER)*, a highly efficient and generic fine-tuning framework that for the first time enables effectively incorporating context into transformer-based language models used in state-of-the-art dense retrieval. By relying on precomputed, fixed document representations and fine-tuning only the query encoder, CODER acts as a lightweight performance enhancing framework that can be applied to virtually any existing dual-encoder model, and used both for standalone single-stage retrieval, as well as for reranking. More than a concrete training method, our work acts as a set of specifications for effective similarity learning with PTLMs and draws the connection between existing best practices to the concept of ranking context.

### 1.2.2 Leveraging contextual similarity learning for bias mitigation

Having introduced a methodology for contextual similarity learning, in Chapter 3 we explore the potential it inherently offers for directly optimizing retrieval with respect to essentially context-dependent, list-wise properties, such as ranking fairness. Compared to the existing alternatives for deep neural retrieval architectures, which are based on adversarial training, we find that our end-to-end differentiable and efficient approach based on CODER can attain much stronger bias mitigation (fairness). At the same time, for the same amount of bias mitigation, it offers significantly better relevance performance (utility). Moreover, our method allows for the first time a finely controllable and predictable intensity of bias mitigation, which is essential for practical deployment in production systems.

### 1.2.3 Enhancing the ranking context of dense retrieval through Reciprocal Nearest Neighbors

Finally, in Chapter 4 we propose novel methods that can enhance ranking context and potentially leverage it more effectively by exploiting relationships between the candidate documents and the query that extend beyond simple geometric proximity. We accomplish this by addressing the problem of sparse relevance annotation in modern large-scale retrieval datasets. To mitigate penalizing the model in case of false negatives during training, we propose evidence-based label smoothing, i.e., propagating relevance from the ground-truth documents to unlabeled documents that are intimately related to them. To that end, we leverage the concept of reciprocal nearest neighbors, taking into account the local density of documents in the representation space, i.e., the degree of connectivity of documents to their surroundings. We find that using reciprocal nearest neighbors provides an additional measure of similarity that is more robust when ranking candidate documents. Importantly, when using this similarity for computing “smooth” labels for documents, we find that the fine-tuning retrievers through the CODER framework offers substantially improved ranking effectiveness.

## Repository

The corresponding code and other resources are available at:
`https://github.com/gzerveas/CODER`

# Bibliography

[1] Bhaskar Mitra and Nick Craswell. An Introduction to Neural Information Retrieval t. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, 2018. ISSN 1554-0669, 1554-0677. doi: 10.1561/1500000061. URL http://www.nowpublishers.com/article/Details/INR-061.

[2] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.224. URL https://aclanthology.org/2021.emnlp-main.224.

[3] Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, and Haifeng Wang. ERNIE-Search: Bridging Cross-Encoder with Dual-Encoder via Self On-the-fly Distillation for Dense Passage Retrieval, May 2022. URL http://arxiv.org/abs/2205.09153. arXiv:2205.09153 [cs].

[4] Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. Adversarial Retriever-Ranker for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/pdf?id=MR7XubKUFB.

[5] Kun Zhou, Yeyun Gong, Xiao Liu, Wayne Xin Zhao, Yelong Shen, Anlei Dong, Jingwen Lu, Rangan Majumder, Ji-rong Wen, and Nan Duan. SimANS: Simple ambiguous negatives sampling for dense text retrieval. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 548–559, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-industry.56.

[6] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.203. URL https://aclanthology.org/2022.acl-long.203.

[7] Giambattista Amati. BM25. In LING LIU and M. TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 257–260. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_921. URL https://doi.org/10.1007/978-0-387-39940-9_921.

[8] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to Match using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299, Republic and Canton of Geneva, CHE, April 2017. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4913-0. doi: 10.1145/3038912.3052579. URL https://doi.org/10.1145/3038912.3052579.

[9] Bhaskar Mitra, Sebastian Hofstatter, Hamed Zamani, and Nick Craswell. Conformer-Kernel with Query Term Independence for Document Retrieval, July 2020. URL http://arxiv.org/abs/2007.10434. arXiv:2007.10434 [cs].

[10] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

[11] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *arXiv:1702.08734 [cs]*, February 2017. URL http://arxiv.org/abs/1702.08734. arXiv: 1702.08734.

[12] Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *arXiv:1901.04085 [cs]*, April 2020. URL http://arxiv.org/abs/1901.04085. arXiv: 1901.04085.

[13] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 85–92, Santa Clara CA USA, September 2019. ACM. ISBN 978-1-4503-6881-0. doi: 10.1145/3341981.3344218. URL https://dl.acm.org/doi/10.1145/3341981.3344218.

[14] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 499–508, New York, NY, USA, July 2020. Association for Computing Machinery. ISBN 978-1-4503-8016-4. doi: 10.1145/3397271.3401104. URL https://doi.org/10.1145/3397271.3401104.

[15] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive

Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*, October 2020. URL http://arxiv.org/abs/2007.00808. arXiv: 2007.00808.

[16] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada, July 2021. ACM. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462880. URL https://dl.acm.org/doi/10.1145/3404835.3462880.

[17] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.466. URL https://aclanthology.org/2021.naacl-main.466.

# Chapter 2

# An efficient framework for improving retrieval through contextual reranking of document embeddings

## 2.1 Introduction

### 2.1.1 Limitations of contemporary training methods for retrieval models

Neural text retrieval models typically rely on a contrastive training optimization that uses *(query, positive document, negative document)* triplets as training samples. This scheme is especially popular, as it is well-suited to the computational constraints of large transformer-based language models such as BERT [1] and its variants for retrieval [2, 3, 4, 5]. Referred to as pair-wise training, the model is asked to score the similarity between the query embedding and a ground truth relevant (positive) document embedding, higher than the one between the query and a negative document embedding.

While such contrastive learning approaches are effective, by only considering pairs of positive and negative documents at a time, they (1) deviate from the target objective of comparing a query against many documents while discarding inter-document information, and (2) depart from core list-wise evaluation metrics like nDCG [6].

Addressing the first shortcoming, recent works have employed "in-batch" negatives: given a batch containing *(query, positive document)* tuples, the negatives to a tuple are set as the known positive documents from other queries within that batch (e.g. [4, 7, 8, 9, 10]). Although this approach efficiently increases the number of negatives to improve performance, the presence of *hard* negative samples[1] is critical in achieving state-of-the-art (SOTA) [9, 10, 11, 12].

The rich literature on *learning-to-rank (L2R)* has outlined compelling reasons for taking into account the context of other candidate documents being jointly ranked for relevance with respect to the same query when scoring each individual document [13, 14, 15]. This *ranking context* is realized in various list-wise optimizations. Learning-to-rank approaches allow for the model to directly optimize IR metrics as opposed to a surrogate pair-wise

[1]Hard negatives look similar in topic and term distribution to relevant documents, while not actually satisfying the information need.

loss as seen in the contrastive training regime. However, despite achieving competitive results in a variety of ranking situations, considerations of computational complexity and stochastic stability practically relegate them to shallow neural models over handcrafted feature vectors [16, 17, 18].

In any case, contemporary methods employed for training the SOTA retrievers that are based on pre-trained transformer-based language models seem oblivious to the beneficial effects of ranking context and are not designed to leverage it in a systematic fashion as a means to improve ranking effectiveness.

### 2.1.2 Our contribution

In this chapter, we extend existing work in negative sampling and list-wise learning to allow large pre-trained language models to take advantage of the context found in a large, *coherent* set of candidate documents. We particularly examine the effect of constituent parts of the query context, i. e. (1) jointly scoring a large number of negatives, (2) using retrieved (query-specific) instead of random negatives, and (3) a fully list-wise loss. To this end, we introduce *COntextual Document Embedding Reranking (CODER)*, a highly efficient and generic fine-tuning framework that for the first time enables incorporating context, previously only considered in learning-to-rank neural networks, into transformer-based language models used in state-of-the-art dense retrieval. CODER acts as a lightweight performance enhancing framework that operates on precomputed document embeddings, while transforming the query to account for new list-wise context information over a large number of query-specific hard negative candidate documents. It can be applied to virtually any existing dual-encoder model, and used both for single- as well as two-stage dense retrieval.

Our contribution is three-fold:

- We introduce an efficient framework which enables leveraging ranking context.
- We conduct a large set of experiments on the MS MARCO [19] and TripClick [20] collections and show that CODER can considerably enhance the effectiveness of a wide class of dense retrieval models at minimal computational cost, while achieving new SOTA results on TripClick.
- We explore the impact of the constituent parts of ranking context in learning effective models.

### 2.1.3 Motivation for simultaneously scoring a large number of negative documents

In contrastive learning, models are trained to assign a higher score for similarity between the query and a positive document than between the query and all negative documents: $s(q, p_j^+) > s(q, p_i^-), \ \forall i, j$, where $p_j^+$ denotes a positive and $p_i^-$ a negative document/passage respectively. In Figure 2.1, vector representations of documents are depicted as points (red for positive, blue for negative documents) in a $d$-dimensional space that is shared with the query representation vector.

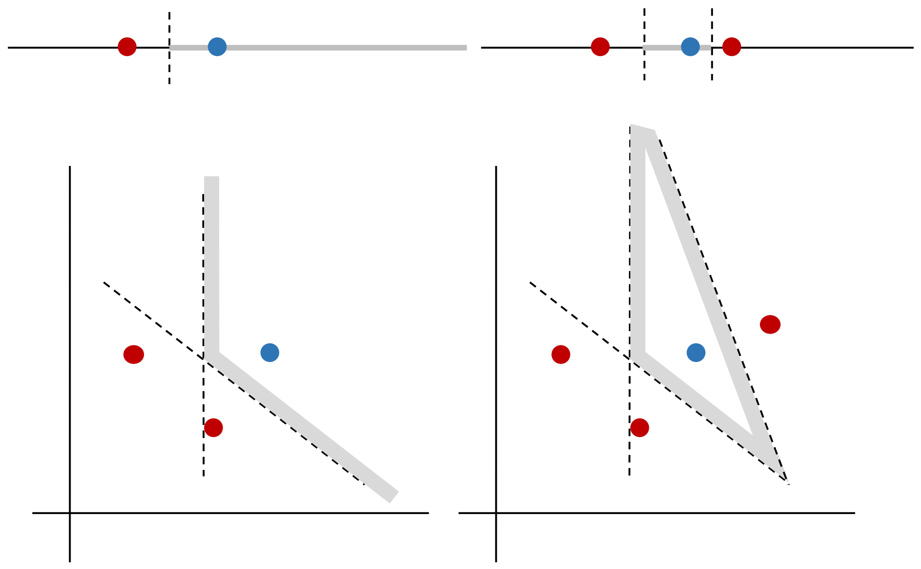


Figure 2.1: Positive (blue) and negative (red) document vectors in a shared document and query embedding space of 1 (upper row) and 2 (lower row) dimensions. The contrastive learning objective simply requires the query representation to be mapped closer to the positive than all negative documents; this means that for a $d$-dimensional space, the query representation need not necessarily lie in proximity to the positive document, but simply within an infinite subspace (left column: grey part of line in 1D space, grey-outlined part of plane in 2D space). At least $d+1$ negative documents are required to constrain the space of favorable query mappings to a bounded convex polytope. Given that the positive document is contained within a simplex formed by $d+1$ negative documents as vertices in that space (interval in $\mathbb{R}$, triangle in $\mathbb{R}^2$, tetrahedron in $\mathbb{R}^3$ etc), the loss function will favor mapping the query onto another simplex in the same space (right column: grey interval in 1D space, grey-outlined triangular area in 2D space).

Because functions used to compute similarity increase with decreasing Euclidean distance, the objective can be fulfilled by learning to map the query within a smaller distance from a given positive document $p_j^+$ compared to all negative documents[2]. However, for a space of dimension $d$, when fewer than $d+1$ negative documents are included in a loss calculation, there is an infinite subspace where the query representation can lie, arbitrarily far from the positive document, and still satisfy this condition.

At least $d+1$ negative documents are required to constrain the space of objective-favored query mappings to a bounded convex polytope: given that the positive document is contained within a simplex formed by $d+1$ negative documents as vertices in that space (interval in $\mathbb{R}$, triangle in $\mathbb{R}^2$, tetrahedron in $\mathbb{R}^3$ etc), the loss function will favor mapping the query onto another simplex in the same space, within which the distance from the positive document will be bounded.

[2]Because document vectors are distributed far from the origin, this will typically be true even when the dot product is used as a similarity function.

Of course, training with fewer than $d+1$ negative documents per query still works: although the loss landscape as revealed by only a few negative documents is a rough approximation, and thus the parameter updates computed by stochastic gradient descent for each batch will be suboptimal and noisy, a good minimum of the loss can still be found in expectation by iterating over the entire training set. However, as the number of negative documents per query increases to $d+1$, the approximations of the gradients of the loss and thus parameter updates at each training step will be more accurate and therefore training will be more efficient.

Increasing the number of negatives to an even higher number is still expected to yield a performance improvement, but at a reduced rate: this is because (a) there is no guarantee that with $d+1$ negatives per query, the positive document will be contained within a simplex of negatives, but with every additional negative this probability increases, and (b) once the positive document is contained within a finite simplex, the probability for every additional negative document to further constraint the bounded subspace where the loss can be minimized becomes increasingly smaller.

The above theoretical analysis can explain the observations made e.g. by Hofstätter et al. [9], who obtained significantly better performance when increasing the batch size from 32 to 256, or by  Qu et al. [10], who used "cross-batch" random negatives (pooled from different GPUs) to effectively increase the number of "in-batch" negatives to a few thousand documents in order to "reduce the discrepancy between training and inference", and noticed a substantial improvement of performance as a result. Also, in agreement to our explanation above, the performance improvement they observed as a function of the number of negatives quickly saturated when exceeding the dimensionality of the document embedding space.

Finally, it is evident from the analysis above that the closer the negative document representations lie to to the ground truth representation (i.e. the more relevant the negatives are), the smaller the bounded convex subspace will be, a fact which supports the observed importance of challenging negatives in literature [9, 10, 11, 12].

## 2.2 Related Work

Recent work has demonstrated the importance of the quality of negative documents used during fine-tuning. Xiong et al. [11] periodically re-encode every query and document in the collection during training in order to mine the most difficult documents to use as negative candidates via approximate nearest neighbor (ANN) search. Improving on this slow and resource-intense process, Zhan et al. [5] (published as Zhan et al. [12]) forego fine-tuning of the document encoder, instead only fine-tuning the query encoder while dynamically mining negatives. TAS-B [9] also improves the quality of negatives by clustering semantically similar queries, such that the in-batch negatives are indirectly related to the ground truth document. While we contribute to this line of research, we show that one can avoid such complexity and directly benefit from list-wise optimization applied on a large, coherent and informative context of candidate documents, retrieved for each query in advance.

Moving from quality to quantity of negative candidates, RocketQA [10] drastically increases the quantity of random in-batch negatives to several thousands by sharing negatives across at least 8 V100 GPU instances. Despite its huge size, this pool of sampled negatives includes only 4 retrieved hard negatives per query and is otherwise almost entirely random, with no shared context across documents. Unlike CODER, this approach necessitates training an expensive cross-encoder model to "denoise" (filter out) retrieved candidates, otherwise yielding poor performance. Moreover, it involves using the cross-encoder to pseudo-label additional data samples, an approach adopted by the currently top performing dual encoder models following up this work, RocketQAv2 [21], which additionally leverages list-wise cross-encoder teaching, and PAIR [22], which includes a loss term capturing similarity between passages.

To address the increasing complexity and computational requirements of training pipelines, Gao and Callan [23, 24] instead propose corpus-specific, self-supervised pre-training with a bespoke transformer add-on (see Baselines in Section 2.4). An orthogonal approach to reduce computational requirements focuses on jointly optimizing query optimizer and product quantization for ANN search [25].

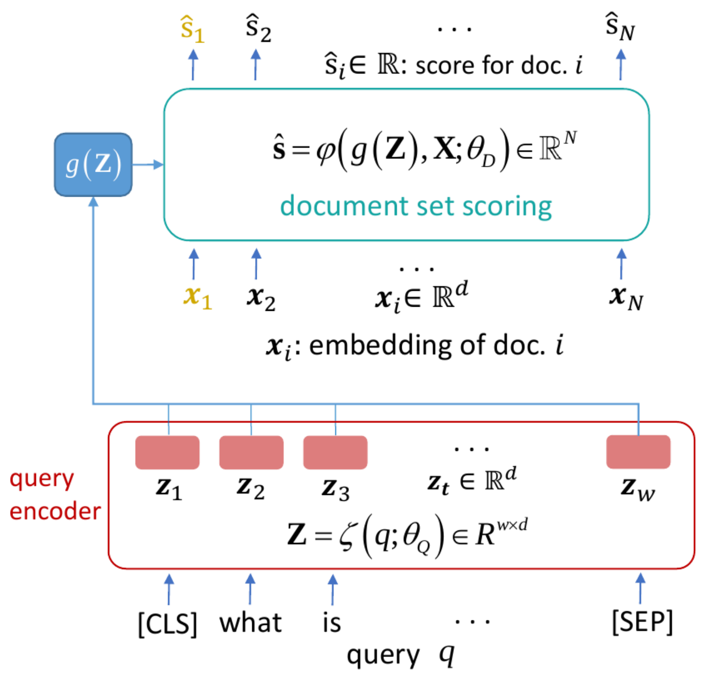
List-wise loss functions have been extensively used within learning-to-rank (L2R), although they have been limited to either shallow neural models [13, 16] or various deep networks Ai et al. [14, 15], Pang et al. [17], Pasumarthi et al. [26] in works focusing on L2R datasets. These consist of handcrafted feature vectors representing query-document similarity such as term overlap, click-through rate, BM25 scores, and other salient features. Effectively applying L2R concepts to transformer-based language models used for ad-hoc dense retrieval is a non-trivial challenge, and represents CODER's extension of prior works.

Finally, there is a body of work utilizing large pre-trained language models for retrieval in cross-encoder [2, 27, 28], late cross-encoder [3, 29], and generative rankers [30], as well as query/document expansion and indexing [31, 32, 33, 34, 35]. Co-BERT [36], a recent method leveraging ranking context, uses a cross-encoder BERT Reranker to select candidates and to compute feature vectors as input for the L2R methods above [17] through query-document term interactions. In comparison, CODER is orders of magnitude more efficient both during training and inference.

All existing approaches either advocate for using a handful of hard negatives (e.g. [7]), or compromise with it due to computational constraints. By recognizing the importance of context, our proposed framework is the first to allow the combination of quantity and quality of negatives for training SOTA dual encoder models with very modest computational resources.

## 2.3 Method

CODER involves fine-tuning a pre-trained query encoder to learn a query representation that is as proximal as possible to the representation of the ground-truth relevant document(s), by adjusting it to better account for the context of multiple query-related documents. The architecture consists of two main components (Fig. 2.2): a query encoder, which builds a query representation, and the document set scoring module, which, given a query representation, jointly scores a set of $N$ precomputed embeddings of positives and hard negatives retrieved by an arbitrary retrieval method, $M_C$. Using precomputed document embeddings reduces computational costs (memory, FLOPs) by a factor of $N$. Thus, unlike all existing approaches, we can afford to use a large number of such hard negatives ($N = 1000$ in our experiments,



$\hat{s}_1$
$\hat{s}_2$
$\hat{s}_N$
$\hat{s}_i \in \mathbb{R}$: score for doc. $i$
$g(\mathbf{Z})$
$\hat{\mathbf{s}} = \varphi(g(\mathbf{Z}), \mathbf{X}; \theta_D) \in \mathbb{R}^N$
document set scoring
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{x}_i \in \mathbb{R}^d$
$\boldsymbol{x}_N$
$\boldsymbol{x}_i$: embedding of doc. $i$
query encoder
$\mathbf{z}_1$
$\mathbf{z}_2$
$\mathbf{z}_3$
$\mathbf{z}_t \in \mathbb{R}^d$
$\mathbf{z}_w$
$\mathbf{Z} = \zeta(q; \theta_Q) \in R^{w \times d}$
[CLS]
what
is
[SEP]
query $q$

Figure 2.2: Schematic diagram of the CODER method. The architecture consists of two main components: the query encoding module, which embeds a tokenized query, and the document set scoring module, which jointly scores a set of $N$ precomputed candidate document embeddings. Here, we experiment specifically with $\varphi = \mathbf{X} \cdot g\,(\mathbf{Z})$, combined with a list-wise loss and large $N = 1000$, which we show allows to effectively leverage ranking context.

unless otherwise noted).

### 2.3.1 Architecture

#### Query Encoder

The query encoder can be any pre-trained transformer encoder, such as BERT [1], DistilBERT [37], RoBERTa [38], or ERNIE [39]. We initialize its weights from an existing model already fine-tuned for retrieval, which we call "*base model*", $M_D$.

Formally, for each query token $q_t$, $t \in \mathbb{N} : 1 \leq t \leq w$, where $w$ is the length of the tokenized query sequence, it extracts a vector representation $\mathbf{z}_t \in \mathbb{R}^d$, where $d$ is the encoder's internal representation dimension. These vectors can be linearly projected to a space of different dimensionality and become $\mathbf{z}'_t \in \mathbb{R}^{d'}$, to match the dimensionality of the document embeddings $d'$, in case the latter differs. In the general case we thus denote the extracted representation of a query $q$ as:

$$\mathbf{Z}' = [\mathbf{z}'_1; \ldots; \mathbf{z}'_w] = \zeta\,(q; \theta_Q) \in \mathbb{R}^{w \times d'}, \tag{2.1}$$

where $\theta_Q$ are the parameters of the query encoder. The individual token representations are aggregated into a single vector using an aggregation function $g$. In our experiments, we let $g(\mathbf{Z}') = \frac{1}{w} \sum_t \mathbf{z}'_t$ be the mean when using RepBERT [4], and $g(\mathbf{Z}') = \mathbf{z}'_1$ (i.e. the representation of the [CLS] token) when using TAS-B [9] as the base model for implementing the query encoder $\zeta$ (see Section 2.4).

**Document scoring function**

The document set scoring function is represented by $\varphi$ which produces a set of $N$ scalar relevance scores $\hat{s}_i \in \mathbb{R}$, $i \in \mathbb{N} : 1 \leq i \leq N$. It takes as an input the aggregated query representation $g(\mathbf{Z}')$ from the previous section, and a set of $N$ document embeddings $\mathbf{x}_i \in \mathbb{R}^m$, $i \in \mathbb{N} : 1 \leq i \leq N$, precomputed by the base model. Using learnable linear projections, the dimensionality of the document embeddings can potentially be changed to accommodate different scoring functions (e.g., transformer blocks with an internal representation dimension $d' \neq m$), while matching the dimensionality of the query embeddings. Succinctly, the output relevance scores are:

$$\hat{\mathbf{s}} = \varphi\left(g(\mathbf{Z}'), \mathbf{X}; \theta_D\right) = \mathbf{X} \cdot g\left(\mathbf{Z}\right) \in \mathbb{R}^N, \tag{2.2}$$

where $\mathbf{X} = [\mathbf{x}_1; \ldots; \mathbf{x}_N] \in \mathbb{R}^{N \times m}$ are the $N$ document embeddings, $\theta_D$ are the parameters of the scoring function, and $d' \equiv d$ (i.e. query and document embeddings have the same dimensionality).

While a variety of functions can be used as a scoring function in our framework, including transformers , for all results presented in this work we leverage the simple inner product, which interestingly achieves significant performance improvements even without the contextualized transformation of document embeddings. It thus appears that jointly scoring a large number of query-specific candidates for the same query within a list-wise loss establishes a strong enough context for improving performance. Beyond computational efficiency, the main advantage of the above function is that it facilitates directly using the fine-tuned query encoder for dense retrieval (single-stage) through fast approximate nearest neighbor search. Instead, a non-linear scoring module would only allow using the model for reranking in a two-stage retrieval setting (candidate retrieval, followed by reranking).

### 2.3.2 Training through CODER

The document representations $\mathbf{X}$ are precomputed using the *document* encoder part of any state-of-the-art dual encoder retrieval model $M_D$. To accelerate training convergence, we initialize our query encoder $\zeta$ from the query encoder of the same dual encoder retrieval model. Throughout training, the parameters $\theta_Q$ of the query encoder (and $\theta_D$ of the scoring function $\varphi$, if the latter is learnable) are fine-tuned. To support a memory and compute-efficient training setting without the need for large or parallel GPUs, the document representations $\mathbf{X}$ remain fixed, although in general they can be transformed by function $\varphi$ before computing the document similarity scores. As with all dense retrieval methods (e.g., [4, 9, 11, 12]), document representations are assumed to have been precomputed and indexed for fast inference runtimes.

A key difference between CODER and all dense retrieval methods is that, for each query, along with the $k$ positive (ground-truth) documents, the model is trained to jointly score a *large number* $N - k$ of top candidate documents retrieved by some *candidate* retrieval method $M_C$. We note that $M_C$ does not need to be the same method as the base method that provides the document representations and the query encoder. This potentially allows leveraging methods with different characteristics (e. g. methods with higher recall versus precision, or a lexical overlap / sparse representation such as BM25), and can prove beneficial, as shown in Section 2.5.

### 2.3.3 Loss function

To best take advantage of jointly scoring $N$ documents for each query, we choose the ListNet loss [13], which is the KL-divergence between a distribution over the predicted scores $\hat{\mathbf{s}}$ (given by Eq. (2.2)) for the $N$ candidate documents, and a distribution over the target (ground-truth) relevance labels $\mathbf{y} \in \mathbb{R}^N$, given by the dataset for the same set of candidates (the relevance score of positive documents is a positive scalar, while for negative or documents whose label is not explicitly defined it is set to $-\infty$):

$$\mathcal{L}\left(\mathbf{y}, \hat{\mathbf{s}}\right) = \mathrm{D_{KL}}\left(\sigma(\mathbf{y}) \mid\mid \sigma(\hat{\mathbf{s}})\right) = -\sum_{i=1}^{N} \sigma(\mathbf{y})_i \log \frac{\sigma(\hat{\mathbf{s}})_i}{\sigma(\mathbf{y})_i} \tag{2.3}$$

where $\sigma$ denotes the softmax function.

Jointly scoring a large number of *retrieved* candidates for each query, in combination with the KL-divergence loss, distinguishes our method from existing dense retrieval methods. This combination establishes and exploits a *context* for each query and it is key for obtaining a performance improvement over the base method, as we show in Section 2.5.4. The benefit is expected to be even greater for datasets which include multiple document labels per query, optionally defined over several levels of relevance. We show such results in Section 2.5.2.

## 2.4 Experimental Setup

**Datasets.** We conduct the experiments on passage and document retrieval tasks,[3] using two large publicly available IR collections in the domains of web and health retrieval. The first dataset is the MS MARCO Passage Retrieval dataset [19], used for training and evaluation. We evaluate the models trained on MS MARCO also on TREC Deep Learning Passage Retrieval track 2019 and 2020 [40, 41]. The second dataset used for training and evaluation is TripClick, a recently introduced health document retrieval dataset [20]. While the training data of MS MARCO contains only approximately 1 relevance judgement per query, the TripClick collection has the advantage of providing a much larger set of relevance information, namely approximately 42, 9, and 3 data points per query in HEAD, TORSO, and TAIL sets, respectively. As shown in the next section, this is particularly beneficial when optimizing over a large ranking context in a list-wise manner.

**Baselines.** We choose several dense retrieval models as baselines, i. e. “base models” subjected to CODER fine-tuning:

[3]When referring to the unit of retrieval we use the terms “passage” and “document” interchangeably.

| **Parameter** | **Value** |
|---|---|
| Max. query length | 32 |
| Max. doc. length (MS MARCO) | 256 |
| Max. doc. length (TripClick) | 512 |
| Batch size | 32 |
| Optimizer | RAdam |
| Adam epsilon | 1.3e-7 |
| Learning rate | 1.73e-6 |
| LR warmup steps | 9000 |
| Weight decay | 9.5e-5 |
| Dropout | 0.1 |
| Max. gradient clipping | 1.0 |
| $d_{\text{model}}$ | 768 |

Table 2.1: Main configuration parameters of CODER (without transformation of document representations).

1. RepBERT [4], a BERT-based model with a typical dual encoder architecture which underpins all state-of-the-art dense retrieval methods, trained using a triplet Max-Margin loss.
2. TAS-B [9], which, besides being a top-performing dense retrieval method on the MS MARCO / TREC-DL 2019, 2020 datasets, it also represents methods that have been optimized with respect to their training process (details in Section 2.2).
3. Finally, to explore the limits of CODER, we use it to fine-tune a trained CoCondenser retriever [24], the state-of-the-art dense retrieval model *that does not* make use of query-document term interactions, cross-encoder teacher models or additional pseudo-labeled data samples, but instead relies on extensive corpus-specific, self-supervised pre-training using a special architecture and contrastive loss component. It is a particularly challenging baseline for our CODER framework, because it has been trained through (a) mining for hard negatives using a trained version of the model itself, and (b) the Negative LogLikelihood (InfoNCE) loss, which is "nearly" list-wise (it differs from our KL-divergence loss only when there are more than one positive candidates).

**Configurations.** For the CODER framework, we use the following notation: $M_F \rightarrow$ CODER$(M_D, M_C)$, where $M_D$ is the base model used to encode documents into document embedding vectors (and initialize the query encoder weights), $M_C$ is the first-stage retrieval method used to procure the candidate (context) documents reranked during the CODER training process, and $M_F$ is the retrieval method used as a first stage when CODER is used as a reranking method during inference; $M_F$ vanishes in case CODER is used directly for single-stage dense retrieval, and the notation CODER$(M_D, M_C)$ is used instead. In addition to TAS-B and RepBERT, we also experiment with BM25 (Anserini implementation [42]) as $M_C$ and $M_F$ methods.

| Model | MS MARCO dev | | TREC DL 2019 | | TREC DL 2020 | | Latency |
|---|---|---|---|---|---|---|---|
| | MRR@10 | nDCG@10 | MRR@10 | nDCG@10 | MRR@10 | nDCG@10 | (ms/query) |
| BM25 [Anserini] | 0.187 | 0.234 | 0.843 / 0.682 | 0.497 / 0.417 | 0.820 / 0.655 | 0.488 / 0.412 | 50[1] |
| L2Re(ANCE) [5] | 0.341 | - | - | 0.675 | - | - | 47 |
| Co-BERT* [36] | - | - | 0.958 | 0.700 | 0.839 | 0.699 | > 1000 |
| ColBERT*v1; v2 [3, 29] | 0.360; 0.397 | - | - | - | - | - | 458 |
| BM25 $\rightarrow$ ColBERT* [3] | 0.349 | - | - | - | - | - | [BM25] + 61 |
| RocketQAv1; v2 [10, 21] | 0.370; 0.388 | - | - | - | - | - | - |
| CoCondenser [our evaluation] | 0.381 | 0.446 | 0.971 / 0.879 | 0.715 / 0.656 | 0.937 / 0.833 | 0.680 / 0.618 | $\simeq$ [RepBERT] |
| RepBERT (abbrev: RB) [our eval.] | 0.304 | 0.359 | 0.917 / 0.766 | 0.616 / 0.548 | 0.902 / 0.763 | 0.621 / 0.561 | 70 |
| BM25 $\rightarrow$ RepBERT | 0.317 | 0.373 | 0.969 / 0.795 | 0.674 / 0.593 | 0.893 / 0.781 | 0.640 / 0.579 | [BM25] + 5.8 |
| BM25 $\rightarrow$ CODER(RB, BM25) | 0.326 | 0.384 | 0.953 / 0.798 | 0.675 / 0.600 | 0.914 / **0.816** | 0.654 / 0.593 | [BM25] + 5.8 |
| BM25 $\rightarrow$ CODER(RB, RB) | **0.327**$^{ab}$ | **0.385**$^{ab}$ | 0.953 / **0.806** | 0.677 / **0.603**$^{a}$ | 0.898 / 0.787 | 0.672 / **0.611**$^{ab}$ | [BM25] + 5.8 |
| RB $\rightarrow$ CODER(RB, RB) | 0.324 | 0.383 | 0.905 / 0.785 | 0.650 / 0.593 | 0.918 / 0.785 | 0.660 / 0.598 | [RepBERT] + 5.8 |
| CODER(RB, BM25) | 0.311 | 0.368 | 0.855 / 0.750 | 0.606 / 0.552 | 0.906 / 0.790 | 0.603 / 0.550 | [RepBERT] |
| CODER(RB, RB) | 0.325 | 0.384 | 0.905 / 0.785 | 0.652 / 0.593 | 0.918 / 0.785 | 0.660 / 0.598 | [RepBERT] |
| TAS-B [9] | 0.340 | 0.402 | 0.892 | 0.712 | 0.843 | 0.693 | 64 |
| TAS-B [our evaluation] | 0.344 | 0.408 | 0.951 / 0.875 | 0.721 / 0.659 | 0.921 / 0.832 | 0.685 / 0.620 | < 50 |
| BM25 $\rightarrow$ TAS-B | 0.343 | 0.404 | 0.971 / 0.857 | 0.723 / 0.648 | 0.918 / 0.838 | 0.696 / **0.633** | [BM25] + 5.5 |
| BM25 $\rightarrow$ CODER(TAS-B, BM25) | 0.349 | 0.409 | 0.983 / 0.872 | 0.727 / 0.654 | 0.935 / **0.846** | 0.690 / 0.629 | [BM25] + 5.5 |
| BM25 $\rightarrow$ CODER(TAS-B, TAS-B) | 0.350 | 0.411 | 0.971 / 0.828 | 0.728 / 0.654 | 0.926 / **0.846** | 0.693 / **0.630** | [BM25] + 5.5 |
| TAS-B $\rightarrow$ CODER(TAS-B, TAS-B) | **0.355**$^{ab}$ | **0.419**$^{ab}$ | 0.966 / 0.857 | 0.728 / **0.668** | 0.923 / **0.844** | 0.686 / 0.623 | [TAS-B] + 5.5 |
| CODER(TAS-B, BM25) | 0.347 | 0.409 | 0.965 / **0.890** | 0.723 / 0.665 | 0.934 / 0.835 | 0.678 / 0.612 | [TAS-B] |
| CODER(TAS-B, TAS-B) | **0.355**$^{ab}$ | **0.419**$^{ab}$ | 0.966 / 0.857 | 0.728 / **0.668** | 0.923 / **0.844** | 0.686 / 0.623 | [TAS-B] |

Table 2.2: Performance for passage ranking when applying CODER to the RepBERT (middle section) and TAS-B (bottom section) base methods. In the notation $M_F \rightarrow$ METHOD$(M_D, M_C)$: $M_F$ is the method used for first stage retrieval when using METHOD for reranking, $M_D$ is the base method and $M_C$ is the retrieval method which provides the context (candidate) passages during training. **Bold font denotes best results within the same section (separated by continuous rules)**. Results of the statistical significance tests (paired $t$-test) are reported only for the best performing models, where the symbols $^a$ and $^b$ denote a significant improvement ($p < 0.05$) with respect to the base $M_D$ and BM25$\rightarrow M_D$, respectively. For TREC DL, two values are given for each metric separated by a slash, corresponding to the lenient / strict (official) interpretation of relevance labels. Models with * use cross-encoder term interactions.

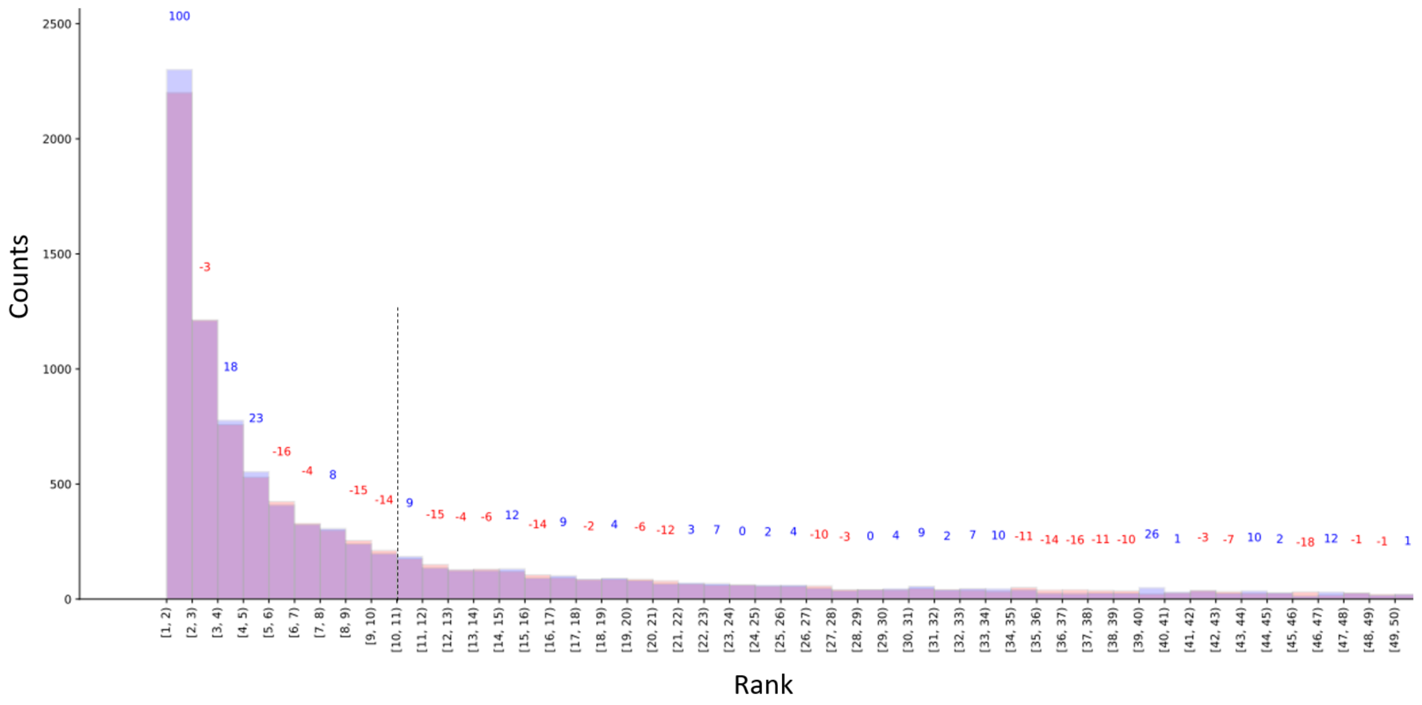


Figure 2.3: Histogram of ranks of ground-truth relevant documents across the MS MARCO validation set. Blue bars: CODER(TAS-B), Red bars: TAS-B. Training through CODER results in the relevant document more frequently ranking at the very top ranking positions.

## 2.5 Results and discussion

### 2.5.1 Results on MS MARCO and TREC DL

After only a fast and efficient fine-tuning (3.5 hours for TAS-B, 4.5 hours for RepBERT on a single NVIDIA TITAN RTX GPU), we observe a substantial performance benefit when applying our CODER framework to TAS-B and RepBERT, as seen in Table 2.2.

CODER improves retrieval performance remarkably compared to both the original (single-stage) RepBERT, as well as two-stage cascade BM25→RepBERT. CODER confers the largest performance benefit on RepBERT when reranking BM25 candidates in a cascade. The single-stage retriever fine-tuned through CODER is much improved compared to the original RepBERT, and almost as effective as the cascade, without introducing any latency or complexity.

Our framework also significantly improves the performance of the highly optimized TAS-B method, which leverages hard negatives and dual knowledge distillation from two powerful cross-encoder models, BERT-Reranker and ColBERT. To better understand where the improvement of ranking metrics comes from, in Fig. 2.3 we plot a histogram of the ranking assigned to the ground-truth relevant document across all predictions on the MS MARCO validation set. We observe that compared to TAS-B (red bards), CODER training shifts the rank of the relevant documents from near-top towards the very top ranking positions.

We furthermore observe that using the CODER-trained query encoder directly for single-

| **Model** | **MRR@10** | **nDCG@10** |
|---|---|---|
| BM25[1] | 0.276 | 0.224 |
| Transformer-Kernel[1] | 0.434 | 0.284 |
| BERT-Dot (SciBERT)[2] | 0.530 | 0.243 |
| BERT-Cat (SciBERT; PMBERT)[2] | 0.595; 0.582 | 0.294; 0.298 |
| RepBERT (abbrev: RB) | 0.526 | 0.255 |
| BM25 $\rightarrow$ RepBERT | 0.538 | 0.262 |
| RB $\rightarrow$ CODER(RB, RB) | **0.637***  | **0.318*** |
| CODER(RB, RB) | 0.634 | 0.316 |

Table 2.3: Performance when applying CODER to RepBERT on the TripClick HEAD dataset, using multi-level (DCTR) relevance labels (metrics cut-off of 10). All CODER results are statistically significant (paired $t$-test, $p < 0.05$) with respect to both the base and BM25$\rightarrow$base methods. The symbol * on best results denotes statistically significant improvement with respect to all baselines. Results with [1] are from Rekabsaz et al. [20], with [2] from Hofstätter et al. [28].

stage dense retrieval is exactly as effective as using CODER to rerank TAS-B candidates in a cascade, on both MS MARCO and TREC DL tracks (Table 2.2). This result suggests that under certain conditions, CODER can generalize its ranking function from the provided training context (limited set of fixed hard negatives) to the entire dataset, even without the use of dynamic negative mining, huge batch sizes or "denoising" as seen in previous work [10, 11].

Putting these results into context, we can see that simply by training through contextual reranking, a dual encoder model such as TAS-B can improve to the point that its reranking effectiveness is higher than a powerful model such as ColBERT, a term-interaction model still fast enough to be practically considered for real-time reranking, but at a fraction of the reranking latency cost (5.5 ms vs 61 ms, i.e., less than 1/10th). Its single-stage ranking performance approaches the one by ColBERT, while being about 10x faster (about 50 ms vs. 458 ms), making it a top-performing method within its latency class.

Finally, to test the limits of CODER (see Section 2.4), we apply it to fine-tuning a trained CoCondenser retriever, the SOTA dense retrieval model that does not rely on using a cross-encoder in its pipeline. We observe a slight improvement of 0.002 MRR@10 and 0.004 Recall@10 (the latter statistically significant) on the MS MARCO validation set. This smaller improvement on MS MARCO is expected, given that it is a dataset providing very few relevance judgements per query and thus (a) poor context for training, and (b) an evaluation setting that may be unsuitable to resolve differences in ranking effectiveness for a contextually-trained model. For this reason, we additionally evaluate the models on TripClick.

### 2.5.2 Results on TripClick

Following Rekabsaz et al. [20], we report performance in terms of MRR@10, nDCG@10 and Recall@10; nDCG@10 is considered the most important metric, as multiple relevant documents per query exist, and the DCTR relevance set additionally uses multiple levels of relevance. The results are presented in Table 2.3: Using the same hyperparameters as in MS MARCO, CODER fine-tuning tremendously improves the performance of RepBERT trained on TripClick, both in reranking as well as in single-stage dense retrieval. The improvement is especially pronounced on the HEAD subset, where many relevance judgements per query are available. CODER achieves the SOTA performance by a large margin, ahead of all published results in the literature and on the TripClick leaderboard[4], including ensembles of heavyweight cross-encoder BERT Rerankers which were pre-trained on the domain-specific PubMedBERT (medicine) and SciBERT (science) corpora [28]. CODER also significantly outperforms the best existing dense retrieval method (BERT-Dot pre-trained on SciBERT [28]) on the TORSO and TAIL subsets (see Table 2.4). However, presumably because these subsets consist of rare queries with significantly fewer relevance judgements per query, it falls behind the domain-pretrained cross-encoders.

Finally, we use TripClick's validation and test sets purely for "zero-shot" evaluating the RepBERT, CODER(RepBERT), CoCondenser and CODER(CoCondenser) models trained exclusively on MS MARCO. We emphasize that, uniquely in this setting, these models have not been trained or fine-tuned on TripClick; they are the same models described in Section 2.5.1, now evaluated on a large dataset with severe distribution shift (biomedical domain). While zero-shot evaluation is used increasingly often to demonstrate the effectiveness and generalizability of large transformer models (e.g. [43, 44, 45]), here we use it as a means to bypass the challenge of the expensive pre-training and fine-tuning of CoCondenser on TripClick. Results are shown in Table 2.5: naturally, we observe that zero-shot performance in absolute terms is low; however, CODER-trained models perform always better. Moreover, the performance order CODER(CoCondenser) > CoCondenser > CODER(RepBERT) > RepBERT, consistent with our observations for MS MARCO, holds also here in almost every comparison.

### 2.5.3 Efficiency

What is the additional cost of using CODER? Using a single NVIDIA TITAN RTX GPU (on a node with Intel Xeon Gold 6142 CPU), it takes about 186 ms to rank 1000 candidates per query in a batch of 32 queries (out of which less than 10ms refer to computing representations and scores, with the rest taken up by batching and loading samples to the GPU), i.e., a latency of 5.5-5.8 ms per query is introduced when using CODER as a second-stage reranker in a cascade. Using CODER as a single-stage dense retriever only requires the same processing time as the base method, e.g. RepBERT or TAS-B. Table 2.2 reports the latency reported in [4], but this in practice will be determined by the time for loading the query sequence to the GPU and encoding it (in our setup, approx. 5.4 ms per query, out of which approx. 0.3 ms is the time to compute the representation), in addition to the time for finding the approximate nearest neighbors using a library such as FAISS [46].

### 2.5.4 Analysis of key factors

#### Importance of context

In this section we wish to assess the intuitions that scoring a document within a context of other documents related to the same query can be advantageous, and that a list-wise loss

[4]https://tripdatabase.github.io/tripclick/

function like the one we present in Section 2.3.3 is better equipped to leverage this context compared to a superposition of separate pair-wise loss components. We first study the impact of varying the type and number of negative documents during training.

In Figure 2.4 we show how the performance of a model initialized from a trained RepBERT base model evolves during fine-tuning through CODER, measured in MRR@10 on our MS MARCO validation set when reranking 1000 candidates first retrieved for each query by BM25. Different curves correspond to different training settings. The leftmost evaluation point corresponds to the best model checkpoint achievable through standard triplet-based training.

**Composition of negatives:** We observe that training with only randomly sampled negatives, even in large numbers, leads to a deterioration of performance. We note that this is the case because through CODER we are optimizing a baseline model which has already been trained to peak performance through random negative documents. Using a small number of retrieved candidates together with in-batch random negatives, as in current SOTA methods [5, 10], again leads to deteriorating performance, even with many random negatives. Only by using a large number of coherent, query-specific negative samples (i.e. retrieved by a retrieval method) for the same query can CODER extract additional performance from the baseline. Moreover, adding 1000 additional random documents per query (in-batch negatives) on top of those does not yield any benefit, presumably because they are not adding any context (as they are unrelated to the query and documents under assessment) and/or they are not sufficiently challenging.

**List-wise loss:** When training with the most commonly used pair-wise loss (Multi-Label Max Margin, purple dashed curve), even when using 1000 retrieved candidates as negatives within exactly the same training setup, i.e. including the same positive and negative examples for the same query in the same batch, we observe that performance only modestly improves performance (+0.009 MRR@10). The improvement can be attributed to the fact that the model now encounters a large number of coherent, query-specific documents for the same query during a single step of training, which offers a more complete and accurate view of the loss landscape and thus leads to a more accurate update of parameters. However, the ranking context is exploited more effectively when using a list-wise KL-divergence loss (dark blue curve), as CODER fine-tuning improves MRR@10 from 0.345 to 0.363 (+0.018).

Finally, we note that the training loss continuously decreased throughout training in all settings mentioned above (see Figure B.2 in the Appendix), including in the ones where performance on the validation set was deteriorating at the same time (Figure 2.4). This fact suggests that deteriorating performance in those settings can be interpreted as overfitting, when there is insufficient information/signal captured by the training objective in order to learn to rank more effectively.

### Number of context documents

We now investigate the importance of the quantity of negative documents per query during training. In Figure 2.4, where the number of BM25-retrieved candidate negatives is adjusted, we observe increasing performance per additional document until we reach a number in the order of the dimensionality of the embedding space (here, 768 for BERT-base). Increasing

| Model | DCTR Head | | RAW Head | | | RAW Torso | | | RAW Tail | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **MRR** | **nDCG** | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** |
| BM25[1] | 0.276 | 0.224 | - | 0.199 | 0.128 | - | 0.206 | 0.262 | - | 0.267 | 0.409 |
| Transformer-Kernel[1] | - | 0.284 | - | 0.284 | 0.167 | - | 0.272 | 0.321 | - | 0.295 | 0.459 |
| BERT-Dot (SciBERT)[2] | 0.530 | 0.243 | - | - | - | - | - | - | - | - | - |
| BERT-Cat (SciBERT)[2] | 0.595 | 0.294 | - | - | - | **0.459** | **0.360** | - | **0.377** | **0.408** | - |
| RepBERT (abbrev: RB) | 0.526 | 0.255 | 0.574 | 0.344 | 0.199 | 0.338 | 0.246 | 0.309 | 0.254 | 0.268 | 0.404 |
| BM25 $\rightarrow$ RepBERT | 0.538 | 0.262 | 0.592 | 0.356 | 0.204 | 0.359 | 0.269 | 0.340 | 0.278 | 0.297 | 0.445 |
| RB $\rightarrow$ CODER(RB, RB) | **0.637***  | **0.318*** | **0.679*** | **0.421*** | **0.235*** | 0.433 | 0.308 | 0.355 | 0.296 | 0.315 | 0.469 |
| CODER(RB, RB) | 0.634 | 0.316 | 0.674 | 0.419 | 0.234 | 0.433 | 0.308 | 0.355 | 0.296 | 0.315 | 0.468 |

Table 2.4: Performance when applying CODER to RepBERT on the TripClick dataset, using multi-level (DCTR) and binary (RAW) relevance labels (cut-off of 10). The symbol * on best results denotes statistically significant (paired $t$-test, $p < 0.05$) improvement with respect to all baselines. Results with [1] are from Rekabsaz et al. [20], with [2] from Hofstätter et al. [28].

| **Model** [Zero-shot] | DCTR Head | | RAW Head | | | RAW Torso | | | RAW Tail | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **MRR** | **nDCG** | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** |
| RepBERT | 0.233 | 0.107 | 0.278 | 0.149 | 0.085 | 0.205 | 0.130 | 0.151 | 0.117 | 0.122 | 0.195 |
| CODER(RepBERT) | 0.244 | 0.113 | 0.294 | 0.157 | 0.091 | 0.211 | 0.139 | 0.166 | 0.127 | 0.137 | 0.223 |
| Cocondenser | 0.242 | 0.114 | 0.293 | 0.156 | 0.091 | **0.217** | 0.144 | 0.178 | 0.153 | 0.162 | 0.254 |
| CODER(Cocondenser) | **0.251** | **0.117** | **0.305** | **0.161** | **0.093** | 0.216 | **0.146** | **0.182** | **0.154** | **0.164** | **0.259** |

Table 2.5: Zero-shot results: all models are **trained on MS MARCO** but **evaluated on TripClick**. For CODER, the parenthesis indicates which model was used to provide negatives and as base for fine-tuning.

the number of negatives beyond this point yields diminishing returns, consistently with our analysis in Section 2.1.3.

We note that these numbers of negative *candidates* (as opposed to random documents) per query are much higher than the ones used in all contemporary work (max. 4 candidates have been employed in Qu et al. [10] and 30 in Gao and Callan [24]); the reason that such a high number is necessary in order to achieve a performance improvement is that we are fine-tuning a model already trained to saturation. Only a large number of retrieved candidate negatives provides enough training signal to overcome overfitting and improve performance. This is evidenced by the fact that in the rest of the settings, training loss was decreasing at the same time that performance on the validation was deteriorating (see Figure B.3 in the Appendix).

The above results suggest that most dense retrieval models would likely benefit from training using a context, i.e., a large number of retrieved candidate documents per query, combined with an appropriate list-wise loss.

### 2.5.5 How context can help even without parametric modeling of document relationships

The improvement of our contextual reranking framework over standard triplet training can be attributed to the fact that the model now encounters many more query-specific candidate documents for the same query during a single step of training, which offers a more complete and precise view of the loss landscape and thus leads to a more accurate update of parameters in the same step (also see discussion in Sec. 2.1.3). However, this alone constitutes a "weak" exploitation of ranking context offered by CODER.

The ranking context is exploited more effectively when using a list-wise KL divergence loss. We hypothesize that one reason is that often, among the retrieved candidates, some documents which have not been labeled as positive, are in fact relevant (i.e. false/mislabeled negatives) [10]. The KL-divergence function is well-positioned to deal with this case, compared to a pair-wise loss (e.g. Max-margin loss) and the Negative Loglikelihood Loss (NLL, a.k.a. InfoNCE) as used e.g. in Karpukhin et al. [7] and Qu et al. [10]. The KL-divergence loss, which compares the distribution of predicted scores against the annotated relevance scores, does not directly penalize assigning a high score to a document annotated as non-relevant; instead, it severely penalizes assigning a low score to a ground-truth relevant document. Therefore, as long as the ground-truth positive document $p$ receives a not-too-low normalized relevance score $\hat{s}_p = \text{softmax}(\varphi(\mathbf{X}))_p$, e.g. $\hat{s}_p > 0.2$, which allows it to escape the very steep part of the loss curve $L(\hat{s}_p) = -\log(\hat{s}_p)$ close to $\hat{s}_p = 0$, the loss will still be small. Thus, it will not severely affect the model's parameters to erroneously force ranking $p$ higher than the false negatives.

Of course, the more such false negative documents exist, receiving non-zero weight in the predicted score distribution, the more difficult it becomes for $\hat{s}_p$ to be high, which leads to a higher loss. However, the existence of more than one labeled positives $(k > 1)$ per query alleviates this problem: the individual normalized scores of the $k$ positives will be affected less by the existence of false negatives, and because of the form of $L(x) = -\log(x)$, the overall loss will be smaller than in the case of $k = 1$. This is an additional benefit the KL-divergence loss has over NLL, as the NLL only takes into account a single positive document at a time.

## 2.6 Conclusion

In this chapter, we have examined the importance of ranking context and the effect of its constituent parts, i.e., a fully list-wise loss, a large number of negatives, and retrieved (query-specific) instead of random negatives, and showed that they are all important ingredients for improving performance. We demonstrated that a lightweight reranking framework designed to leverage context is sufficient to significantly enhance the effectiveness of a wide range of dense retrieval models, without expensive cross-encoder distillation, pseudo-labeling or "denoising" negatives. The only computational overhead is a fast, resource-light fine-tuning process, with little (when the model is used as a reranker) to no (when used as a single-stage retriever) extra computational cost during inference.

We have thus presented a practical method to leverage the benefits of contextual similarity learning, previously utilized only in Learning-to-Rank approaches, in the training of state-of-the-art dense retrieval models based on pre-trained transformer language models.

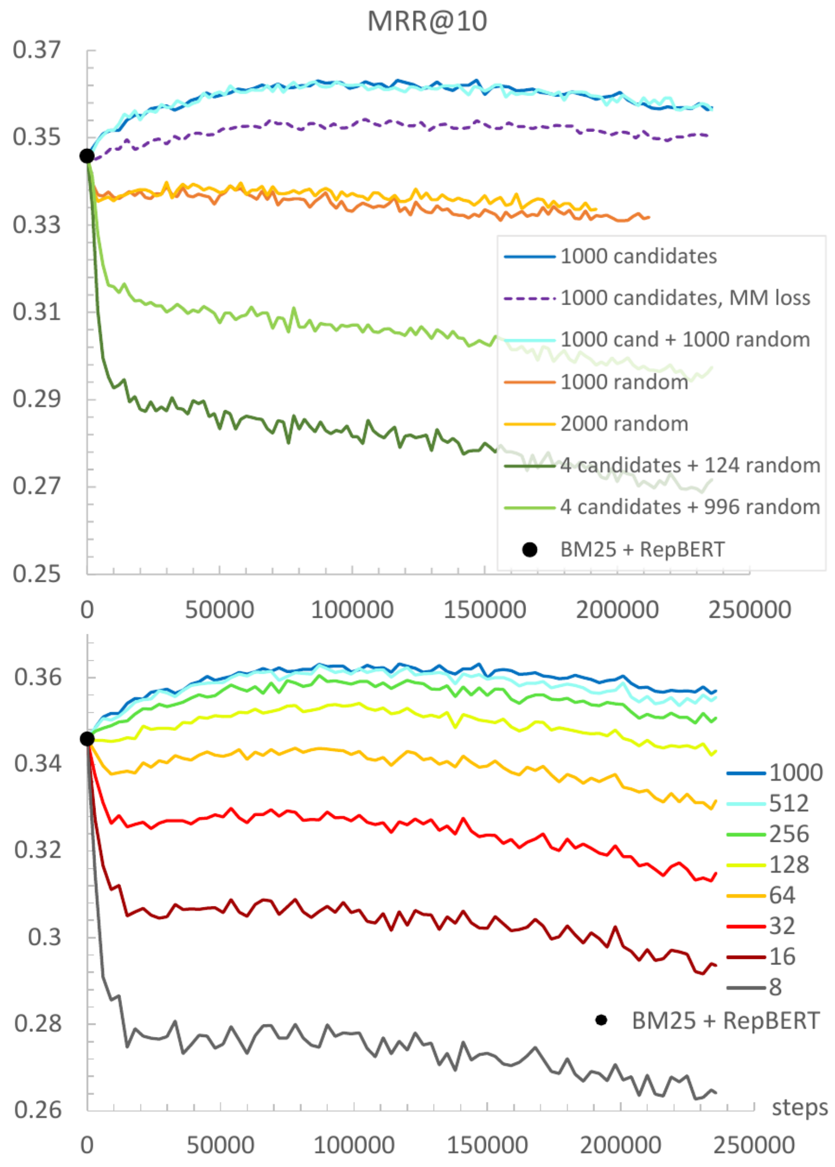


Figure 2.4: Performance of BM25→CODER(RepBERT) on MS MARCO validation set over training steps. The left-most point corresponds to reranking BM25 candidates using the fully trained RepBERT. **Top**: Effect of type and number of documents used as negatives. The purple dashed curve corresponds to training with a pair-wise (Max Margin) loss. **Bottom**: Effect of number of BM25 candidates used as negatives during training.

# Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019. doi: 10.18653/v1/N19-1423.

[2] Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *arXiv:1901.04085 [cs]*, April 2020. URL http://arxiv.org/abs/1901.04085. arXiv: 1901.04085.

[3] Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020.

[4] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv:2006.15498 [cs]*, July 2020. URL http://arxiv.org/abs/2006.15498. arXiv: 2006.15498.

[5] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Learning To Retrieve: How to Train a Dense Retrieval Model Effectively and Efficiently. *arXiv:2010.10469 [cs]*, October 2020. URL http://arxiv.org/abs/2010.10469. arXiv: 2010.10469.

[6] Clément Calauzènes, Nicolas Usunier, and Patrick Gallinari. On the (non-)existence of convex, calibrated surrogate losses for ranking. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/50f3f8c42b998a48057e9d33f4144b8b-Paper.pdf.

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

[8] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for*

*Computational Linguistics*, 9:329–345, April 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00369. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00369/100684/Sparse-Dense-and-Attentional-Representations-for.

[9] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and A. Hanbury. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *SIGIR*, 2021. doi: 10.1145/3404835.3462891.

[10] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *arXiv:2010.08191 [cs]*, May 2021. URL http://arxiv.org/abs/2010.08191. arXiv: 2010.08191.

[11] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*, October 2020. URL http://arxiv.org/abs/2007.00808. arXiv: 2007.00808.

[12] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada, July 2021. ACM. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462880. URL https://dl.acm.org/doi/10.1145/3404835.3462880.

[13] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

[14] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 85–92, Santa Clara CA USA, September 2019. ACM. ISBN 978-1-4503-6881-0. doi: 10.1145/3341981.3344218. URL https://dl.acm.org/doi/10.1145/3341981.3344218.

[15] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. Learning a Deep Listwise Context Model for Ranking Refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 135–144, New York, NY, USA, June 2018. Association for Computing Machinery. ISBN 978-1-4503-5657-2. doi: 10.1145/3209978.3209985. URL https://doi.org/10.1145/3209978.3209985.

[16] Sebastian Bruch, Shuguang Han, Mike Bendersky, and Marc Najork. A stochastic treatment of learning to rank scoring functions. In *Proceedings of the 13th ACM*

*International Conference on Web Search and Data Mining (WSDM 2020)*, pages 61–69, 2020.

[17] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. SetRank: Learning a Permutation-Invariant Ranking Model for Information Retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 499–508, New York, NY, USA, July 2020. Association for Computing Machinery. ISBN 978-1-4503-8016-4. doi: 10.1145/3397271.3401104. URL https://doi.org/10.1145/3397271.3401104.

[18] Zhizhong Chen and Carsten Eickhoff. Poolrank: Max/min pooling-based ranking loss for listwise learning & ranking balance. *ArXiv*, abs/2108.03586, 2021.

[19] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]*, October 2018. URL http://arxiv.org/abs/1611.09268. arXiv: 1611.09268.

[20] Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2513. Association for Computing Machinery, New York, NY, USA, July 2021. ISBN 978-1-4503-8037-9. URL https://doi.org/10.1145/3404835.3463242.

[21] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.224. URL https://aclanthology.org/2021.emnlp-main.224.

[22] Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, 2021. doi: 10.18653/v1/2021.findings-acl.191. URL http://arxiv.org/abs/2108.06027. arXiv: 2108.06027.

[23] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. In *EMNLP*, 2021.

[24] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *ArXiv*, abs/2108.05540, 2021.

[25] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. *Jointly Optimizing Query Encoder and Product Quantization to Improve Retrieval Performance*, page 2487–2496. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450384469. URL https://doi.org/10.1145/3459637.3482358.

[26] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 2970–2978, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330677. URL https://doi.org/10.1145/3292500.3330677.

[27] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4): 1–325, 2021.

[28] Sebastian Hofstätter, Sophia Althammer, Mete Sertkan, and Allan Hanbury. Establishing Strong Baselines for TripClick Health Retrieval. *arXiv:2201.00365 [cs]*, January 2022. URL http://arxiv.org/abs/2201.00365. arXiv: 2201.00365.

[29] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction, December 2021. URL http://arxiv.org/abs/2112.01488. arXiv:2112.01488 [cs].

[30] Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. *A Modern Perspective on Query Likelihood with Deep Generative Retrieval Models*, page 185–195. Association for Computing Machinery, New York, NY, USA, 2021. ISBN 9781450386111. URL https://doi.org/10.1145/3471158.3472229.

[31] Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. BERT-QE: Contextualized query expansion for document re-ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4718–4728, 2020.

[32] Shahrzad Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. Ceqe: Contextualized embeddings for query expansion. In *European Conference on Information Retrieval*, pages 467–482. Springer, 2021.

[33] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1723–1727, 2021.

[34] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, 2020.

[35] Luyu Gao, Zhuyun Dai, and Jamie Callan. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[36] Xiaoyang Chen, Kai Hui, Ben He, Xianpei Han, Le Sun, and Zheng Ye. Incorporating Ranking Context for End-to-End BERT Re-ranking. In Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty, editors, *Advances in Information Retrieval*, Lecture Notes in Computer Science, pages 111–127, Cham, 2022. Springer International Publishing. ISBN 978-3-030-99736-6. doi: 10.1007/978-3-030-99736-6_8.

[37] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*, February 2020. URL http://arxiv.org/abs/1910.01108. arXiv: 1910.01108.

[38] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019. URL http://arxiv.org/abs/1907.11692. arXiv: 1907.11692.

[39] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced Language Representation with Informative Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1139. URL https://www.aclweb.org/anthology/P19-1139.

[40] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820*, 2020.

[41] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the trec 2020 deep learning track. *arXiv preprint arXiv:2102.07662*, 2021.

[42] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1253–1256, New York, NY, USA, August 2017. Association for Computing Machinery. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080721. URL https://doi.org/10.1145/3077136.3080721.

[43] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models

are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[44] Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language Models are General-Purpose Interfaces, June 2022. URL http://arxiv.org/abs/2206.06336. arXiv:2206.06336 [cs].

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision, February 2021. URL http://arxiv.org/abs/2103.00020. arXiv:2103.00020 [cs].

[46] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *arXiv:1702.08734 [cs]*, February 2017. URL http://arxiv.org/abs/1702.08734. arXiv: 1702.08734.

# Chapter 3

# Mitigating bias in search results through contextual document reranking and neutrality regularization

## 3.1 Introduction

Information Retrieval (IR) systems reflect and may even exaggerate societal biases and stereotypes in their results [1, 2, 3, 4, 5]. If optimization of the underlying IR models is left to exclusively utility-oriented objectives, search engines, through the continual feedback loop of user interactions, can reinforce these biases in society (and hence back in IR systems) [6, 7, 8, 9, 10]. This accentuates the need for bias-aware IR models, in which fairness constraints are imposed with an adjustable degree of effect to control the fairness-utility trade-off in retrieval results [11, 12, 13, 14, 15, 16, 17].

To this end, Rekabsaz et al. [12] recently introduced MSMARCO$_{\text{FAIR}}$, a reproducible evaluation framework to measure gender bias in the text contents of search results, particularly suited to assessing the interplay of bias mitigation and utility in deep IR models. The framework identifies a set of *bias-sensitive queries*, singled out from the queries of the MS MARCO dataset [18]. Given such a query, an effective and bias-aware IR model should highly rank relevant documents with a balanced or neutral representation of genders in their text contents.

Beyond the notion of bias with respect to so-called *protected attributes* such as gender or ethnicity, we note that neutrality of documents can also refer to opinion or political biases: if a query can be answered by a relevant document with purely factual or unbiased content, this document is preferable to a similarly relevant but overtly biased document, which, notwithstanding reliability, may contribute to polarization.

As an example, a query such as “what is the role of a governor?” can and should be answered in a gender-neutral way. A document reading *“The governor is the chief executive of the state.* His *duties include ... /* he *is responsible for ...”* contains words charged/biased with respect to gender and induces an unnecessarily gender-biased exposition. Thus, it would be desirable to rank higher a document offering equally relevant information but using either gender-neutral words or *gender-representative* words in a balanced way (e.g. “he or she”).

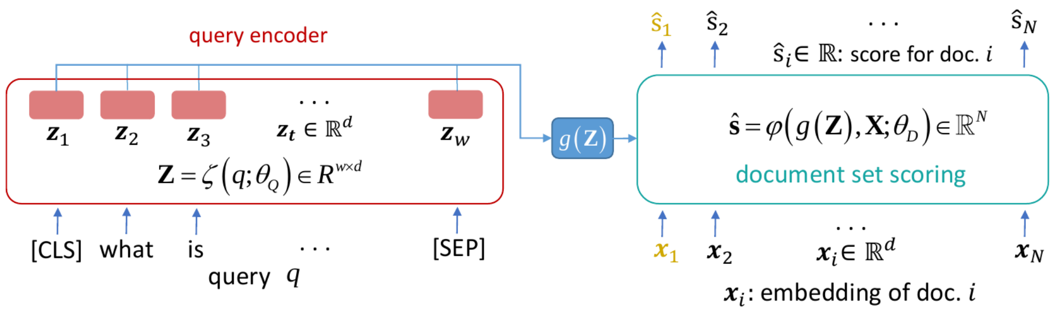

query encoder
$z_1$ $z_2$ $z_3$ $\cdots$ $z_t \in \mathbb{R}^d$ $z_w$
$\mathbf{Z} = \zeta(q;\theta_Q) \in R^{w\times d}$
[CLS] what is $\cdots$ [SEP]
query $q$
$g(\mathbf{Z})$
$\hat{s}_1$ $\hat{s}_2$ $\cdots$ $\hat{s}_N$
$\hat{s}_i \in \mathbb{R}$: score for doc. $i$
$\hat{\mathbf{s}} = \varphi(g(\mathbf{Z}), \mathbf{X}; \theta_D) \in \mathbb{R}^N$
document set scoring
$x_1$ $x_2$ $\cdots$ $x_i \in \mathbb{R}^d$ $x_N$
$x_i$: embedding of doc. $i$

Figure 3.1: Schematic diagram of the contextual document embedding reranking (CODER) framework.

In this example, gender-representative words are pronouns, but additional instances include *"man"/"woman"*, *"father"/"mother"*, *"actor"/"actress"*, male/female names, etc. Similarly, a document answering a purely factual query about the current state of the economy (e.g. "is inflation now higher than in 70's") could be penalized if it contained words indicating political bias, such as *"trumpism"*, *"wokeness"*, *"MAGA"*, *"neoliberal"*, *"leftist"*, etc.

We note that this notion of *fairness through neutrality* is not applicable to all queries; rather, we measure it on a set of expert-curated queries for which bias is considered "socially problematic" [19, 20]. Furthermore, it is not applicable to all retrieval use cases; for example, when ranking job applicants, who are inevitably gendered, a more suitable framework would optimize for *fairness of exposure* [17] (e.g. making sure that, given the same level of fitness, female applicants are ranked higher than male applicants as often as male applicants are ranked higher than female applicants).

A variety of past works have proposed effective methods to address bias mitigation, often using some sort of list-wise optimization such as Gumbel-Softmax, stochastic optimization, or reinforcement learning [11, 17, 21, 22, 23]. However, to the best of our knowledge, almost all prior works operate solely on extremely shallow ranking models. As Cohen [24] shows, naively applying shallow methods to deep transformer based architectures often presents significant challenges. Given the scope of this contribution, we therefore consider the relevant work of Rekabsaz et al. [12], where the authors propose integrating adversarial training in deep ranking models in order to improve bias mitigation. Their adversarial method aims to remove gender-related information from the model's internal embeddings, making the predictions (potentially) agnostic to the explicit/implicit presence of gender concepts in a given query-document pair.

The inherent point-wise nature of this adversarial method fits well to the current dominating paradigm of point-/pair-wise optimization [25, 26, 27, 28], embraced mainly due to the practical and conceptual complexities of training deep IR models [24]. Despite the benefits of this approach in mitigating gender bias, previous work [12, 29, 30, 31] and our own experiments show that adversarial training can be highly unstable, with unabated fluctuations over fairness and utility metrics. This issue significantly impedes identifying a model checkpoint that reliably yields generalizable performance and operates within a desired range in the fairness-utility trade-off – a necessary aspect for the wide adoption of bias-aware IR models in practice.

We approach this topic by proposing a novel list-wise bias mitigation method that leverages CODER, the retrieval framework we introduced in Chapter 1, which enables set-based training of any pretrained dual-encoder deep IR model. The latter is a class of models that represents the state of the art in dense retrieval (e.g. [32, 33, 34]). The guiding principle of bias mitigation is that neutral/unbiased documents should be ranked higher than biased documents of the same or similar relevance. To this end, we extend CODER with a novel list-wise regularization term defined to support neutral documents in final relevance predictions.

We apply our bias-aware optimization method using CODER with TAS-B [32] as the base transformer encoder model. We compare our method with adversarial training for bias mitigation of TAS-B and a cross-encoder (query-document term interaction) BERT Reranker [35], the current SOTA for bias mitigation proposed by Rekabsaz et al. [12]. Ranking results as well as training dynamics are evaluated in terms of utility (MRR, and Recall) and neutrality/fairness (NFaiRR) metrics on the MSMARCO$_{\text{FAIR}}$ collection. Our results show that besides achieving state-of-the-art performance in terms of fairness for the same utility, our set-based neutrality regularization method, in contrast to adversarial alternatives, provides a stable optimization of the network in a short training time, and allows predictably adjusting the intensity of the trade-off between fairness and utility, in a far wider range.

Our contribution thus represents a significant step towards advancing bias mitigation in search, from a theoretical discussion point or a largely experimental research topic, to a practical approach that can be readily integrated in state-of-the-art retrieval systems deployed in industry.

## 3.2 Method

Our approach optimizes the parameters of a retrieval model such that it assigns scores to documents in proportion to their relevance to a query, while at the same time directly imposing a neutrality constraint on the top-ranked documents. To achieve this, we need a training setup where a large number of documents is simultaneously scored for the same query, and therefore the standard training setup using *(query, positive document, negative document)* triplets is unsuitable. Making use of in-batch documents (e.g., [33, 36, 37]) can indeed provide a large number of random documents as negatives; however, random negatives are only very rarely related to the query or each other, and have been convincingly shown to be less effective than retrieved negatives [33, 38, 39, 40]. Importantly, we wish to impose neutrality on the *top-ranked* documents retrieved by a system, whereas randomly sampled documents are extremely unlikely to end up in high-ranking positions and are thus poor targets for regularization. For these reasons, we instead utilize CODER, the contextual document embedding reranking framework we introduced in Chapter 1, which, given a query, jointly scores a large set of candidate documents that together constitute a *ranking context*.

Here, for quick reference, we provide a brief description of the framework, with more details and a schematic diagram (Figure 3.1) given in Chapter 1. A pre-trained transformer encoder $\zeta$ first transforms a tokenized query $q$ of length $w$ into a sequence of $d$-dimensional embedding vectors: $\mathbf{Z} = [\mathbf{z}_1; \ldots; \mathbf{z}_w] = \zeta\left(q; \theta_Q\right) \in \mathbb{R}^{w \times d}$, out of which an aggregator function $g(\mathbf{Z})$ extracts a single vector. In this work, as $\zeta$ we choose the query encoder from

TAS-B [32], which is based on DistilBERT [41] and was the most effective base model evaluated by Zerveas et al. [40]. The aggregation function here simply selects the output embedding corresponding to the first query token, i.e., `[CLS]`: $g(\mathbf{Z}) = \mathbf{z}_1 \in \mathbb{R}^d$.

A scoring function $\varphi$ computes a scalar relevance score $\hat{s}_i$ for each document embedding $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \ldots, N$, based on their similarity to the query embedding $g(\mathbf{Z})$. The set of $N$ documents consists of the ground-truth relevant document(s) and the top candidates retrieved for the same query by an arbitrary retrieval method (here, BM25 [42] by Anserini [43]) in advance. Their embeddings have been precomputed by the document encoder of a dual-encoder model (here TAS-B). As we have seen in the previous chapter (and also observed in other work, e.g. [33]) a large number of negative documents is essential for providing adequate signal to effectively capture relevance, and as before, we use $N = 1000$. Although the scoring function can in general be parametric, here again we simply use the dot-product, which is commonly used for evaluating similarity and was shown in the previous chapter to be effective:

$$\hat{\mathbf{s}} = \varphi\left(g(\mathbf{Z}), \mathbf{X}\right) = \mathbf{X} \cdot g(\mathbf{Z}) \in \mathbb{R}^N \tag{3.1}$$

Throughout training, the parameters of the query encoder are fine-tuned through the ListNet loss [44], which for a given query is equivalent to the KL-divergence between a distribution over the target (ground-truth) relevance labels $\mathbf{y} \in \mathbb{R}^N$, defined for the set of $N$ candidate documents (where the relevance of all documents not explicitly defined is assumed to be 0), and a distribution over the corresponding predicted scores $\hat{\mathbf{s}}$:

$$\mathcal{L}_u\left(\mathbf{y}, \hat{\mathbf{s}}\right) = \mathrm{D}_{\mathrm{KL}}\left(\sigma(\mathbf{y}) \mid\mid \sigma(\hat{\mathbf{s}})\right) = -\sum_{i=1}^{N} \sigma(\mathbf{y})_i \log \frac{\sigma(\hat{\mathbf{s}})_i}{\sigma(\mathbf{y})_i} \tag{3.2}$$

where $\sigma$ denotes the softmax function.

The loss function above guides parameter optimization towards maximizing the relevance of top-ranked documents, i.e., the *utility* for the user. To impose neutrality on the top-ranked documents, we add the following *neutrality* loss term to obtain the total loss:

$$\mathcal{L}_n\left(\mathbf{y}_n, \hat{\mathbf{s}}\right) = \mathrm{D}_{\mathrm{KL}}\left(\sigma(\hat{\mathbf{s}}) \mid\mid \sigma(\mathbf{y_n})\right) = -\sum_{i=1}^{C} \sigma(\hat{\mathbf{s}})_i \log \frac{\sigma(\mathbf{y_n})_i}{\sigma(\hat{\mathbf{s}})_i} \tag{3.3}$$

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_u + \lambda_r \mathcal{L}_n \tag{3.4}$$

where $\lambda_r$ is the regularization coefficient, $C$ is the cut-off rank for considering neutrality (we use $C = 10$), and $\mathbf{y_n}$ are the neutrality scores for each document. These are computed following Rekabsaz et al. [12], and are based on the frequency of occurrence of terms indicative of bias with respect to the protected attribute.

We note that the order of distributions in the asymmetric KL-divergence is reversed in the two loss terms: in the utility loss, we primarily penalize assigning a low score to ground-truth relevant documents (rather than the case of assigning a high score to documents which have not been annotated as relevant). This is desirable, among other reasons, because relevance annotations are sparse and many candidate documents can be relevant without having been marked as such [33]. By contrast, the neutrality loss primarily penalizes assigning high scores to documents with low neutrality scores, rather than the case of assigning a low score to neutral documents (since neutrality alone is not indicative of relevance).

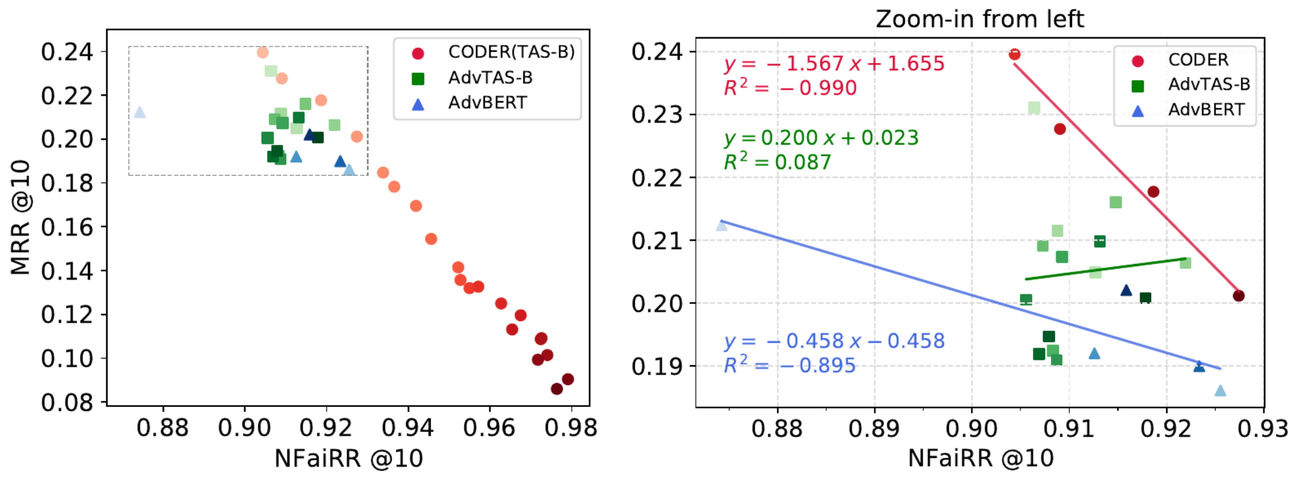


Figure 3.2: Utility (MRR@10) versus fairness (NFaiRR@10). The intensity of color corresponds to an increasing adversarial or regularization factor. Compared to adversarial baselines, regularization with CODER allows modulating fairness to much higher values, while for the same values of fairness, utility is higher.

## 3.3 Experiments

### 3.3.1 Resources

Our experimental setting closely follows Rekabsaz et al. [12]. Fairness and utility of the models are evaluated on 215 curated bias-sensitive queries, i. e. "gender-neutral queries for which biases in their retrieval results are considered as socially problematic" [19, 20], provided by MSMARCO$_{\text{FAIR}}$. The models are trained on the data provided by the MS MARCO Passage Retrieval collection [18], and retrieval is conducted on the collection's passages. The protected attribute is gender, defined in binary fashion using the gender-representative words (158 words for each gender) provided in previous work [12, 45]. These words are used to calculate the neutrality score for each document with the term occurrence threshold set to 1 (c. f. Rekabsaz et al. [12]).

### 3.3.2 Retrieval Models

All models are trained and evaluated by reranking candidates first retrieved by BM25 [42]. **CODER(TAS-B)** is our proposed bias mitigation approach explained in Section 3.2. We select TAS-B [32] as a base transformer encoder for CODER, due to its superior retrieval performance in reranking and dense retrieval scenarios. **AdvBERT** is the model introduced by Rekabsaz et al. [12] which applies adversarial training to a BERT Reranker [35]. The adversarial network in AdvBERT is defined on the output vector of the `[CLS]` token when both query and documents are passed to the BERT model. Following Rekabsaz et al. [12], the prediction label of the adversarial network is defined as a binary variable, which is set to 1 (gendered) if either the given query or document are not fully gender neutral texts, and 0 otherwise. AdvBERT approaches removing gender-related information in models using a gradient reversal mechanism [46]. The gradient of the loss corresponding to the adversarial "gender detector" is scaled by the *adversarial factor* $\lambda_a$, which allows tuning the intensity of bias mitigation. AdvBERT is the best performing model reported in Rekabsaz et al. [12], achieved by fine-tuning the BERT-Mini [47] model. **AdvTAS-B** applies a similar adversarial training procedure as AdvBERT to the TAS-B model, providing an adversarial baseline directly comparable with CODER(TAS-B). Because AdvBERT is a cross-encoder model, while TAS-B is a dual encoder, in AdvTAS-B the adversarial network is defined over the concatenation of the query and document embeddings (the `[CLS]` output of the corresponding encoder), and training aims to remove gender-related information in both query and document encoders.

### 3.3.3 Evaluation of Bias Mitigation and Utility

The fairness of the ranking models is evaluated in terms of the *Normalized Fairness of Retrieval Results (NFaiRR)* metric [12]. The NFaiRR metric measures to what extent the contents of the retrieved documents show a balanced representation of a protected attribute (gender in our experiments). This is done by first calculating FaiRR scores as the sum over the neutrality scores of the top retrieved documents, weighted by their ranking positions. The NFaiRR metric provides comparable results across queries by normalizing the per-query FaiRR scores over the ideal FaiRR inferred from a background set of documents (e.g. the top documents retrieved by a baseline BM25 model [12]). We calculate the NFaiRR metric with a cutoff at 10 for each bias-sensitive query, and report the average results over queries. The utility of the models is evaluated with common metrics for MS MARCO (which defines relevance in a binary fashion and is a sparsely annotated collection, most often with a single relevant passage per query), namely mean reciprocal rank (MRR) and Recall, both at cutoff 10.

### 3.3.4 Training, Model Selection and Hyperparameters

When tuning the intensity of bias mitigation, we need to train a model for each value of the regularization coefficient or adversarial factor. Which time-dependent model instance (checkpoint) should we choose as "best", in order to evaluate the method's performance?

Since there is a trade-off between utility and fairness, we follow the principled approach proposed by Rekabsaz et al. [12]: we max-min normalize MRR and NFaiRR to a range between 0 and 1, and choose the instance where their harmonic mean (F1 score) is maximum over the entire training session. In the case of adversarial methods, because validation performance fluctuates persistently during training, it is not clear when to stop training. By contrast, CODER shows a smooth convergence behavior (see Fig.3.4) and in practice would benefit from stopping criteria based on the relative improvement of metrics. However, to avoid giving this advantage to CODER, while still reflecting practical concerns for model training and selection, we fix the maximum training time of all models to a value that we estimated to be sufficient for each model to achieve its "best" performance (as defined above), after running a few tentative training sessions. Consequently, regardless of the regularization/adversarial factor, this value was set to 10 hours for CODER, 12 hours for AdvTAS-B, and 8 days for

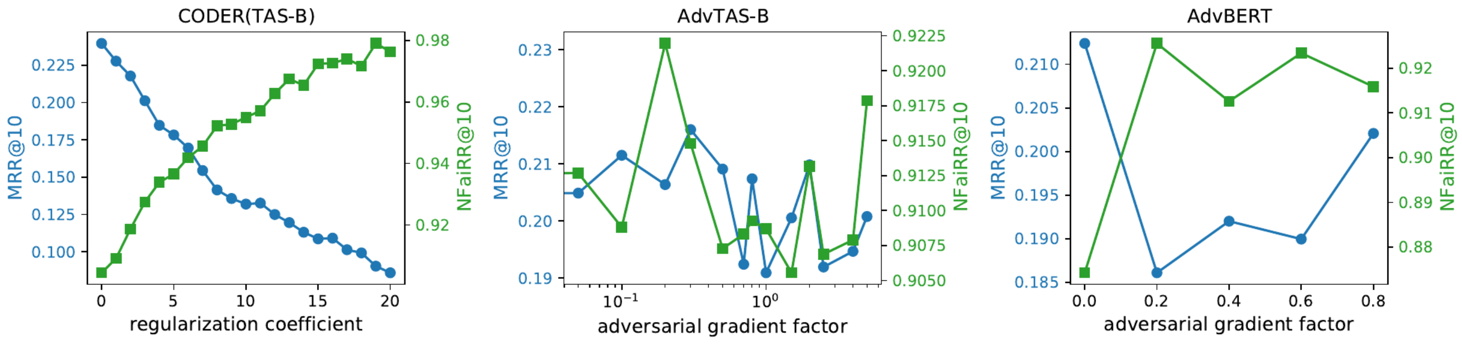


Figure 3.3: Modulation of utility and fairness by controlling the regularization factor (CODER) or the factor of adversarial gradient (AdvBERT, AdvTAS-B). Regularization with CODER allows to select the utility vs. fairness trade-off in a finely controllable and predictable manner.

AdvBERT[1].

We nevertheless note that, unlike in the case of the adversarial models, in the case of CODER the "best" performance was most often reported at or close to the very end of training, which indicates that its maximum performance is likely underestimated and that it would benefit from a training time dependent on the regularization coefficient.

To train CODER, we use the same hyperparameters as in Chapter 1, but increase batch size from 32 to 64 to accelerate convergence.

## 3.4 Results

Figure 3.2 depicts how retrieval performance changes in terms of utility and fairness/neutrality (measured as described in Section 3.3 in terms of MRR@10 and NFaiRR@10 respectively) as we progressively increase the intensity of bias mitigation (shown by the intensity of marker color), starting from 0. Compared to the adversarial baselines, it is evident that regularization with CODER allows modulating fairness to much higher values. Importantly, our method yields an approximately linear trade-off between utility and fairness and allows finely controlling it through the regularization coefficient $\lambda_r$. This is more directly shown in Figure 3.3. By contrast, in the case of adversarial training, although fairness can be increased to some extent, the dependence of utility and fairness on the adv. gradient factor $\lambda_a$ is complicated and unpredictable. It is thus difficult to select a desired point in the trade-off, and the corresponding evaluations appear as a disorderly point cloud on Figure 3.2.

Furthermore, for the same values of fairness, Figure 3.2 shows that CODER can achieve substantially higher utility (evaluation points lie higher and to the right of all baseline points). Given that AdvBERT is the hitherto state-of-the-art method for this task and dataset, our neutrality regularization method based on CODER therefore achieves the new state-of-the-art performance.

[1]AdvBERT requires a much longer time because it is a model based on much slower self-attention over the concatenated query and document, and because it is fine-tuned from the standard BERT-Mini, as opposed to TAS-B, an encoder already pretrained for retrieval on MS MARCO.

| | | F1 | NFaiRR@10 | MRR@10 | Recall@10 |
|---|---|---|---|---|---|
| Mean | CODER(TAS-B) | **0.356** | 0.915 | 0.222 | 0.429 |
| | AdvTAS-B | 0.351 | 0.911 | 0.217 | 0.381 |
| | AdvBERT | 0.318 | 0.919 | 0.193 | 0.402 |
| Median | CODER(TAS-B) | **0.358** | 0.914 | 0.223 | 0.428 |
| | AdvTAS-B | 0.346 | 0.911 | 0.214 | 0.386 |
| | AdvBERT | 0.316 | 0.920 | 0.191 | 0.401 |
| Max | CODER(TAS-B) | **0.379** | 0.927 | 0.240 | 0.450 |
| | AdvTAS-B | 0.348 | 0.915 | 0.231 | 0.394 |
| | AdvBERT | 0.331 | 0.926 | 0.202 | 0.435 |

Table 3.1: Comparison of ranking performance based on top 4 F1 scores within the NFaiRR range [0.9 - 0.93].

In the limited range of NFAiRR@10 $\in [0.90, 0.93]$ we find the top 4 points in terms of F1 scores (harmonic mean between MRR@10 and NFaiRR@10) for each method and display statistics of performance metrics in Table 3.1. NFaiRR itself is provided only as a reference in this table, since it acts as the control parameter and the range is merely chosen to have overlapping values.

Besides unpredictability with respect to the bias mitigation control factor, the performance of models undergoing adversarial training fluctuates haphazardly during training and it is thus very difficult to know whether the model has reached peak performance in order to stop training. In practice, one resorts to using a fixed training time. By contrast, in Figure 3.4 we observe that regularization with CODER shows smooth convergence patterns and allows setting a stopping criterion based on monitoring performance, which in turn allows an adaptable training time for each regularization coefficient and can yield better performance (although for the sake of comparison we didn't use this technique in this work).

Finally, we note that Rekabsaz et al. [12] also introduced TRECDL$_{\text{FAIR}}$, a subset of the TREC Deep Learning Track 2019 queries, but discovered that this set is not very challenging, and all methods they examined performed well. Indeed, we find that CODER(TAS-B) with $\lambda_r = 1$ can attain an MRR@10 score of $1.0$ at a NFaiRR@10 score of 0.967, and when boosting NFaiRR@10 to 0.985 with $\lambda_r = 8$, MRR@10 only drops to 0.944, which is a stronger performance than all reported methods in Rekabsaz et al. [12].

## 3.5 Conclusion

In this chapter, we have introduced a novel method for reducing bias in search results, which is based on directly imposing a neutrality regularization loss to the documents most highly scored for relevance. To achieve this, we leveraged a contextual document embedding reranking framework, which, for the same query, jointly scores a large set of retrieved candidate documents that together constitute a retrieval context. We demonstrated that our method can lead to much stronger bias mitigation/fairness compared to the existing alternatives for

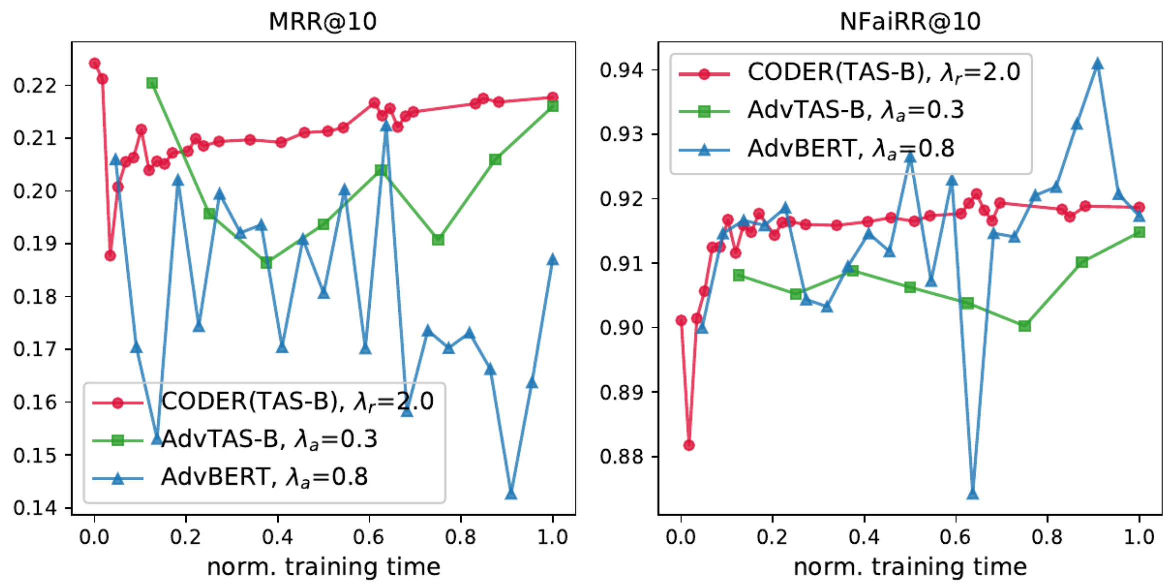


Figure 3.4: Utility (left) and fairness (right) on `MSMARCO` dev during training, for a particular regularization or adversarial factor, chosen such that the models perform comparably. Training times are significantly shorter for CODER and thus the horizontal axis is normalized in the same range.

deep neural retrieval architectures, which are based on adversarial training. At the same time, it achieves the state-of-the-art performance with respect to utility (relevance) for the same amount of bias mitigation. Finally, our method allows for a more finely controllable and predictable intensity of bias mitigation, which is of paramount importance with respect to widespread adoption.

# Bibliography

[1] Matthew Kay, Cynthia Matuszek, and Sean A Munson. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3819–3828, 2015.

[2] Jahna Otterbacher, Jo Bates, and Paul Clough. Competent men and warm women: Gender stereotypes and backlash in image search results. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 6620–6631, 2017.

[3] Navid Rekabsaz and Markus Schedl. Do neural ranking models intensify gender bias? In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2065–2068, 2020.

[4] Ruoyuan Gao and Chirag Shah. Toward creating a fairer ranking in search engine results. *Information Processing & Management*, page 102138, 2020.

[5] Le Chen, Ruijun Ma, Anikó Hannák, and Christo Wilson. Investigating the impact of gender on rank in resume search engines. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.

[6] Robert Epstein and Ronald E. Robertson. The search engine manipulation effect (SEME) and its possible impact on the outcomes of elections. *Proceedings of the National Academy of Sciences*, 112(33):E4512–E4521, August 2015. doi: 10.1073/pnas.1419828112. URL https://www.pnas.org/doi/10.1073/pnas.1419828112. Publisher: Proceedings of the National Academy of Sciences.

[7] Tim Draws, Nava Tintarev, Ujwal Gadiraju, Alessandro Bozzon, and Benjamin Timmermans. This Is Not What We Ordered: Exploring Why Biased Search Result Rankings Affect User Attitudes on Debated Topics. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 295–305. Association for Computing Machinery, New York, NY, USA, July 2021. ISBN 978-1-4503-8037-9. URL https://doi.org/10.1145/3404835.3462851.

[8] Frances A. Pogacar, Amira Ghenai, Mark D. Smucker, and Charles L.A. Clarke. The Positive and Negative Influence of Search Results on People's Decisions about the Efficacy of Medical Treatments. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 209–216, New York, NY, USA,

October 2017. Association for Computing Machinery. ISBN 978-1-4503-4490-6. doi: 10.1145/3121050.3121074. URL https://doi.org/10.1145/3121050.3121074.

[9] Ryen W. White and Eric Horvitz. Belief Dynamics and Biases in Web Search. *ACM Transactions on Information Systems*, 33(4):18:1–18:46, May 2015. ISSN 1046-8188. doi: 10.1145/2746229. URL https://doi.org/10.1145/2746229.

[10] Leif Azzopardi. Cognitive Biases in Search: A Review and Reflection of Cognitive Biases in Information Retrieval. In *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, CHIIR '21, pages 27–37, New York, NY, USA, March 2021. Association for Computing Machinery. ISBN 978-1-4503-8055-3. doi: 10.1145/3406522.3446023. URL https://doi.org/10.1145/3406522.3446023.

[11] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 429–438. ACM, 2020. doi: 10.1145/3397271.3401100. URL https://doi.org/10.1145/3397271.3401100.

[12] Navid Rekabsaz, Simone Kopeinik, and Markus Schedl. Societal biases in retrieved contents: Measurement framework and adversarial mitigation for bert rankers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[13] Meike Zehlike and Carlos Castillo. Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of The Web Conference*, pages 2849–2855, 2020.

[14] Amin Bigdeli, Negar Arabzadeh, Shirin Seyedsalehi, Morteza Zihayat, and Ebrahim Bagheri. On the orthogonality of bias and utility in ad hoc retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1748–1752, 2021.

[15] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 405–414, 2018.

[16] Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[17] Fernando Diaz, Bhaskar Mitra, Michael D. Ekstrand, Asia J. Biega, and Ben Carterette. Evaluating stochastic rankings with expected exposure. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, page 275–284, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3411962. URL https://doi.org/10.1145/3340531.3411962.

[18] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]*, October 2018. URL http://arxiv.org/abs/1611.09268. arXiv: 1611.09268.

[19] Klara Krieg, Emilia Parada-Cabaleiro, Gertraud Medicus, Oleg Lesota, Markus Schedl, and Navid Rekabsaz. Grep-biasir: A dataset for investigating gender representation-bias in information retrieval results. *arXiv preprint arXiv:2201.07754*, 2022.

[20] Klara Krieg, Emilia Parada-Cabaleiro, Markus Schedl, and Navid Rekabsaz. Do perceived gender biases in retrieval results affect relevance judgements? In *Proceedings of the European Conference on Information Retrieval, Workshop on Algorithmic Bias in Search and Recommendation (ECIR-BIAS 2022)*, 2022.

[21] Ashudeep Singh and Thorsten Joachims. Policy learning for fairness in ranking. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5427–5437, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/9e82757e9a1c12cb710ad680db11f6f1-Abstract.html.

[22] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228, 2018.

[23] Harrie Oosterhuis. Computationally efficient optimization of plackett-luce ranking models for relevance and fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[24] Daniel Cohen. Allowing for the grounded use of temporal difference learning in large ranking models via substate updates. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 438–448, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. URL https://doi.org/10.1145/3404835.3462952.

[25] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv:2006.15498 [cs]*, July 2020. URL http://arxiv.org/abs/2006.15498. arXiv: 2006.15498.

[26] Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2513. Association for Computing Machinery, New York, NY, USA, July 2021. ISBN 978-1-4503-8037-9. URL https://doi.org/10.1145/3404835.3463242.

[27] Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *arXiv:2004.12832 [cs]*, June 2020. URL http://arxiv.org/abs/2004.12832. arXiv: 2004.12832.

[28] Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. A modern perspective on query likelihood with deep generative retrieval models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 185–195, 2021.

[29] Daniel Cohen, Bhaskar Mitra, Katja Hofmann, and W. Bruce Croft. Cross domain regularization for neural ranking models using adversarial learning. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1025–1028, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210141. URL https://doi.org/10.1145/3209978.3210141.

[30] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems*, 33:21476–21487, 2020.

[31] Christian Ganhör, David Penz, Navid Rekabsaz, Oleg Lesota, and Markus Schedl. Mitigating consumer biases in recommendations with adversarial training. In *Proceedings of the 45th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2022*. ACM, 2022.

[32] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and A. Hanbury. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *SIGIR*, 2021. doi: 10.1145/3404835.3462891.

[33] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *arXiv:2010.08191 [cs]*, May 2021. URL http://arxiv.org/abs/2010.08191. arXiv: 2010.08191.

[34] Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, 2021. doi: 10.18653/v1/2021.findings-acl.191. URL http://arxiv.org/abs/2108.06027. arXiv: 2108.06027.

[35] Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *arXiv:1901.04085 [cs]*, April 2020. URL http://arxiv.org/abs/1901.04085. arXiv: 1901.04085.

[36] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain

Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

[37] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, Dense, and Attentional Representations for Text Retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, April 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00369. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00369/100684/Sparse-Dense-and-Attentional-Representations-for.

[38] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*, October 2020. URL http://arxiv.org/abs/2007.00808. arXiv: 2007.00808.

[39] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada, July 2021. ACM. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462880. URL https://dl.acm.org/doi/10.1145/3404835.3462880.

[40] George Zerveas, Navid Rekabsaz, Daniel Cohen, and Carsten Eickhoff. Coder: An efficient framework for improving retrieval through contextual document embedding reranking, 2021.

[41] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv:1910.01108 [cs]*, February 2020. URL http://arxiv.org/abs/1910.01108. arXiv: 1910.01108.

[42] Fabio Crestani, Mounia Lalmas, Cornelis J. Van Rijsbergen, and Iain Campbell. “is this document relevant?…probably”: A survey of probabilistic models in information retrieval. *ACM Comput. Surv.*, 30(4):528–552, dec 1998. ISSN 0360-0300. doi: 10.1145/299917.299920. URL https://doi.org/10.1145/299917.299920.

[43] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’17, pages 1253–1256, New York, NY, USA, August 2017. Association for Computing Machinery. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080721. URL https://doi.org/10.1145/3077136.3080721.

[44] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML ’07, pages 129–136, New York, NY, USA,

June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

[45] Navid Rekabsaz, Robert West, James Henderson, and Allan Hanbury. Measuring societal biases from text corpora with smoothed first-order co-occurrence. In *Proceedings of the Fifteenth International AAAI Conference on Web and Social Media, ICWSM 2021, held virtually, June 7-10, 2021*, pages 549–560. AAAI Press, 2021.

[46] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning*, pages 1180–1189. PMLR, 2015.

[47] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models, 2019.

# Chapter 4

# Enhancing the ranking context of dense retrieval through Reciprocal Nearest Neighbors

## 4.1 Introduction

### 4.1.1 Addressing the problem of sparse annotation

The training of state-of-the-art ad-hoc text retrieval models [1, 2, 3, 4, 5, 6, 7, 8], which are based on pre-trained transformer Language Models, relies on large-scale datasets that are sparsely annotated, typically comprising only a small number of relevance judgements for each query.[1] These labels are usually derived from submitting the strongest pseudo-relevance signals in user click logs to human judges for verification. Despite potential future endeavors to extend annotation, this sparsity and the resulting issue of false negatives [10, 11] – i.e., only a minuscule fraction of all documents pertinent to a query are ever seen by users or judges and identified as relevant – will inevitably persist. To eliminate the sparsity, it would be necessary to acquire either human judgements, or perhaps expensive evaluations by Large Language Models, to verify the relevance of the *entire* document collection (typically tens of millions of documents) with respect to *every* query in the dataset, leading to an intractable Cartesian product. Consequently, it is crucial to explore optimizing the utilization of existing information, and extract richer structural relationships between documents and queries, without additional annotations.

To this end, in the present work we follow a two-pronged approach: first, we employ the concept of *reciprocal nearest neighbors* (rNN) to improve the estimation of semantic similarity between embeddings of queries and documents. Two documents $c_i$ and $c_j$ are said to be $k$-reciprocal nearest neighbors if $c_j$ is within the $k$-nearest neighbors of $c_i$, and at the same time $c_i$ is within the $k$-nearest neighbors of $c_j$. Second, we attempt to enhance the query-specific *ranking context* used to train dense retrievers, going beyond the notion of using mined candidates merely as negatives for contrastive learning. Specifically, we use the

[1]E.g., on average 1.06 documents per query in MS MARCO [9].

similarity of ground-truth documents to candidates in the same ranking context as the query as evidence to guide the model's predicted relevance probability distribution over candidates.

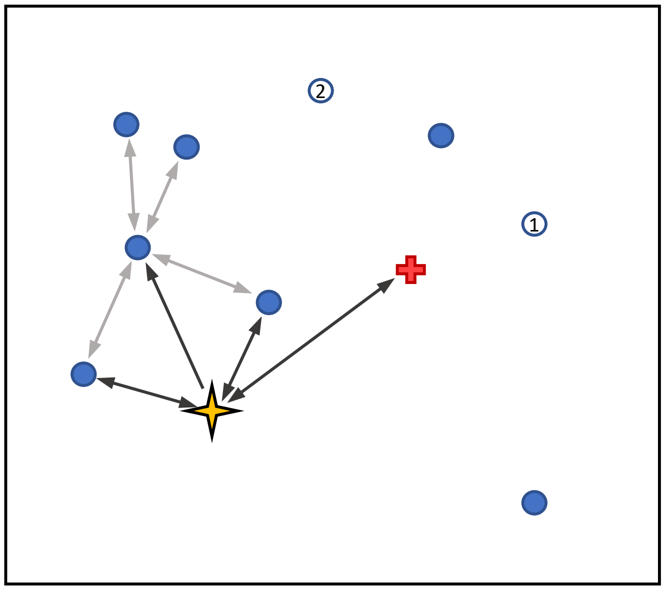


Figure 4.1: Query (yellow star), positive (red cross) and negative (full blue circles) document embedding vectors in a shared 2D representation space. Based on top-4 Nearest Neighbors, the positive would be ranked lower than the 3 nearest neighbors of the query. When using top-4 Reciprocal Nearest Neighbors, its ranking is improved, because of its reciprocal relationship to the query, which one of 3 nearest neighbors of the query lacks. Adding an extra negative to the context (circle #1) does not affect this ranking, but the second extra negative (#2) disrupts the reciprocal relationship, becoming the 4th nearest neighbor of the positive.

### 4.1.2 Limitations of contemporary dense retrieval

Dense retrieval, the state-of-the-art approach for single-stage ad-hoc retrieval, is premised on modeling relevance between a query and a document as the geometric proximity (e.g., dot-product or cosine similarity) between their respective embeddings in the common representation vector space. Top retrieval results are therefore the documents whose embeddings are the nearest neighbors of the query embedding. However, this modeling assumption may be sub-optimal: previous work in the field of image re-identification has shown that, while geometric similarity can easily differentiate between candidate embeddings in near proximity from a query embedding, the differences between relevance scores of candidate embeddings become vanishingly small as distance from the query increases [12]. It was found that the degree of overlap between sets of reciprocal nearest neighbors can be used to compute an improved measure of similarity between query and candidate embeddings [13].

Moreover, geometric similarity is used in mining "hard" negatives, which have been consistently found to improve performance compared to random in-batch negatives [3, 10, 14, 15]. Hard negatives are typically the top-ranked candidates retrieved by a dense retriever (nearest neighbors to a query embedding) that are not explicitly annotated as relevant in the dataset.

On the one hand, the effectiveness of mined negatives is limited by how effectively this dense retriever can already embed queries and relevant documents in close proximity within the shared representation space, although the periodical or dynamic retrieval of negatives during training can partially alleviate this problem [3, 14]. On the other hand, when the retriever used to mine hard negatives indeed succeeds in retrieving candidates that are semantically relevant to the query, these are often not marked as positives due to the sparsity of annotation and are thus spuriously used as negatives for contrastive learning (false negatives)[2], confounding the training signal [10, 11].

### 4.1.3 The contribution of this work

In this work we investigate to what degree these issues can be mitigated through the use of reciprocal nearest neighbors, essentially extracting additional relationship information between queries and documents beyond flat geometric distances, such as the local degree of node connectivity. Furthermore, unlike all existing dense retrieval methods, instead of using candidates exclusively as negatives, we propose using their estimated similarity to the ground-truth document(s) as evidence for label smoothing; we thus redistribute probability weight in the target score distribution from the ground truth to a larger number of likely false negatives.

Finally, our work places a strong emphasis on computational efficiency: label smoothing can be performed entirely offline on CPUs and can be trivially parallelized, while no latency is introduced during training and our models can be trained (e.g., on MS MARCO) within hours, using a single GPU with a batch size of 32. Reranking based on reciprocal nearest neighbors, when used, introduces a few milliseconds latency per query on a CPU.

By contrast, the current state-of-the-art dense retrieval methods (e.g. [4, 10]) depend on the existence of better performing, but computationally demanding re-ranking models such as cross-encoders, which are typically run offline on several GPUs with huge batch sizes and are used either for pseudo-labeling additional training data, for discarding negatives which are likely unlabeled positives (i.e., false negatives), or directly for distillation through a teacher-student training scheme. However, besides the very high computational cost of such pipelines, the existence of a model that is more powerful than the retrieval model we wish to train is a very restrictive constraint, and cannot be taken for granted in many practical settings.

**Synopsis of contributions:**

- We propose evidence-based label smoothing, a novel method which mitigates the problem of false negatives by leveraging the similarity of candidate documents *within the ranking context of a query* to the annotated ground truth in order to compute soft relevance labels. Different from existing methods like teacher-student distillation or

[2]Qu et al. [10] estimate that about 70% of the top 5 candidates retrieved by a top-performing dense retrieval model that are not labeled as positive are actually relevant.

pseudo-labeling, our approach does not rely on the existence of more powerful retrieval methods.

- We explore the applicability of the concept of reciprocal nearest neighbors in improving the similarity metric between query and document embeddings in the novel setting of ad-hoc text retrieval. Its applicability is far from guaranteed, because this setting substantially differs from the previously proposed setting for image re-identification, beyond a simple change of input modality: queries and documents are entities with very distinct characteristics from one another, differing in length, style, vocabulary, and intent. Essentially, we wish to model relevance and not the invariance of the same entity under different photographic conditions.

- Through extensive experiments on two different large-scale ad-hoc retrieval datasets, we demonstrate that the concept of reciprocal nearest neighbors can indeed enhance the ranking context in a computationally efficient way, both when reranking candidates at inference time, as well as when applied for evidence-based label smoothing intended for training.

- We find that besides degrading training, false negatives also pose challenges for evaluation, affecting performance bench-marking. Specifically, We find that in sparsely annotated datasets like MS MARCO, validation loss may be a better predictor of model generalization than IR metrics such as MRR, and that evaluation on datasets with higher annotation depth (such as TREC DL), as well as zero-shot evaluation, can better reflect the ranking effectiveness of models.

## 4.2 Related work

Our proposed label smoothing, which encourages the model to assign higher relevance scores to documents intimately related to the ground truth, conceptually finds support in prior work that proposed *local relevance score regularization* [16], adjusting retrieval scores to respect local inter-document consistency. Despite the entirely different methodology, both methods are premised on the intuition that documents lying closely together in the representation vector space should have similar scores; this in turn is related to the *cluster hypothesis*, which states that closely related documents (and thus proximal in terms of vector representations) tend to be relevant to the same request [17].

In Chapter 2 we argued that jointly scoring a large number of candidate documents (positives and negatives) closely related to the same query within a list-wise loss constitutes a query-specific ranking context that benefits the assessment of relevance of each individual candidate document with respect to the query. We thus extended well-established insights and empirical findings from Learning-to-Rank literature [18, 19, 20] to the realm of dense retrieval through transformer-based Language Models. While in-depth annotation of candidate documents (i.e., hundreds of relevance judgements per query) explicitly provides a rich context for each query in Learning-to-Rank datasets [21, 22, 23], such information is not available in the sparsely annotated, large-scale datasets used to train dense retrieval models.

The relationship exploited thus far to “build a context” (practically, this means mining hard negatives), is simply that of geometric proximity between the embeddings of a query and candidate documents.

Addressing the problem of sparse annotation, several works have utilized the relevance estimates from supervised (e.g. 4, 10, 24) or unsupervised (e.g. lexical: 25, 26) retrieval methods or other dataset-specific heuristics (e.g. bibliography citations: 27) to derive soft labels for documents used to train a model, e.g., in a teacher-student distillation scheme. In this work, we instead shift the perspective from assigning labels based on similarity *with respect to the query*, to similarity *with respect to the ground-truth document(s)*, but within a query-specific ranking context. We furthermore leverage the concept of reciprocal nearest neighbors, introduced as a reranking method for image re-identification [12, 13], to improve the similarity estimate.

False negatives have been identified as a significant challenge by prior work, which has employed powerful but computationally expensive cross-encoders [1] to discard documents that receive a high similarity score to the query and are thus likely relevant from the pool of hard negatives [4, 10]. However, discarding top-ranking hard negatives also discards potentially useful information for training.

Recently, Zhou et al. [11] tackled the problem of false negatives through selective sampling of negatives around the rank of the ground-truth document, avoiding candidates that are ranked either much higher than the ground truth (probable false negatives) or much lower (too easy negatives). This approach differs from ours in the perspective of similarity (query-centric vs ground-truth-centric), and in the fact that information is again discarded from the context, as only a small number of negatives is sampled around the positive. Additionally, a query latency of up to 650 ms is added during training.

Ren et al. [5] leverage the similarity of candidate documents to the ground truth document (positive), but in a different way and to a different end compared to our work: all documents in the batch (“in-batch negatives”) as well as retrieved candidates are used as negatives in an InfoNCE loss term, which penalizes the model when it assigns a low similarity score between a single positive and the query compared to the similarity score it assigns to pairs of this positive with all other candidates. Thus, it requires that the ground truth lies closer to the query than other candidates, but the detrimental effect of false negatives on the training signal fully persists.

By contrast, our method jointly takes into account all positives and other candidates in the ranking context, and through a KL-divergence loss term requires that the predicted relevance of the query with respect to all documents in the ranking context has a similar probability distribution to the target distribution, i.e., the distribution of similarity between all ground truth positives and all candidate documents in the context. False negatives are thus highly likely to receive a non-zero probability in the target distribution, and the penalty when assigning a non-zero relevance score to false negatives is lower.

## 4.3 Methods

### 4.3.1 Similarity metric based on Reciprocal Nearest Neighbors

Nearest Neighbors are conventionally retrieved based on the geometric similarity (here, inner product) between embedding vectors of a query $q$ and candidate document $c_i$: $s(q, c_i) = \langle x_q, x_{c_i} \rangle$, with $x_q = m(q)$ and $x_{c_i} = m(c_i)$ embeddings obtained by a trained retrieval model $m$. In this section, we additionally define the Jaccard similarity $s_J$ that measures the overlap between the sets of *reciprocal neighbors* of $q$ and $c_i$. We provide a derivation of $s_J$ below.

Let $\mathcal{C}$ be a collection of documents, including the query used for search, and $\text{NN}(q, k)$ denote the set of $k$-nearest neighbors of a probe $q \in \mathcal{C}$ – besides the query, $q$ here can also be a document or any other element that can be embedded in the common representation space. If $d(q, c_i) \equiv d_g(\mathbf{x}_q, \mathbf{x}_{c_i}),\ c_i \in \mathcal{C}$ is a metric (distance) in the vector space within which the embeddings of the query $\mathbf{x}_q$ and documents $\mathbf{x}_i$ reside, we can formally write:

$$\text{NN}(q, k) = \{c_i \mid d_g(\mathbf{x}_{c_i}, \mathbf{x}_q) \leq d_g(\mathbf{x}_{c_k}, \mathbf{x}_q)\},\ \forall i \in \mathbb{N} : 1 \leq i \leq |\mathcal{C}|, \tag{4.1}$$

where $|\cdot|$ denotes the cardinality of a set, and document $c_k$ is the $k$-nearest neighbor of the query based on $d$, i.e., the $k$-th element in the list of all documents in $\mathcal{C}$ sorted by distance $d$ from the query in ascending order[3]. Naturally, $|\text{NN}(q, k)| = k$.

The set of $k$-reciprocal nearest neighbors can then be defined as:

$$\mathcal{R}(q, k) = \{c_i \mid c_i \in \text{NN}(q, k) \wedge q \in \text{NN}(c_i, k)\}, \tag{4.2}$$

$$\mathcal{R}(c_j, k) = \{c_i \mid c_i \in \text{NN}(c_j, k) \wedge c_j \in \text{NN}(c_i, k)\} \tag{4.3}$$

i.e., to be considered a $k$-reciprocal neighbor, a document must be included in the $k$-nearest neighbors of the query, but at the same time the query must also be included in the $k$-nearest neighbors of the same document. This stricter condition results in a stronger similarity relationship than simple nearest neighbors, and $|\mathcal{R}(q, k)| \leq k$.

Since using the above definition as-is can be overly restrictive, prior work has proposed applying it iteratively in order to construct an extended set of highly related documents to the query that would have otherwise been excluded. Thus, Zhong et al. [13] define the extended set:

$$\begin{aligned} \mathcal{R}^*(q, k) &\coloneqq \mathcal{R}(q, k) \cup \mathcal{R}(c_i, \tau k), \\ \text{s.t. } \big|\mathcal{R}(q, k) \cap \mathcal{R}(c_i, \tau k)\big| &\geq \frac{2}{3}\big|\mathcal{R}(c_i, \tau k)\big|, \\ &\forall c_i \in \mathcal{R}(q, k). \end{aligned} \tag{4.4}$$

Effectively, we examine the set of $\tau k$-nearest reciprocal neighbors of each reciprocal neighbor of $q$ (where $\tau \in [0, 1]$ is a real parameter), and provided that it already has a substantial overlap with the original set of reciprocal neighbors of $q$, we add it to the extended set. The

[3]When some measure of similarity $s$ is used instead of a distance $d$, the relationship equivalently becomes: $s(\mathbf{x}_{c_i}, \mathbf{x}_q) \geq s(\mathbf{x}_{c_k}, \mathbf{x}_q)$, and the $k$-nearest neighbors are the first $k$ documents sorted by $s$ in descending order.

underlying assumption is that if a document is closely related to a set of documents that are closely related to the query, then it is most likely itself related to the query, even if there is no direct connection in terms of geometric proximity. Thus, one can improve recall at the possible expense of precision.

Although using this new set of neighbors as the new set of candidates and sorting them by their distance $d$ can form the basis of a retrieval method, Zhong et al. [13] additionally proceed to define a new distance that takes into account this set, which is used alongside $d$. Specifically, they use the Jaccard distance between the (extended) reciprocal neighbor sets of a query $q$ and documents $c_i$:

$$d_J(q, c_i) = 1 - \frac{\left|\mathcal{R}^*(q,k) \cap \mathcal{R}^*(c_i,k)\right|}{\left|\mathcal{R}^*(q,k) \cup \mathcal{R}^*(c_i,k)\right|}. \tag{4.5}$$

This distance quantifies similarity between two elements (here, $q$ and $c_i$) as a measure of overlap between sets of neighbors robustly related to each of them.

To reduce the computational complexity of computing the Jaccard distance, which relies on the time-consuming, CPU-bound operations of finding the intersection and union of sets, one may instead carry out the computation with algebraic operations, by defining for each element $q \in \mathcal{C}$ sparse vectors of dimensionality $|\mathcal{C}|$, where non-zero dimensions denote graph connectivity to other documents. Instead of using binary vectors, one may assign to each neighbor $c_i$ a weight that depends on its geometric distance to the probe $q$. Thus, following Zhong et al. [13], we define the elements of reciprocal connectivity vectors $\mathbf{v}'_q \in |\mathcal{C}|$ as follows:

$$v'_{q,c_i} = \begin{cases} f_w\left(d(q,c_i)\right) & \text{if } c_i \in \mathcal{R}^*(q,k) \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

While Zhong et al. [13] exclusively use $f_w(x) = \exp(-x)$, one one can use any monotonically decreasing function, and we found that $f_w(x) = -x$ in fact performs better in our experiments.

Instead of directly using the sparse vectors above, which would yield a discretized similarity metric, we additionally perform a local expansion, mixing each one of them (including the query) with its $k_{\text{exp}}$ neighboring vectors (again including the query, if among the neighbors):

$$\mathbf{v}_{c_i} = \frac{1}{k_{\text{exp}}} \sum_{j=1}^{k_{\text{exp}}} \mathbf{v}'_{c_j} \ , \ \forall c_j \in \text{NN}(c_i, k_{\text{exp}}) \ . \tag{4.7}$$

It is possible to use the element-wise $\min$ and $\max$ operators on the expanded sparse vectors from Eq. (4.7)) to compute the number of candidates in the intersection and union sets of Eq. (4.5) respectively as:

$$\left|\mathcal{R}^*(q,k) \cap \mathcal{R}^*(c_i,k)\right| = \sum \min(\mathbf{v}_q, \mathbf{v}_{c_i}) \tag{4.8}$$

$$\left|\mathcal{R}^*(q,k) \cup \mathcal{R}^*(c_i,k)\right| = \sum \max(\mathbf{v}_q, \mathbf{v}_{c_i}), \tag{4.9}$$

and thus the Jaccard distance in Eq. (4.5) can be written as:

$$d_J(q, c_i) = 1 - \frac{\sum_{j=1}^{|\mathcal{C}|} \min(v_{q,c_j}, v_{c_i,c_j})}{\sum_{j=1}^{|\mathcal{C}|} \max(v_{q,c_j}, v_{c_i,c_j})}. \tag{4.10}$$

Equivalently, Jaccard similarity can be therefore written as:

$$s_J(q, c_i) = 1 - s_J(q, c_i) = \frac{\sum_{j=1}^{|\mathcal{C}|} \min(v_{q,c_j}, v_{c_i,c_j})}{\sum_{j=1}^{|\mathcal{C}|} \max(v_{q,c_j}, v_{c_i,c_j})}. \tag{4.11}$$

Finally, instead of the pure Jaccard similarity $s_J$, we use a linear mixture with the geometric similarity $s$ controlled by hyperparameter $\lambda \in [0, 1]$:

$$s^*(q, c_i) = \lambda\, s(q, c_i) + (1 - \lambda)\, s_J(q, c_i), \tag{4.12}$$

which we found to perform better both for reranking (as in 13), as well as for label smoothing.

Importantly, unlike prior work [12, 13], which considered the entire gallery (collection) of images as a *reranking context* for each probe, we only use as a context a limited number of candidates previously retrieved for each query. This is done both for computational tractability, as well as to constrain the context to be query-specific when computing the similarity of documents to the ground truth; documents can otherwise be similar to each other with respect to many different topics unrelated to the query. We empirically validate this choice in Section 4.5.1.

### 4.3.2 Evidence-based label smoothing

The full procedure for evidence-based label smoothing is given in Algorithm 1, and in the following sections we provide a rationale for the method in detail.

#### Main principles

Although training with a KL-divergence loss and a large number of (positive and negative) candidates per query may provide some robustness with respect to false negatives (see Section 2.5.5), it would be still beneficial to identify the unlabeled positives among the mined negatives, in order to rectify the signal for learning similarity. Given that for approx. 95% of all queries in MS MARCO the ground-truth relevance distribution over all candidates is a 1-hot vector, in order to match it with the estimated score distribution, the loss pushes the model to assign all probability mass on a single relevant document, which is likely counterproductive and mis-calibrates relevance scores.

Label noise, and especially false negatives, exist in most datasets. *Uniform* label smoothing is a well-established technique [28] that is used to mitigate the effects of label noise and improve score calibration, and was recently also employed for contrastive learning [29]. It involves removing a small proportion $\epsilon \in [0, 1]$ of the probability mass corresponding to the ground-truth class and uniformly redistributing it among the rest of the classes, thus converting, e.g., a 1-hot vector $\mathbf{y} = [1, 0, \ldots, 0] \in \mathbb{R}^N$ to:

$$\mathbf{y}^* = [1 - \epsilon,\ \epsilon/(N - 1), \ldots, \epsilon/(N - 1)] \in \mathbb{R}^N \tag{4.13}$$

Nevertheless, naively distributing the probability mass $\epsilon$ uniformly among all candidates, as in Eq. (4.13), would result in true negatives predominantly receiving a portion of it, apart from the small number of false negatives[4].

For this reason, we instead propose correcting the sparse annotation vectors by selectively distributing relevance probability among negatives that are highly likely to be positive, or at least are ambiguous with respect to their relevance to the query. The proportion of probability mass each candidate shall receive depends on its degree of similarity to the annotated ground-truth document, which can be quantified by the Jaccard distance of Eq. (4.10), if we wish to exclusively consider reciprocal nearest neighbors, or the mixed geometric-Jaccard distance of Eq. (4.12), which allows any candidate close to the ground-truth to be considered.

### Target relevance score transformations

In standard contrastive learning, including when using a KL-divergence loss, as in CODER [15], there is a very stark difference between the probability of the handful of ground-truth documents and the zero probability of the negatives in the target (ground-truth) distribution.

In evidence-based label smoothing, we are using the continuous similarity scores of candidates with respect to the ground-truth positive document(s) as soft labels for training, which means that that there is a reduced contrast between the highest and smallest score values. Additionally, the output values of the model's similarity estimate reside within an arbitrary value range, determined primarily by the model's weights, and for the same rank, there is a large variance of values between queries (Fig. 4.2). This means that after passing through a softmax, which is highly non-linear, the target score distribution will be either concentrated or diffuse, depending on the range of score values for each particular query. Normalizing values into the same range will facilitate learning consistent relevance estimates. Furthermore, given a single query, we wish that target scores rapidly decrease as the rank increases (Fig. 4.3).

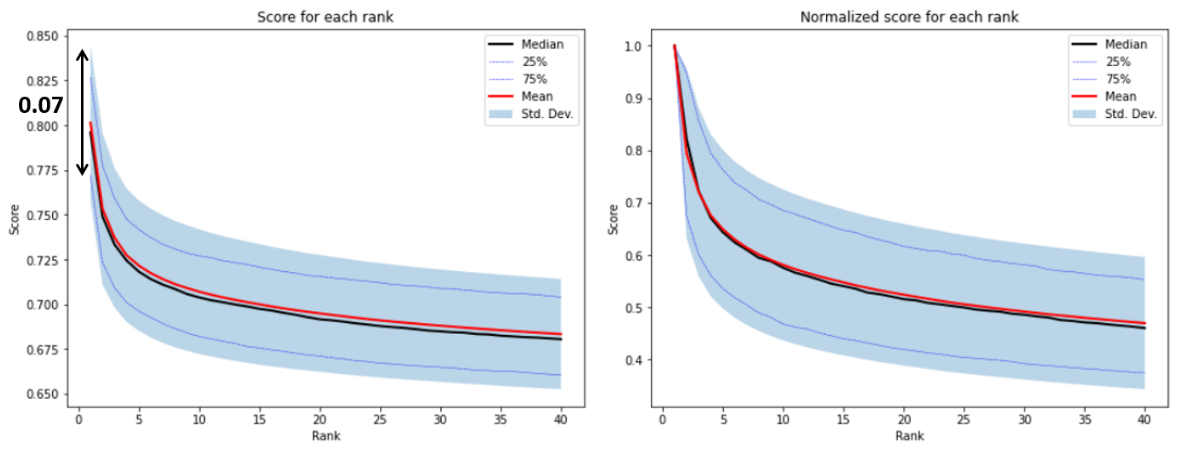


Figure 4.2: Similarity scores per rank across a large number of queries.

[4]Indeed, Qu et al. [10] observe that among mined "hard negative" candidates, the percentage of false negatives falls to 4% by rank 40.

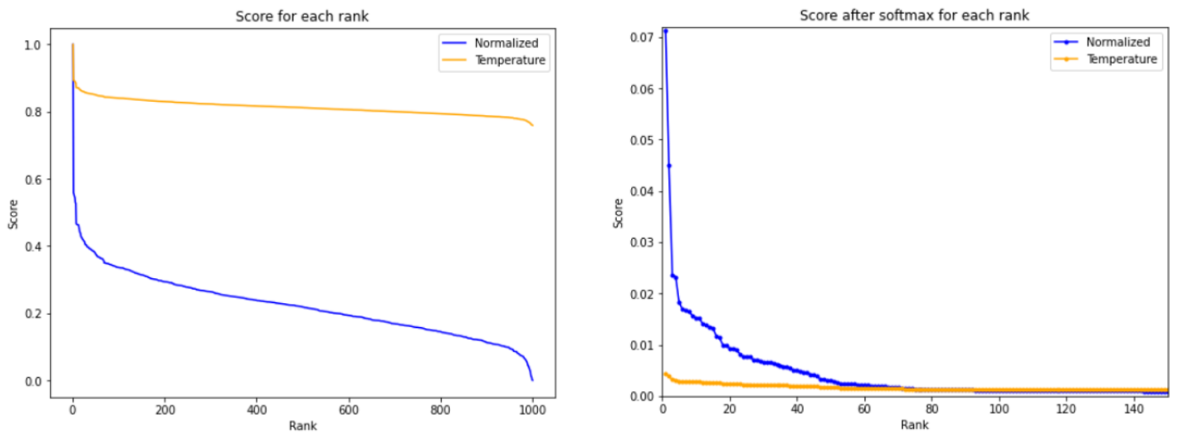


Figure 4.3: Similarity scores of the top 1000 candidates for a single query, sorted in descending order. Since they are used as training labels, to avoid very diffuse estimated score distributions, we need to ensure that there is a large contrast between the top and bottom candidates and that probability (i.e. values after the scores pass through a softmax) abruptly decreases after the first few ranks. We achieve this through appropriate normalization - here, max-min (blue) instead of dividing by max (orange).

Therefore, to facilitate learning, we wish to ensure that (a) there is large enough contrast between the first and last ranks, and (b) this is true for all queries. We can achieve this by applying a normalizing function $f_n$, such as max-min, on the vector $\mathbf{s} \in \mathbb{R}^N$ of candidate scores for a single query:

$$f_n(\mathbf{s}) = \frac{\mathbf{s} - \min(\mathbf{s})}{\max(\mathbf{s}) - \min(\mathbf{s})} \tag{4.14}$$

or the following, which is based on the standard deviation $\sigma$ across $N$ candidate scores for a single query:

$$f_n(\mathbf{s}) = \frac{\mathbf{s} - \min(\mathbf{s})}{\sigma}, \quad \sigma = \sqrt{\frac{\sum_i \left(s_i - \sum_j s_j/N\right)^2}{N}} \tag{4.15}$$

Therefore, since the value range of similarity scores that each model outputs is effectively arbitrary, before applying a softmax to obtain a distribution over candidates, we perform normalizing transformations as described above, and additionally multiply the values of the original ground-truth documents by a factor $b > 1$ to normalize the range and increase the contrast between the top and trailing candidates. Furthermore, we limit the number of candidates that receive a probability above 0 to the top $n_{\text{max}}$ candidates in terms of their similarity to the ground-truth document(s).

We found that these transformations primarily depend on the dataset rather than the model, and that training without limiting $n_{\text{max}}$ leads to overly diffuse score distributions. In case more than one ground-truth documents exist for the same query, the similarity of each candidate is the mean similarity over all ground-truth documents.

**Algorithm 1** Evidence-based label smoothing

```
Require: Dense retrieval model $m$, set of queries $\mathcal{Q}$, document collection $\mathcal{C}$, set of all ground-truth
    label documents per query $\bigcup \mathcal{L}(q),\ \forall q \in \mathcal{Q}$
1: Compute embedding vectors $x_q = m(q),\ \forall q \in \mathcal{Q}$ and $x_{c_i} = m(c_i),\ \forall c_i \in \mathcal{C}$.
2: for each query $q$ do
3:     Retrieve top-$N$ Nearest Neighbors per query based on geometric similarity: $s(q, c_i) = \langle x_q, x_{c_i} \rangle$ for all $c_i \in C$.
4:     for each candidate $c_i, i = 1, \ldots, N$ do
5:         Compute relevance score $r''$ as mixed geometric and reciprocal-NN Jaccard similarity $s_J$ with respect to all ground-truth documents $l$:
               $$r''(q, c_i) = \frac{1}{|\mathcal{L}(q)|} \sum_{l \in \mathcal{L}(q)} s^*(l, c_i),$$
               $$s^*(l, c_i) = \lambda \cdot s(l, c_i) + (1 - \lambda) \cdot s_J(l, c_i), \ \ 0 \ \ < \lambda < 1$$
6:         Transform scores by applying normalization function $f_n$, boost factor $b$ and cut-off threshold $n_{\text{max}}$:
               $$r'(q, c_i) = \begin{cases} b \cdot f_n\left(r''(q, c_i)\right) & \text{if } c_i \in \mathcal{L}(q), \\ -\infty & \text{if } i > n_{\text{max}}, \\ f_n\left(r''(q, c_i)\right) & \text{otherwise.} \end{cases}$$
7:     end for
8: end for
9: Fine-tune model $m$ with target distribution:
       $$\mathbf{r}(q) = \text{softmax}\left(\mathbf{r}'(q)\right),$$
   and loss function:
       $$\mathfrak{L}\left(\mathbf{r}(q), \hat{\mathbf{s}}(\mathbf{q})\right) = \mathrm{D}_{\text{KL}}\left(\mathbf{r}(q) \ || \ \hat{\mathbf{s}}(q)\right),$$
   where $\hat{\mathbf{s}}(q) = \text{softmax}\left(\hat{\mathbf{s}}'(q)/T\right)$ is the model-predicted score distribution, with $T$ a learnable temperature parameter.
```

### 4.3.3 Computational efficiency

Computing rNN similarity involves computing pairwise similarities among $N + 1$ ranking context elements (including the query), and computing the overlap between the top-$k$ reciprocal nearest neighbors of each candidate requires, for each candidate, sorting the rest of the $N$ candidates by their similarity. The computational cost per query is thus $\mathrm{O}(N^2)$ and $\mathrm{O}(N^2 \log N)$, respectively; if we are only interested in the top-$k$ reranked candidates, the latter can be reduced to $\mathrm{O}(N^2 \log k)$. Of course, all pairwise distances between document embeddings (and corresponding sorted lists) can be pre-computed in advance, offline, which would reduce the cost to $\mathrm{O}(N)$ and $\mathrm{O}(N \log k)$ per query respectively.

We find (Sections 4.5.1, B.2) that a small subset of the full ranking context with size $N_r < N$ is generally sufficient when computing rNN-based similarities. For MS MARCO, $N_r = 60$ and the delay per query when reranking on a single CPU and core (AMD EPYC

7532, 2400 MHz) is about 5 ms (Fig. 4.4).

Evidence-based label smoothing imposes no cost during training or inference; it only requires *offline* computation of rNN-similarities for each query context $N_r$ and sorting/top-$k$ as above, followed by simple vectorized transformations, e.g. max-min normalization. Furthermore, all computations above can be trivially ('*embarrassingly*') parallelized in a multi-CPU/core setup.

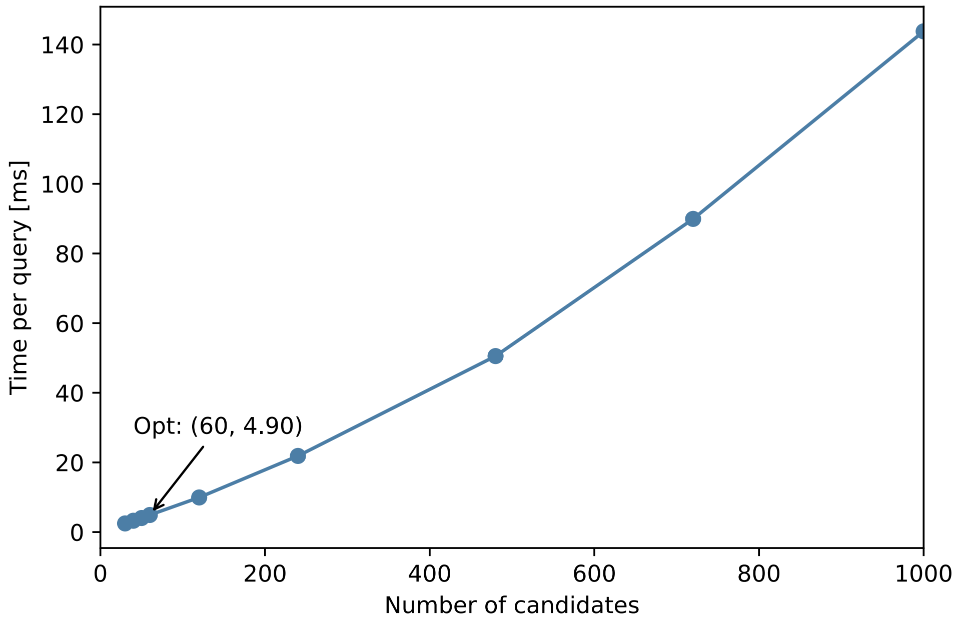


Figure 4.4: Time delay per query (in milliseconds) when reranking using reciprocal nearest neighbors-based similarity, as the number of candidates in the ranking context grows. Hyperparameters are the same as in Fig. 4.5. Processing time scales according to $O(N^2)$. Processor (1 CPU, 1 core): AMD EPYC 7532 32-Core Processor, 2400 MHz.

## 4.4 Experimental setting

**Datasets.** To evaluate the effectiveness of our methods, we use two large-scale, publicly available ad-hoc retrieval collections: the MS MARCO Passage Retrieval dataset [9], and TripClick, a health document retrieval dataset [30]. Each has distinct characteristics and represents one of the two realistic data settings practically available for training dense retrieval models (see details in Appendix A.1, A.2).

**Baselines.** To compute the similarity metric based on reciprocal nearest neighbors, and thus the scores used to either rerank candidates at inference time or calculate the smoothed labels for training, we only need access to the encoder extracting the document and query embeddings. The methods we propose are therefore applicable in principle to any dual-encoder dense retriever. However, we eschew training pipelines based on cross-encoders, both to ensure computational efficiency, as well as to eliminate the dependence on more powerful

| **Hyperparameter** | **TAS-B** | **CODER(TAS-B)** | **CoCondenser** | **CODER(CoCondenser)** |
|---|---|---|---|---|
| $N_r$: context size | 60 | 60 | 53 | 63 |
| $k$: num. NN | 21 | 21 | 21 | 19 |
| $k_{\text{exp}}$: num. NN for expansion | 3 | 3 | 5 | 8 |
| $\tau$: trust factor | 0 | 0 | 0.128 | 0.5 |
| $\lambda$: linear comb. coeff. | 0.451 | 0.451 | 0.469 | 0.473 |

Table 4.1: Hyperparameters for reranking with Reciprocal Nearest Neighbors, MS MARCO.

retrieval methods. Instead, we choose CODER [15], a fine-tuning framework that enhances the performance of dense retrievers used as "base models" through a large ranking context of query-specific candidate documents and a list-wise loss: it serves as a natural framework to evaluate evidence-based label smoothing, because it allows us to directly utilize a large number of soft labels per query, while being very light-weight computationally.

As in Chapter 2, we select the following base models subjected to CODER fine-tuning :
1. RepBERT [31], a BERT-based model with a typical dual encoder architecture which underpins all state-of-the-art dense retrieval methods, trained using a triplet Max-Margin loss.
2. TAS-B [24], one of the top-performing dense retrieval methods on the MS MARCO / TREC-DL 2019, 2020 datasets, which has been optimized with respect to their training process, involving a sophisticated selection of negative documents through clustering of topically related queries.
3. CoCondenser [6], the state-of-the-art dense retrieval model, *excluding* those which make use of heavyweight cross-encoder (query-document term interaction) teacher models or additional pseudo-labeled data samples; it relies on corpus-specific, self-supervised pre-training through a special architecture and contrastive loss component.

## 4.5 Results and Discussion

### 4.5.1 Inference-time reranking with reciprocal nearest neighbors

In this work, rather than directly employing reciprocal nearest neighbors for inference-time reranking, we primarily focus on using them as a means to enhance evidence-based label smoothing. Nevertheless, it is still important to evaluate their effectiveness at improving the similarity metric between queries and documents. This is far from guaranteed in the web retrieval setting, because it substantially differs from the previously proposed setting of image re-identification [12, 13], beyond a simple change of input modality: while all image samples are homologous entities, queries and documents are entities with very distinct characteristics from one another, differing in length, style, vocabulary, and intent. Essentially, we wish to model *relevance*, not the invariance of the same entity under different photographic conditions.

We provide a full description of the inference-time reranking evaluation experiments in the Appendix Section B.2, and summarize the salient results below.

Across all query sets in two important evaluation settings, MS MARCO (Table 4.2) and TripClick (Table 4.3), we observe that using a similarity based on reciprocal nearest neighbors can consistently improve ranking effectiveness for all tested models. The magnitude of

| Model | MS MARCO dev.small | | | MS MARCO dev | | | TREC DL 2019 | | | TREC DL 2020 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** |
| RocketQAv2 [4] | 0.388 | - | - | - | - | - | - | - | - | - | - | - |
| ERNIE-Search [8] | 0.401 | - | - | - | - | - | - | - | - | - | - | - |
| AR2 [7] | 0.395 | - | - | - | - | - | - | - | - | - | - | - |
| AR2 + SimANS [11] | 0.409 | - | - | - | - | - | - | - | - | - | - | - |
| TAS-B | 0.344 | 0.408 | 0.619 | 0.344 | 0.407 | 0.618 | 0.875 | 0.659 | 0.222 | **0.832** | 0.620 | 0.302 |
| R. TAS-B | **0.347** | **0.411** | **0.625** | **0.346** | **0.410** | **0.623** | **0.886** | **0.664** | **0.226** | 0.828 | **0.627** | **0.311** |
| CODER(TAS-B) | 0.355 | 0.419 | 0.633 | 0.353 | 0.416 | 0.627 | **0.857** | 0.668 | 0.224 | 0.844 | 0.623 | 0.306 |
| R. CODER(TAS-B) | **0.357** | **0.421** | **0.637** | **0.354** | **0.418** | **0.631** | 0.853 | **0.679** | **0.231** | **0.860** | **0.634** | **0.317** |
| CoCondenser | 0.381 | 0.446 | 0.665 | **0.381** | 0.446 | 0.664 | **0.879** | 0.656 | **0.226** | **0.833** | 0.618 | 0.301 |
| R. CoCondenser | **0.384** | **0.449** | **0.670** | **0.381** | **0.447** | **0.666** | 0.877 | **0.658** | **0.226** | **0.833** | **0.627** | **0.306** |
| CODER(CoCond) | 0.382 | 0.447 | 0.668 | 0.382 | 0.447 | 0.665 | **0.895** | 0.655 | 0.228 | **0.844** | 0.639 | 0.314 |
| R. CODER(CoCond) | **0.384** | **0.450** | **0.671** | **0.383** | **0.448** | **0.667** | **0.895** | **0.664** | **0.230** | **0.844** | **0.641** | **0.316** |

Table 4.2: Recip. NN reranking, MS MARCO collection. Metrics cut-off @10. **Bold**: best in model class. As a reference, at the top we include all SOTA dense retrieval models from literature that ourperform the methods we evaluated, noting that, unlike ours, they all rely heavily on cross-encoders for training (e.g. distillation, ranking, pseudolabeling etc). Blue: our contributions.

improvement is generally small, but becomes substantial when measured on the TREC DL datasets (approx. +0.010 nDCG@10), where a greater annotation depth and multi-level relevance labels potentially allow to better differentiate between methods.

We furthermore observe that ranking effectiveness initially improves when increasing the size of the ranking context (i.e., the number of candidates considered for reranking), which is expected, because the probability to include a remote ground-truth document in the context increases. However, as this size further increases, ranking effectiveness saturates, often peaking at a context size of a few tens of candidates (Figures 4.5, B.4, B.6, B.8). We hypothesize that this happens because, as we keep adding negatives in the context, the chance that they disrupt the reciprocal neighborhood relationship between query and positive document(s) increases (see Figure 4.1).

We therefore conclude that we may use a relatively small number $N$ of context candidates for computing reciprocal nearest neighbor similarities, which is convenient because computational complexity scales with $O(N^2)$. In MS MARCO, a context of 60 candidates corresponds to peak effectiveness for CODER(TAS-B) and introduces a CPU processing delay of only about 5 milliseconds per query (Figure 4.4). We expect the optimal context size to depend on the average rank that ground-truth documents tend to receive, and for models of similar ranking effectiveness, this would primarily be determined by the characteristics of the dataset.

Indeed, we find that the hyperparameters in computing rNN-based similarity (e.g. $k$, $\lambda$, $\tau$, $f_w$), as well as the context size $N$, predominantly depend on the dataset, and to a much lesser extent on the dense retriever: hyperparameters optimized for CODER(TAS-B) worked very well for TAS-B, CoCondenser and CODER(CoCondenser) on MS MARCO, but very poorly when transferred to TripClick.

| Model | DCTR Head MRR | DCTR Head nDCG | RAW Head MRR | RAW Head nDCG | RAW Head Recall |
|---|---|---|---|---|---|
| BM25[1] | 0.276 | 0.224 | - | 0.199 | 0.128 |
| BERT-Dot (SciBERT)[2] | 0.530 | 0.243 | - | - | - |
| BERT-Cat (SciBERT)[2] | <u>0.595</u> | <u>0.294</u> | - | - | - |
| RepBERT [abbrev: RB] | <u>0.526</u> | 0.255 | 0.574 | 0.344 | 0.199 |
| R. RepBERT | 0.525 | <u>0.256</u> | <u>0.575</u> | <u>0.346</u> | <u>0.200</u> |
| CODER(RB) | 0.634 | 0.316 | 0.674 | <u>0.419</u> | <u>0.234</u> |
| R. CODER(RB) | <u>0.638</u> | <u>0.317</u> | <u>0.679</u> | 0.418 | <u>0.234</u> |
| RB + CODER(RB) | 0.637 | 0.318 | 0.679 | 0.421 | 0.235 |
| RB + R. CODER(RB) | **<u>0.641</u>** | **<u>0.319</u>** | **<u>0.681</u>** | **<u>0.422</u>** | **<u>0.236</u>** |

Table 4.3: Recip. NN reranking, TripClick Test (cut-off @10). **Bold**: overall best, <u>underline</u>: best in model class. Row [1]: from [30], [2]: [32]. Blue: our contributions.

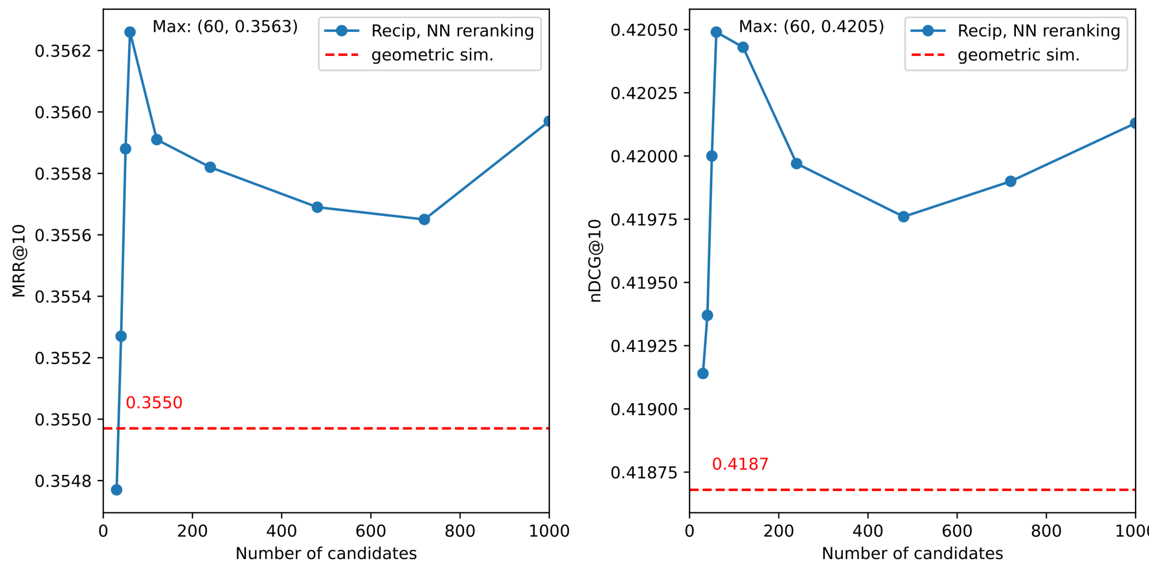


Figure 4.5: Performance of reciprocal nearest neighbors-based reranking of CODER(TAS-B) results on MS MARCO `dev.small`, as the number of candidates in the ranking context grows. Hyperparameters are optimized for a context of 1000 candidates. Performance is slightly improved compared to ranking exclusively based on geometric similarity and peaks at 60 in-context candidates.

| Model | TREC DL 2019 | | | | TREC DL 2020 | | | |
|---|---|---|---|---|---|---|---|---|
| | MRR | nDCG | MAP | Recall | MRR | nDCG | MAP | Recall |
| TAS-B | 0.875 | 0.659 | 0.222 | 0.259 | 0.832 | 0.620 | 0.302 | 0.363 |
| CODER(TAS-B) | 0.857 | 0.668 | 0.224 | 0.270 | 0.844 | 0.623 | 0.306 | 0.365 |
| CODER(TAS-B) + uniform sm. | 0.857 | 0.669 | 0.223 | 0.273 | 0.835 | 0.619 | 0.304 | 0.360 |
| CODER(TAS-B) + geom. smooth labels | 0.848 | 0.665 | 0.220 | 0.271 | 0.842 | 0.626 | 0.310 | 0.370 |
| CODER(TAS-B) + rNN smooth labels | 0.857 | 0.671 | 0.226 | 0.276 | **0.862** | 0.632 | 0.315 | 0.369 |
| **CODER(TAS-B) + mixed rNN/geom. smooth lab.** | **0.889** | **0.675** | **0.227** | **0.277** | 0.842 | **0.637** | **0.318** | **0.376** |
| CoCondenser | 0.879 | 0.656 | 0.226 | 0.269 | 0.833 | 0.618 | 0.301 | 0.366 |
| CODER(CoCondenser) | **0.895** | 0.655 | 0.228 | 0.269 | 0.844 | 0.639 | 0.314 | **0.384** |
| **CODER(CoCondenser) + mixed rNN/geom. smooth lab.** | 0.884 | **0.661** | **0.232** | **0.278** | **0.856** | **0.646** | **0.316** | 0.383 |

Table 4.4: Evaluation of label smoothing applied to training CODER(TAS-B) on MS MARCO. Metrics cut-off @10. **Bold**: best performance in each model class. Blue: our contributions.

### 4.5.2 Evidence-based label smoothing

#### Training on MS MARCO

In order to achieve the best possible results using evidence-based label smoothing, one should ideally optimize the hyperparameters related to rNN-based similarity for the specific task of training a retrieval model with recomputed soft labels. However, to avoid repeatedly computing soft labels for the training set, we simply chose an rNN configuration that was optimized for reranking a large pool of candidates ($N = 1000$) in the MS MARCO collection – i.e., the same one used in the previous section. Although this configuration may not be optimal for our specific task (e.g., small changes in score values might be sufficient for reranking candidates but ineffective as soft training labels), we expect that it can still provide a reliable lower bound of optimal performance.

Table 4.5 shows an example case of label smoothing. We observe that although only a single document is annotated as relevant in the data set (ground truth), the top documents selected by evidence-based label smoothing will deservedly receive higher than zero target similarity, as they are all relevant. In fact, in this case, most documents even up to rank 300 were relevant, although in many other cases relevance decreases after the top few ranks, which justifies using a cut-off maximum number of soft-positives $n_{\max}$. Tables B.3 and B.4 in the Appendix show more examples of label smoothing, which are characteristic of our general observations through manual inspection.

We next evaluate the effects of training with evidence-based smooth labels on ranking effectiveness. Figure 4.6 shows how the ranking performance of the TAS-B base model (leftmost point, step 0) on the validation set evolves throughout fine-tuning through CODER. The red curve corresponds to additionally using evidence-based label smoothing computed with reciprocal NN-based similarity (rNN-related hyperparameters are the same as in Section 4.5.1), whereas for the blue curve the smooth label distribution is computed using pure geometric similarity. We observe the following seemingly paradoxical phenomenon: compared to plain CODER training, label smoothing significantly reduces the *validation* loss (computed with the original labels, top panel), indicating that the ground truth passages are now receiving proportionally higher scores in the estimated relevance distribution, but the retrieval metric

Query: *"what is the appropriate amount of daily protein intake"*

| rank | docID | text |
|---:|---:|---|
| 1 | 1832110* | Since 1 gram of protein = 4 calories, divide protein calories by four: 360/4 = 90 grams of protein per day. No matter what your calculations are, remember that there are no magic foods or supplements that can replace the right training and the right diet. |
| 2 | 623435 | For a 140lb female, calorie intake=1800 calories, protein=20%: 1800 x .20 = 360 calories from protein. Since 1 gram of protein = 4 calories, divide protein calories by four: 360/4 = 90 grams of protein per day. No matter what your calculations are, remember that there are no magic foods or supplements that can replace the right training and the right diet. |
| 3 | 5775976 | Protein should comprise 10 - 15% of a healthy diet. If you eat 1500 calories per day, then you should eat about 56 grams of protein. Take 1500 calories times 15%, then divide by 4 calories per gram. 1500 calories seems low for someone in a weight training program. Your calorie requirements may be as high as 60 calories per kilogram per day. |
| 4 | 362449 | So, if you consume 2,000 calories per day, at least 200 should come from protein, or about 50 grams. You should try to eat around one gram of protein per one kilogram of body weight, or around 0.4 grams per pound. An easier way to figure this out in your head is to take your weight, divide it in half, and subtract 10. |
| 5 | 697249 | As a general rule, between 10 percent and 15 percent of your total calories should come from protein. So, if you consume 2,000 calories per day, at least 200 should come from protein, or about 50 grams. You should try to eat around one gram of protein per one kilogram of body weight, or around 0.4 grams per pound. An easier way to figure this out in your head is to take your weight, divide it in half, and subtract 10. |
| 6 | 5996638 | All proteins have 4 calories per gram, so if you normally consume around 2,200 calories per day, you need 55 to 192 grams of protein each day. Having a vigorous workout regimen may require you to stick to the higher end of the recommendation. |
| 7 | 1413743 | As a general rule, between 10 percent and 15 percent of your total calories should come from protein. So, if you consume 2,000 calories per day, at least 200 should come from protein, or about 50 grams. You should try to eat around one gram of protein per one kilogram of body weight, or around 0.4 grams per pound. An easier way to figure this out in your head is to take your weight, divide it in half, and subtract 10. The total will be the number of grams of protein you should consume each day. |
| 8 | 4013046 | If you take in 1,800 calories per day, 180 to 630 of those calories should be from protein. Each gram of protein has 4 calories, according to the U.S. Food and Drug Administration, so you need to take in 45 to 157.5 grams of protein to meet these recommendations. |
| 9 | 2069624 | For example, if you should weigh 160 pounds but you currently weigh 200 pounds, then your goal for protein intake is in the range of 120 to 150 grams of protein per day. Since each gram of protein is four calories, this means 480 to 600 calories per day from protein. |

Table 4.5: Top documents selected by evidence-based label smoothing to receive higher than zero target relevance. Although only the 1st document (ID: 18322110) is annotated as relevant in the dataset, in this case we observe that all identified documents are actually relevant (false negatives), and will rightfully be treated as soft positives.

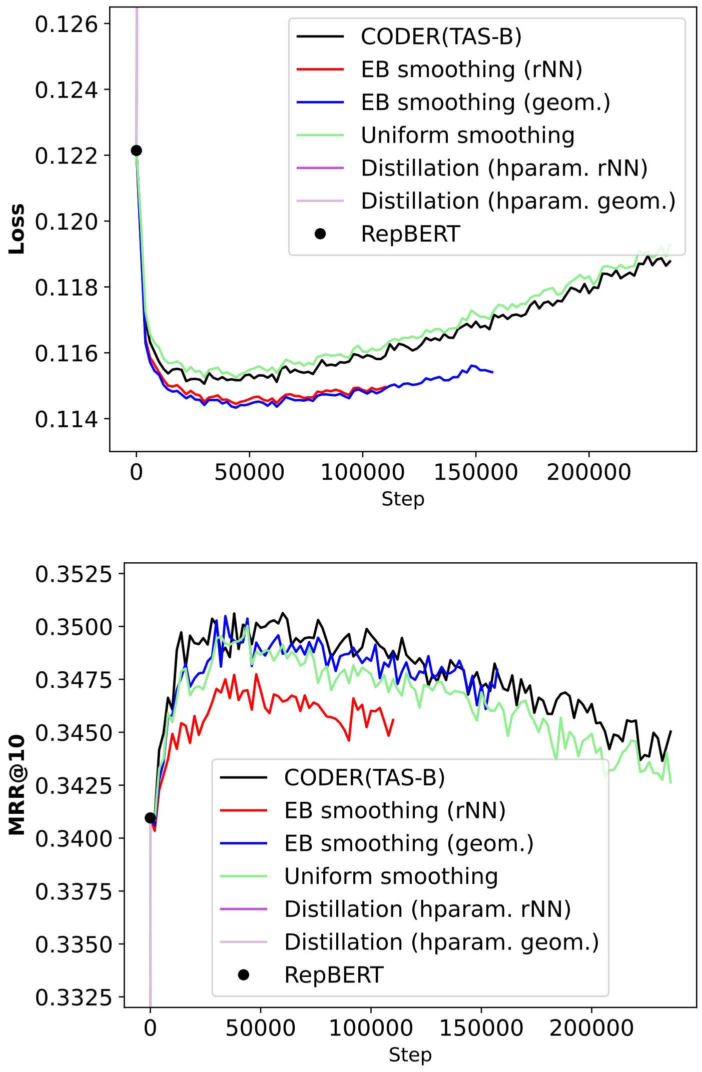


Figure 4.6: Evolution of performance of TAS-B (left-most point, step 0) on MS MARCO dev validation set, as the model is being fine-tuned through CODER. The red curve corresponds to using evidence-based (EB) label smoothing computed with rNN-based similarity, whereas for the blue curve the smooth label distribution is computed using pure geometric similarity. EB label smoothing significantly reduces validation loss (computed with the original labels, top), indicating that the ground truth passages are receiving higher probability in the estimated relevance distribution, but the retrieval metric (bottom) fails to register an improvement due to annotation sparsity (compare with Fig. 4.7, for a dataset with greater annotation depth). Distillation leads to precipitous degradation of performance.

(bottom panel) does not register an improvement.

In fact, this phenomenon may be fully explained through the presence of false negatives: through the smooth target label distribution, the model learns to assign high relevance scores to a larger number of documents (diffuse distribution). Therefore, although on average it places a proportionally higher relevance distribution weight to the ground truth document compared to plain CODER, essentially improving the relevance estimate for the ground truth, at the same time it distributes relevance weight to a higher number of candidates, such that the ground truth ends up being ranked slightly lower (see Figure 4.8).

| **EB label smoothing hyperparam.** | **CODER(TAS-B)** | **CODER(CoCondenser)** |
|---|---|---|
| $b$: boost factor | 1.222 | 1.525 |
| $n_{\text{max}}$: softmax cut-off | 4 | 32 |
| $f_n$: normalization func. | max-min | std-based |
| learning rate: peak value | 1.73e-06 | 1.37e-06 |
| learning rate: linear warm-up steps | 9000 | 12000 |

Table 4.6: Hyperparameters for training with evidence-based label smoothing, MS MARCO. The hyperparameters related to computing rNN-based similarity are the same as in Table 4.1.

We can easily confirm that the model trained with evidence-based label smoothing indeed promotes a more diffuse relevance distribution (systematically distributes relevance probability among more candidates besides the top-ranked). Using all predictions on the validation set, we measure that the average entropy of the relevance probability distribution is 6% higher compared to training without label smoothing. We also measure that the top-ranked candidate now on average receives 10% lower relevance probability, and that the average number of candidates with probability greater or equal to 5% relative to rank 1 is 12% higher.

The crucial question therefore is, whether the candidates now receiving a higher relevance score than before are actually relevant.

Manual inspection of ranking results obtained after training with evidence-based label smoothing indicates that this is indeed most often the case: Table 4.7 shows an example where, although the ranking of the ground-truth positive document decreased from 1 to 3, thereby reducing the MRR from 1 to 0.333, the two documents superseding the ground truth in rank are actually relevant, and so are in fact all top 9 documents. Table B.5 in the Appendix shows another such case, where the ground-truth positive document was demoted from rank 3 to rank 6, but in fact all documents ranked higher than itself were more relevant and informative. Our inspection also revealed that the drop in rank is often caused by the presence of unlabeled near-duplicate documents.

Since the MS MARCO dev dataset almost always contains only a single positive-labeled passage per query, it would miss any positives promoted to higher ranks than the ground truth, and it is therefore fundamentally ill-suited for measuring ranking effectiveness improvements by a training scheme that primarily promotes a diffuse relevance distribution over several candidates. For this reason, we must rely on datasets containing more judgements per query, such as the TREC DL 2019, 2020 datasets.

Table 4.4 shows that evidence-based label smoothing using a similarity based on reciprocal nearest neighbors can significantly improve the performance of each dense retriever even beyond the benefit of the plain CODER fine-tuning framework. Furthermore, using an rNN-based Jaccard similarity as a metric for computing the soft labels yields significantly better performance than using geometric similarity, and the best results are achieved when using a linear combination of the two metrics.

Query: *"how long is a semester"*

| rank | docID | text | Relev. |
|---|---|---|---|
| 1 | 2388818 | A semester divides the school year into two equal parts. This means that if the academic year is 32 weeks, each semester is 16 weeks long. In different countries, in different colleges and universities, the academic year has no standard, so the length may vary. For colleges and universities that operate on a regular two semester academic year, a semester generally runs anywhere from 14 to 16 weeks depending on the institution. | True |
| 2 | 4615643 | A semester divides the school year into two equal parts. This means that if the academic year is 32 weeks, each semester is 16 weeks long. In different countries, in different colleges and universities, the academic year has no standard, so the length may vary. | True |
| 3 | 1006820* | A semester (from the Latin meaning six-monthly) has come to mean either of two academic terms, generally excluding the summer or January terms, if any, and so can be 12 to 20 weeks long. The word semester is sometimes used as a synonym for a term, as in a summer semester. | True* |
| 4 | 445722 | Make sure you know that term is a generic word that is commonly used in educational colleges and universities, to outline the length of an academic schedule. It is mostly used in institutions in Britain whereas semester is a more popular word in the educational colleges in the United States. The length of a semester is 6 months and there are normally 2 semesters each year. However, a few schools have trimesters and quarters which is 3 to 4 terms in a year. | True |
| 5 | 1054466 | How long does a semester last in college? A semester divides the school year into two equal parts. This means that if the academic year is 32 weeks, each semester is 16 weeks long. In different countries, in different colleges and universities, the academic year has no standard, so the length may vary. | True |
| 6 | 1965901 | The semester system divides the calendar year into two semesters of 16 to 18 weeks each, plus summer sessions of varying lengths. The two semesters together constitute 32 to 36 weeks of instruction, so that three academic quarters equal two academic semesters. semester (from the Latin meaning six-monthly) has come to mean either of two academic terms, generally excluding the summer or January terms, if any, and so can be 12 to 20 weeks long. The word semester is sometimes used as a synonym for a term, as in a summer semester. | True |
| 7 | 5840984 | A semester divides the school year into two equal parts. This means that if the academic year is 32 weeks, each semester is 16 weeks long. In different countries, in different colleges and universities, the academic year has no standard, so the length may vary. community college semester lasts roughly 4.5 months. This will vary depending on how long each specific college allows for Christmas break and New Years break. | True |
| 8 | 5808478 | A typical college semester can be defined as fifteen weeks long, depending on the school. With a typical fifteen-week-long semester, the academic calendar is divided into three semesters. The fall and spring semesters will both be fifteen weeks long and the third semester, summer, will usually be shorter. The summer semester is generally about twelve weeks long. You might find semester lengths vary from school to school, within a range of one to three weeks. | True |
| 9 | 3425574 | A typical college semester has three months. However, the length of a semester varies from state to state. The fall and spring semesters each have 15 weeks, while the summer semester is usually shorter with about 12 weeks. | True |

Table 4.7: Top documents retrieved by a CODER(TAS-B) model trained through EB label smoothing. Although only the document marked with * (ID: 1006820) is annotated as relevant in the dataset, we observe that all top documents are actually relevant (false negatives), and thus the MRR metric would erroneously decrease from 1 (as the ground-truth positive previously ranked 1st) to 0.333.

| Model | DCTR Head | | | RAW Head | | |
|---|---|---|---|---|---|---|
| | **MRR** | **nDCG** | **Recall** | **MRR** | **nDCG** | **Recall** |
| RepBERT | 0.526 | 0.255 | 0.242 | 0.574 | 0.344 | 0.199 |
| CODER(RB) | 0.610 | 0.300 | 0.276 | 0.656 | 0.401 | 0.228 |
| CODER(RB) hparam. | 0.608 | 0.300 | 0.277 | 0.649 | 0.401 | 0.229 |
| **CODER(RB) + EB smooth.** | **0.611** | **0.305** | **0.280** | **0.661** | **0.411** | **0.234** |

Table 4.8: Evaluation of evidence-based label smoothing (mixed rNN - geom. similarity) on TripClick `HEAD Test`. Models were trained on TripClick `HEAD` ∪ `TORSO Train` and validated on `HEAD Val`. Metrics cut-off @10. "hparam": model trained with same hyperparameters as the one with label smoothing. Blue: our contributions.

**Zero-shot evaluation on TripClick.** As TripClick also contains several (pseudo-relevance) labels per query, we additionally evaluate the MS MARCO-trained models zero-shot (i.e., without any training) on TripClick Test and Val (Figures B.1, B.2, Appendix). We again observe that evidence-based label smoothing with an rNN-based metric improves performance compared to plain CODER; however, we note that in this zero-shot setting, the best performing models were not in general the same as the best performing models on TREC DL. The best ranking performance was achieved by CODER(TAS-B) using soft labels from pure rNN-based Jaccard similarity.

We thus find that in sparsely annotated datasets like MS MARCO, validation loss might be a better predictor of model generalization than IR metrics such as MRR, and that evaluation on datasets with higher annotation depth (such as TREC DL or TripClick), potentially even in a zero-shot setting, might better reflect the ranking effectiveness of models.

**Comparison to distillation and uniform label smoothing.** A critical difference of evidence-based label smoothing from distillation is that soft document labels are computed based on their similarity to the ground truth instead of the query. To demonstrate the importance of this change of perspective, we show how CODER fine-tuning performs when using soft labels coming from geometric similarity with respect to the query, as in distillation (Figure 4.6, purple curves): even when applying the same transformations to the scores as in the case of evidence-based label smoothing, the model's performance rapidly degrades instead of improving. This is expected, because distillation only works when a superior model is available; training cannot be bootstrapped from the scores of the model itself.

We also observe that, unlike evidence-based label smoothing, uniform label smoothing fails to noticeably improve performance compared to plain CODER fine-tuning (Figure 4.6, Table 4.4), even when we ensure that the exact same probability weight as in the case of evidence-based smoothing is distributed from the ground-truth positive(s) among the rest of the candidates.

### Training on TripClick

Finally, we examine how EB label smoothing performs when training in an important alternative setting, TripClick: a dataset with significantly more relevance labels per query, that come from pseudo-relevance feedback without human judgements. Unlike above, here we investigate the joint optimization of rNN-related parameters together with training-specific parameters (e.g., learning rate and linear warm-up steps), instead of using the same rNN-related hyperparameters for label smoothing as for reranking. To allow this, we train on the union of the `HEAD` and `TORSO` training subsets (avg. 42 and 9 annotations per query, respectively), and omit the `TAIL` subset, which consists of a large number of rare queries (each with only 3 annotations on average). We use `HEAD Val` as a validation set, and evaluate on `HEAD Test`.

Table 4.8 and Figure 4.7 show that training with mixed geometric/rNN-based smooth labels significantly improves performance also in this dataset setting compared to plain CODER training (+0.010 nDCG@10). To ensure that any improvement cannot be attributed to better hyperparameters found during the joint optimization described above, we also apply the same hyperparameters to plain CODER training (denoted "hyperparam." in the table).

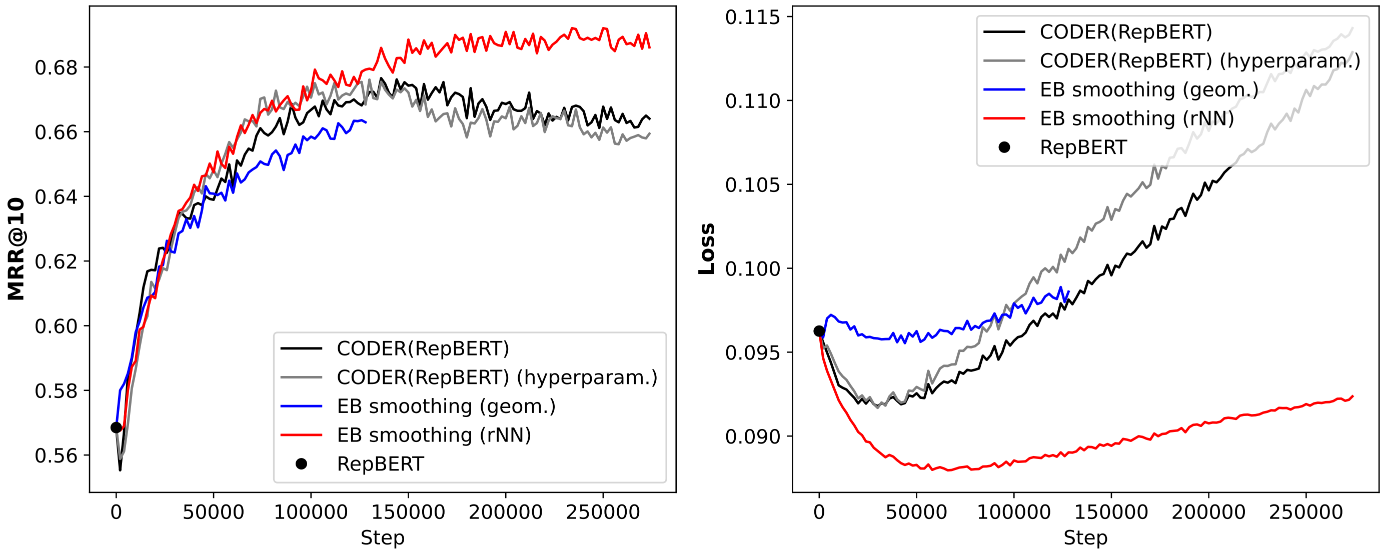


Figure 4.7: Evolution of performance of RepBERT (left-most point, step 0) on the TripClick `HEAD Val` validation set, as the model is being fine-tuned through CODER on TripClick `HEAD∪TORSO Train`. The red curve corresponds to additionally using evidence-based label smoothing computed with a reciprocal NN-based similarity component, whereas for the blue curve the smooth label distribution is computed using pure geometric similarity. Only evidence-based smoothing with rNN similarity substantially improves performance compared to plain CODER(RepBERT), despite "CODER(RepBERT) (hyperparam.)" and "EB smoothing" with geometric similarity using the same training hyperparameters.

We observe similar improvements on `TORSO Test` and `TORSO Val` ( Table 4.9).

## 4.6 Limitations and applicability to other frameworks

In principle, the methods we propose in this chapter are applicable to any dual-encoder dense retriever: computing the similarity metric based on reciprocal nearest neighbors only requires access to the encoder extracting the document and query embeddings.

| | Test RAW Torso | | | Val RAW Torso | | |
|---|---|---|---|---|---|---|
| **Model** | **MRR@10** | **nDCG@10** | **Recall@10** | **MRR@10** | **nDCG@10** | **Recall@10** |
| RepBERT | 0.338 | 0.247 | 0.309 | 0.398 | 0.288 | 0.342 |
| CODER(RepBERT) | 0.390 | 0.276 | 0.330 | 0.426 | 0.310 | 0.354 |
| CODER(RepBERT) hyperparam. | 0.389 | 0.277 | 0.331 | **0.428** | 0.310 | 0.354 |
| **CODER(RepBERT) + mixed rNN/geom. smooth label.** | **0.391** | **0.282** | **0.340** | 0.421 | **0.312** | **0.367** |

Table 4.9: Label smoothing applied to CODER(RepBERT) trained on TripClick `HEAD` ∪ `TORSO Train`, validated on `HEAD Val`; evaluation on `TORSO Test` and `Val`.

However, we note that the reason we were able to compute the soft labels for evidence-based label smoothing completely offline was that we utilized CODER as a fine-tuning framework: CODER only fine-tunes the query encoder, using fixed document representations. Using evidence-based label smoothing in a training method with learnable document embeddings means that the rNN-based similarity has to be computed dynamically at each training step (or periodically every few training steps), because their mutual distances/similarities will change during training, albeit slowly. Similarly, every time candidates/negatives are retrieved dynamically (periodically, as in 14, or at each step, as in 3) the rNN-based similarity has to be recomputed among this new set. Nevertheless, as we discuss in the paper, we only need to use a context of tens or at most a couple of hundred candidates in order to compute the rNN-based similarity most effectively. Even in these cases, this would therefore introduce at most up to a hundred milliseconds of training delay per batch, while inference would remain unaffected. Most likely, the initial "soft" (smooth) labels would be anyway useful as a mitigation for false labels, thereby improving the training signal and calibrating the model's relevance scores; dynamically recomputing the soft labels as document representations change would simply offer an additional improvement to effectiveness of the method.

The KL-divergence loss used within the CODER framework is a suitable choice for utilizing the soft labels computed by evidence-based label smoothing in training dense retrievers. Since practically all contemporary dense retrieval training methods rely on a contrastive loss such as InfoNCE (also known as Negative Log-Likelihood), to directly allow incorporating the recomputed labels, one would have to binarize them, e.g. by setting a relevance score threshold, above and below which the documents would be considered positive or negative respectively. However, this forced discretization will presumably yield a sub-optimal effect. Instead, a likely more suitable option would be to simply replace the InfoNCE loss used in these methods with KL-divergence; it is an approach that allows seamless utilization of the continuous "smooth" scores, and shows promise, as existing training pipelines have already successfully integrated this loss as part of a knowledge distillation scheme (using scores assigned by a cross-encoder as a teacher) [4].

## 4.7 Conclusion

In this chapter we have proposed evidence-based label smoothing to address sparse annotation in dense retrieval datasets. In this method, to mitigate penalizing the model in case of false negatives during training, we compute the target relevance distribution by assigning non-zero relevance probabilities to candidates most similar to the documents annotated as ground truth. To estimate similarity we leverage reciprocal nearest neighbors, which allows considering local connectivity in the shared representation space, and can independently be used for reranking.

Extensive experiments on two large-scale retrieval datasets and three dense retrieval models demonstrate that our method can effectively improve ranking, while being computationally efficient and foregoing the use of resource-heavy cross-encoders.

Finally, we have shown evidence that evaluating on sparsely annotated datasets like MS MARCO `dev` may systematically underestimate models with less sharp (i.e. more diffuse) relevance score distributions. We have found that in such datasets, the KL-divergence loss between predicted and target distribution may in fact better capture the actual ranking effectiveness of the model. To evaluate retrieval models more reliably, it appears to be indispensable to evaluate on datasets with greater annotation depth (e.g. TREC DL), even when evaluation is performed zero-shot (i.e. without previously training on that dataset), and even when the annotations originate from click-through models and not human judgments (e.g. TripClick).

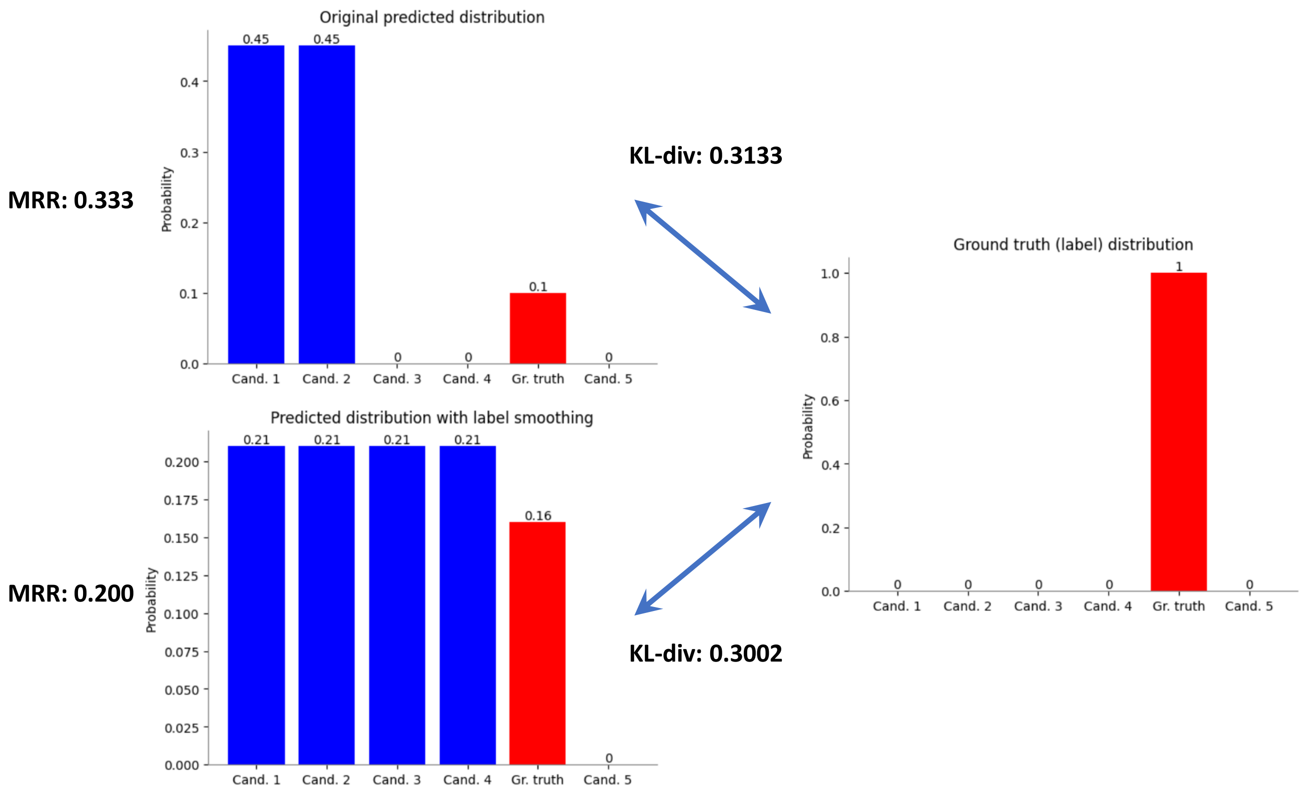


Figure 4.8: Because many more documents receive higher than zero relevance in the target distribution after label smoothing, by design it promotes a diffuse predicted distribution (bottom). Thus, although the predicted relevance of the ground-truth positive document is now significantly higher compared to when not using label smoothing (top), indicating a model improvement, the document ends up ranking lower because of the dispersed relevance estimates, and thus the MRR metric decreases. By contrast, the KL-divergence (i.e. loss function) correctly captures the improvement in assessing the relevance of the ground-truth positive. We note that in sparsely annotated datasets like MS MARCO, the "1-hot" ground-truth annotations (right) are very often incorrect among the top ranks, and some of the candidates ranked more highly than the ground truth (e.g. Candidates 3 and 4 in the figure) may actually be relevant, which would render the MRR metric spurious; 10 estimate that about 70% of the top 5 candidates retrieved by a top-performing dense retrieval model that are not labeled as positive are in fact relevant.

# Bibliography

[1] Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *arXiv:1901.04085 [cs]*, April 2020. URL http://arxiv.org/abs/1901.04085. arXiv: 1901.04085.

[2] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction, December 2021. URL http://arxiv.org/abs/2112.01488. arXiv:2112.01488 [cs].

[3] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada, July 2021. ACM. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462880. URL https://dl.acm.org/doi/10.1145/3404835.3462880.

[4] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.224. URL https://aclanthology.org/2021.emnlp-main.224.

[5] Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, 2021. doi: 10.18653/v1/2021.findings-acl.191. URL http://arxiv.org/abs/2108.06027. arXiv: 2108.06027.

[6] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *ArXiv*, abs/2108.05540, 2021.

[7] Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. Adversarial Retriever-Ranker for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/pdf?id=MR7XubKUFB.

[8] Yuxiang Lu, Yiding Liu, Jiaxiang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, and Haifeng Wang. ERNIE-Search: Bridging Cross-Encoder with Dual-Encoder via Self On-the-fly Distillation for Dense Passage Retrieval, May 2022. URL http://arxiv.org/abs/2205.09153. arXiv:2205.09153 [cs].

[9] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]*, October 2018. URL http://arxiv.org/abs/1611.09268. arXiv: 1611.09268.

[10] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.466. URL https://aclanthology.org/2021.naacl-main.466.

[11] Kun Zhou, Yeyun Gong, Xiao Liu, Wayne Xin Zhao, Yelong Shen, Anlei Dong, Jingwen Lu, Rangan Majumder, Ji-rong Wen, and Nan Duan. SimANS: Simple ambiguous negatives sampling for dense text retrieval. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 548–559, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-industry.56.

[12] Danfeng Qin, Stephan Gammeter, Lukas Bossard, Till Quack, and Luc van Gool. Hello neighbor: Accurate object retrieval with k-reciprocal nearest neighbors. In *CVPR 2011*, pages 777–784, June 2011. doi: 10.1109/CVPR.2011.5995373. ISSN: 1063-6919.

[13] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking Person Re-identification with k-Reciprocal Encoding. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3652–3661, July 2017. doi: 10.1109/CVPR.2017.389. ISSN: 1063-6919.

[14] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*, October 2020. URL http://arxiv.org/abs/2007.00808. arXiv: 2007.00808.

[15] George Zerveas, Navid Rekabsaz, Daniel Cohen, and Carsten Eickhoff. CODER: An efficient framework for improving retrieval through COntextual document embedding reranking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10626–10644, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.727.

[16] Fernando Diaz. Regularizing query-based retrieval scores. *Information Retrieval*, 10(6): 531–562, October 2007. ISSN 1386-4564, 1573-7659. doi: 10.1007/s10791-007-9034-8. URL http://link.springer.com/10.1007/s10791-007-9034-8.

[17] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7(5):217–240, December 1971. ISSN 0020-0271. doi: 10.1016/0020-0271(71)90051-9. URL https://www.sciencedirect.com/science/article/pii/0020027171900519.

[18] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, June 2007. Association for Computing Machinery. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

[19] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks. *arXiv:1811.04415 [cs]*, August 2019. URL http://arxiv.org/abs/1811.04415. arXiv: 1811.04415.

[20] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. Learning a Deep Listwise Context Model for Ranking Refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 135–144, New York, NY, USA, June 2018. Association for Computing Machinery. ISBN 978-1-4503-5657-2. doi: 10.1145/3209978.3209985. URL https://doi.org/10.1145/3209978.3209985.

[21] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4): 346–374, August 2010. ISSN 1386-4564, 1573-7659. doi: 10.1007/s10791-009-9123-y. URL http://link.springer.com/10.1007/s10791-009-9123-y.

[22] Olivier Chapelle and Yi Chang. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14*, YLRC'10, pages 1–24, Haifa, Israel, June 2010. JMLR.org.

[23] Domenico Dato, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Nicola Tonellotto, and Rossano Venturini. Fast Ranking with Additive Ensembles of Oblivious and Non-Oblivious Regression Trees. *ACM Transactions on Information Systems*, 35(2):15:1–15:31, December 2016. ISSN 1046-8188. doi: 10.1145/2987380. URL https://doi.org/10.1145/2987380.

[24] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and A. Hanbury. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *SIGIR*, 2021. doi: 10.1145/3404835.3462891.

[25] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th*

*International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 65–74, New York, NY, USA, August 2017. Association for Computing Machinery. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080832. URL https://dl.acm.org/doi/10.1145/3077136.3080832.

[26] Dany Haddad and Joydeep Ghosh. Learning More From Less: Towards Strengthening Weak Supervision for Ad-Hoc Retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 857–860, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6172-9. doi: 10.1145/3331184.3331272. URL https://dl.acm.org/doi/10.1145/3331184.3331272.

[27] Gianluca Moro and Lorenzo Valgimigli. Efficient Self-Supervised Metric Information Retrieval: A Bibliography Based Method Applied to COVID Literature. *Sensors*, 21 (19):6430, January 2021. ISSN 1424-8220. doi: 10.3390/s21196430. URL https://www.mdpi.com/1424-8220/21/19/6430. Number: 19 Publisher: Multidisciplinary Digital Publishing Institute.

[28] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

[29] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a Visual Language Model for Few-Shot Learning, November 2022. URL http://arxiv.org/abs/2204.14198. arXiv:2204.14198 [cs].

[30] Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2513. Association for Computing Machinery, New York, NY, USA, July 2021. ISBN 978-1-4503-8037-9. URL https://doi.org/10.1145/3404835.3463242.

[31] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. RepBERT: Contextualized Text Embeddings for First-Stage Retrieval. *arXiv:2006.15498 [cs]*, July 2020. URL http://arxiv.org/abs/2006.15498. arXiv: 2006.15498.

[32] Sebastian Hofstätter, Sophia Althammer, Mete Sertkan, and Allan Hanbury. Establishing Strong Baselines for TripClick Health Retrieval. *arXiv:2201.00365 [cs]*, January 2022. URL http://arxiv.org/abs/2201.00365. arXiv: 2201.00365.

# Chapter 5

# Conclusion and outlook

## 5.1 A new framework for contextual ranking and similarity learning for Large Language Models

The focus of this thesis was to investigate the potential of contextual ranking and similarity learning in improving the state-of-the-art information retrieval methods, which are based on large, pre-trained transformer language models (PTLMs). As an overarching conclusion, we have gathered strong evidence in support of the thesis that utilizing ranking context can substantially benefit the training and ranking effectiveness of PTLM-based retrievers, and we have introduced practical methods towards that end.

We have identified that individual best practices employed in contemporary dense retrieval methods, such as mining hard negatives and using a large number of negatives for contrastive learning, can be seen as partial steps towards building a *ranking context*. Ranking context, which was systematically exploited in pre-PTLM Learning-to-Rank models, is to be understood as a large enough set of documents in meaningful relationship to the same query (and thus to one-another), that are all jointly assessed with respect to their similarity to the query.

We have examined the constituent parts of ranking context, that is, specificity to the query, a large number of candidates, and the use of a list-wise loss, and found that together they can significantly contribute to the ranking effectiveness of trained models. To leverage ranking context in a way compatible with the computational constraints introduced by large transformer-based models, we have proposed CODER (COntextual Document Embedding Reranking), an efficient and light-weight framework that relies on training with precomputed, fixed document representations and only fine-tuning the query encoder part of the common dual encoder architecture. Using an existing trained retriever as a starting point, contextual similarity training can thus substantially boost its ranking performance at the minimal cost of a short fine-tuning on a single GPU.

Importantly, incorporating ranking context into the training of PTLM retrieval models enables us to optimize them with respect to context-dependent properties: that is, properties such as ranking fairness, that only manifest themselves when examining the ranking of an entire set of documents. We have demonstrated how adding a neutrality regularization term to the relevance-optimizing loss term allows one to easily train a model that systematically promotes more gender-neutral documents. The intensity of regularization during training allows us to continuously and predictably modulate the neutrality-boosting effect, unlike existing alternatives for bias mitigation, such as adversarial training, and the approach offers both higher ranking effectiveness and higher neutrality compared to these alternatives. We have thus introduced the first truly practical approach for training PTLM retrieval models by inherently taking into account fairness, source bias and other important properties besides ranking effectiveness.

Finally, we investigated the potential for improving dense retrieval when leveraging a similarity metric beyond simple geometric proximity between query and document embeddings. Using the concept of Reciprocal Nearest Neighbors to compute this metric allows us to take into account the local density of documents, i.e., the degree of connectivity of query and documents to other documents in the common representation space. Although we showed that such a metric is in itself useful when ranking, we further utilized it as part of a label smoothing scheme, in which the degree of similarity of unlabeled documents to a ground-truth relevant document serves as evidence for distributing higher than zero relevance probability among candidates in the same ranking context. In practice, using these smooth labels enhances the ranking context available for training retrieval models, mitigating the sparse relevance annotation problem of the large datasets their training relies on and improving their ranking effectiveness.

## 5.2 (Very) Large Language Models for in-context ranking

Since Brown et al. [1] convincingly demonstrated the capabilities of very large, pre-trained Language Models (LLMs) for few-shot in-context learning, they have been applied with impressive success on most Natural Language Processing tasks. It then comes as little surprise that they have been recently also employed for in-context re-ranking of retrieved passages.

In particular, Sun et al. [2] use ChatGPT (i.e., GPT-3.5) and GPT-4, each hundreds of billions of parameters large, to rerank a small number of top passages retrieved for a given query, by means of providing the query, the numbered passages and a natural language instruction within a prompt, asking the LLM to output a *permutation* of the passages in the order of descending relevance. Because of the models' context size limitations constraining the prompt length, reranking entails successively sorting subsets of the passages within a sliding window of size $k$. Starting from the bottom of the list, the bottom $k$ passages are reranked, and the window is shifted by a stride of $k/2$ ranks higher to form a new, overlapping context window, where the top reranked passages of the previous window are compared with the bottom passages of the new, shifted window. The process is repeated until the passages deemed most relevant float to the top. This approach was found significantly more effective than alternative approaches, where the LLM was asked to output a relevance score for a passage given a query, and results are sorted by score.

We note that, because of the onerous latency and computational cost it introduces, this approach is limited to only reranking top results retrieved by other methods. Therefore, it is not in competition with dense retrieval, which procures candidates as a first-stage method, but neither is it yet truly competitive with cross-encoder reranking, which is currently much faster

| **Model** | **TREC DL'19** | **TREC DL'20** |
|---|---|---|
| BM25 | 0.506 | 0.480 |
| **Supervised** (cross-encoder reranking) | | |
| monoBERT (340M) | 0.705 | 0.673 |
| monoT5 (220M) | 0.715 | 0.670 |
| monoT5 (3B) | 0.718 | 0.689 |
| Cohere Rerank-v2 | 0.732 | 0.671 |
| DeBERTa-Large (304M) | 0.689 | 0.614 |
| Llama-7B | 0.692 | 0.590 |
| **Distillation** (cross-encoder trained from ChatGPT labels) | | |
| DeBERTa-Large (304M) | 0.707 | 0.672 |
| Llama-7B | 0.718 | 0.669 |
| **LLM API** (permutation generation) | | |
| GPT-3.5-turbo | 0.658 | 0.629 |
| GPT-4 | **0.756** | **0.706** |
| **Ours** (dense retrieval) | | |
| CODER(TAS-B) (66M) | 0.728 | 0.686 |
| CODER(TAS-B) + rNN rerank | **0.740** | 0.695 |
| CODER(CoCondenser) (110M) | 0.715 | 0.697 |
| CODER(CoCondenser) + rNN rerank | 0.723 | **0.698** |

Table 5.1: Ranking effectiveness (nDCG@10) on TREC DL 2019, 2020 of SOTA models trained on MS MARCO (ChatGPT and GPT-4 are not trained, but evaluated few-shot through API calls). Results other than CODER come from Sun et al. [2]. To directly compare, here we follow the same "lax" method of computing nDCG, where a ground truth score of "1" is considered relevant (and is not mapped to 0, as per official TREC guidelines).

and cheaper to run compared to in-context reranking through LLMs. Even if considered mainly for academic reasons, it is however still interesting to examine the performance achieved by in-context reranking with LLMs, to surmise its role in current and future information retrieval.

Table 5.1 shows how LLM reranking through APIs performs on the TREC DL 2019, 2020 evaluation sets compared to the top-performing cross-encoders trained on MS MARCO. We observe that, although GPT-3.5 cannot outperform the top specialized cross-encoders, GPT-4, which is substantially larger and more extensively pre-trained, presently sets the record performance.

How do our own models compare? Interestingly, we see that our best performing models on these two evaluation sets (from Chapter 4) outperform both the largest and most powerful cross-encoders trained on MS MARCO labels (which introduce hundreds of milliseconds of latency per query), as well as those trained using distillation from GPT-3.5 annotations. In fact, they essentially perform on par with GPT-4, which, as we noted above, is not really a practical or scalable method for retrieval. This is result is very impressive, considering that our models (a) comprise 66M and 110M parameters, instead of several hundred billion, and (b) can be directly used for first-stage retrieval, with an additional reranking latency of a few milliseconds per query; their are thus orders of magnitude faster and cheaper to run.

Consequently, we can conclude that a suitable training setting, such as the contextual similarity learning framework introduced in this thesis, has a tremendous effect on performance, and can easily compete with the largest, most computationally expensive models to date.

Instead of using LLMs for reranking documents, it would appear that their greatest potential lies in their unparalleled capability for generation: one can thus envisage their use for enhancing the training context of supervised models by generating synthetic documents of specified relevance levels, given a query. We intend to explore this use in future work.

## 5.3 Remaining challenges and future directions

### 5.3.1 Expanding the generality of the contextual similarity learning framework

In the frame of this thesis, we have demonstrated that training state-of-the-art dense retrieval models stands to benefit from contextual similarity learning, and proposed practical methods to achieve this.

The principles we have examined, but also the corresponding methods we proposed, are applicable to any dual-encoder dense retriever – some, like evidence-based label smoothing, can also be used to train cross-encoder models. However, in its present form, the main framework we have introduced can only fine-tune existing models as a separate step, because it relies on static, precomputed document embeddings. To directly integrate our methods into the most advanced, computationally expensive pipelines that are used to train the SOTA dense retrieval models from scratch, we would need to make certain modifications.

In Section 2.5.4 we have seen that while a large number of context candidates is important, there are diminishing returns as this number grows beyond a few hundred documents. To allow trainable document representations, we could thus limit the context size to, e.g., 256 pre-retrieved candidates, while dispensing with the large number of random in-batch negatives, which, as we have seen, do not meaningfully contribute to improving ranking effectiveness, despite their ubiquitous use in contemporary literature *in lieu* of a large enough number of retrieved candidates. Presumably, random in-batch negatives could still be useful at the beginning of training, and could be progressively phased out in favor of retrieved candidates as training continuous. These candidates can be retrieved dynamically, as proposed by Xiong et al. [3] and Zhan et al. [4].

At the same time, instead of the InfoNCE loss, that is employed by contemporary pipelines, it is straightforward to use the same list-wise loss as in CODER, i.e. the KL-divergence between the target relevance distribution and the model-estimated relevance score distribution. Besides better exploiting ranking context (see Section 2.5.4), this would also allow the seamless use of the soft labels computed by the evidence-based label smoothing method; otherwise, one would need to binarize the continuous target relevance labels and use them either as positives or negatives in contrastive learning, which is likely sub-optimal.

With these changes, the principles of contextual similarity learning we have explored could be incorporated into a pipeline that involves a large number of GPUs and could train SOTA dense retrieval models from scratch (e.g. [5, 6]).

Finally, techniques for parameter-efficient fine-tuning, such as adapters [7] and LoRa [8], combined with modern methods for training models using low-precision arithmetic (quantization) [9] are a very promising approach for incorporating contextual similarity learning into the training pipeline of dense retrieval models, while allowing optimization of both the query as well as the document representations.

### 5.3.2 Extending the versatility of the bias mitigation approach

The modifications in the above section would allow optimizing SOTA models for properties like fairness exactly in the way described in Chapter 3. However, we note that the practicality of this method is still limited, in the sense that one has to decide in advance the strength of regularization applied during training, and therefore the point in which the deployed model will be operating in the trade-off between utility and fairness. Additionally, we have only tested that the approach works when optimizing for fairness with respect to a single protected attribute (e.g. gender), while in practice we would prefer fair outcomes with respect to a combination of such attributes (gender, nationality, ethnicity, race, religious background, etc). It principle, one could train the retrieval model within CODER by adding several regularization loss terms, one for each attribute, at the same time, but it remains to be empirically validated.

As an alternative, in work following up on our work presented in Chapter 3 [10], it was demonstrated that it is possible to separately train individual adapters, one for each such protected attribute and one for the relevance ranking task, and then finally train an AdapterFusion module [11] to use all adapters simultaneously. However, training through CODER with several regularization terms simultaneously is a much simpler and efficient approach, and should be investigated in future work.

# Bibliography

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[2] Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents, October 2023. URL http://arxiv.org/abs/2304.09542. arXiv:2304.09542 [cs].

[3] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*, October 2020. URL http://arxiv.org/abs/2007.00808. arXiv: 2007.00808.

[4] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada, July 2021. ACM. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462880. URL https://dl.acm.org/doi/10.1145/3404835.3462880.

[5] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.466. URL https://aclanthology.org/2021.naacl-main.466.

[6] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *ArXiv*, abs/2108.05540, 2021.

[7] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

[8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. URL http://arxiv.org/abs/2106.09685. arXiv:2106.09685 [cs].

[9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[10] Deepak Kumar, Oleg Lesota, George Zerveas, Daniel Cohen, Carsten Eickhoff, Markus Schedl, and Navid Rekabsaz. Parameter-efficient modularised bias mitigation via AdapterFusion. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2738–2751, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.201. URL https://aclanthology.org/2023.eacl-main.201.

[11] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39.

# Appendix A

# General

## A.1 Data

### A.1.1 MS MARCO and TREC Deep Learning

Following the standard practice in related contemporary literature, we use the MS MARCO dataset [1], which has been sourced from open-domain logs of the Bing search engine, for training and evaluating our models. The MS MARCO passage collection contains about 8.8 million passages and the training set contains about 503k queries labeled with one or (rarely) more relevant passages (1.06 passages per query, on average), on a single level of relevance.

For validation of the trained models we use a subset of 10k samples from "MS MARCO dev", which is a set containing about 56k labeled queries, and refer to it as "MS MARCO dev 10k". As a test set we use a different, officially designated subset of "MS MARCO dev", originally called "MS MARCO dev.small", which contains 6980 queries. Often, in literature and leaderboards it is misleadingly referred to as "MS MARCO dev".

Because of the very limited annotation depth (sparsity) in the above evaluation sets, we also evaluate on the TREC Deep Learning track 2019 and 2020 test sets, each containing 43 and 54 queries respectively, labeled to an average "depth" of more than 210 document judgements per query, and using 4 levels of relevance: "Not Relevant" (0), "Related" (1), "Highly Relevant" (2) and "Perfect" (3). According to the official (strict) interpretation of relevance labels[1], a level of 1 should not be considered relevant and thus be treated just like a level of 0, while the lenient interpretation considers passages of level 1 relevant when calculating metrics.

### A.1.2 TripClick

TripClick is a recently introduced health IR dataset [2] based on click logs that refer to about 1.5M MEDLINE articles. The approx. 700k unique queries in its training set are split into 3 subsets, HEAD, TORSO and TAIL, based on their frequency of occurrence: queries in TAIL are asked only once or a couple of times, while queries in HEAD have been asked tens or hundreds of times. As a result, each query in HEAD, TORSO and TAIL on average ends up with 41.9, 9.1 and 2.8 pseudo-relevance labels, using a click-through model (RAW) where

[1] https://trec.nist.gov/data/deep2019.html

every clicked document is considered relevant. The dataset also includes alternative relevance labels using the Document Click-Through Rate (DCTR), on 4 distinct levels (the latter follow the same definitions as the TREC Deep Learning evaluation sets). We note that, although the number of labels per query is much higher than MS MARCO, unlike the latter, these labels have not been verified by human judges.

For validation and evaluation of our models we use the officially designated validation and test set, respectively (3.5k queries each).

## A.2 Evaluation principles

All training and evaluation experiments are produced with the same seed for pseudo-random number generators. We use mean reciprocal rank (MRR), normalized discounted cumulative gain (nDCG), mean average precision (MAP) and recall to evaluate the models on TREC DL tracks, MS MARCO and TripClick, in line with past work (e.g. [2, 3, 4, 5]). While relevance judgements are well-defined in MS MARCO and TripClick, for the TREC DL tracks there exist strict and lenient interpretations of the relevance scores of judged passages (see Section A.1). In this work, unless otherwise noted, we use the official, strict interpretation. We calculate the metrics using the official TREC evaluation software.[2]

## A.3 Social impact considerations

By being computationally efficient and foregoing the use of resource-heavy cross-encoders in its pipeline, our proposed contextual similarity learning methods allow top-performing dense retrieval models to be fine-tuned on MS MARCO within 7 hours on a single GPU. We therefore believe that it is well-aligned with the goal of training models in an environmentally sustainable way, the importance of which has been recently acknowledged by the scientific community Information Retrieval and more broadly [6].

Additionally, as we describe in Chapter 3, our framework enables us to optimize dense retrieval models so as to mitigate bias with respect to attributes such as gender, nationality, political ideology and others, and promote neutrality among the top-ranked documents retrieved for a user's query.

On the other hand, the transformer-based Information Retrieval models examined in our study may intrinsically exhibit societal biases and stereotypes. As prior research has discussed [7, 8, 9, 10, 11, 12, 13], these biases stem from the latent biases acquired by transformer-based language models throughout their pre-training, as well as the fine-tuning process on IR collections. Consequently, the practical use of these models might result in prejudiced treatment towards various social groups (e.g., as manifested in their representation or ranking in retrieval result lists). We therefore firmly encourage a mindful and accountable application of these models.

[2] trec.nist.gov/trec_eval/index.html

# Appendix B

# Supplementary material

## B.1 Using a transformer as a contextual document scoring function

We have experimented with more complex, parametric functions $\varphi$ as contextual scoring modules, including feed-forward neural networks and stacks of transformer blocks (see diagram in Figure B.1), which explicitly model inter-document relationships. When we use a transformer encoder as a scoring module, we do not use positional encodings over the input document embeddings, because we require permutation invariance: given fixed positional encodings of ranking order, it would be trivial for the model to learn to assign higher scores for documents of higher rank; at the same time, there is no meaningful sequence order over document embeddings other than relevance ranking.

Although a transformer encoder of 2 blocks was able to marginally outperform the simple function of Equation (2.2), the small improvement (at least as measured on `MS MARCO dev` makes it hard to justify precluding the option of single-stage dense retrieval, as well as the additional computational cost. Future work may more thoroughly investigate the use of a transformer-based scoring function, adding more transformer blocks, which we have not been able to implement due to computational constraints - self-attention incurs a $O(N^2)$ GPU memory dependence on the number of context documents $N$.

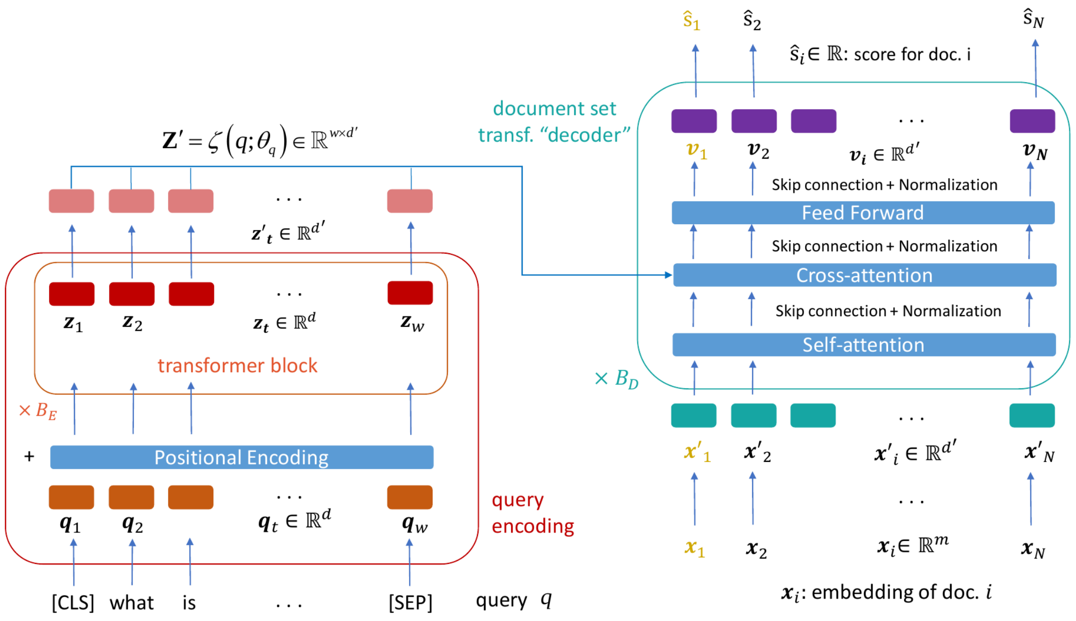


Figure B.1: Schematic diagram of CODER employing a transformer "decoder" as a document set scoring module.

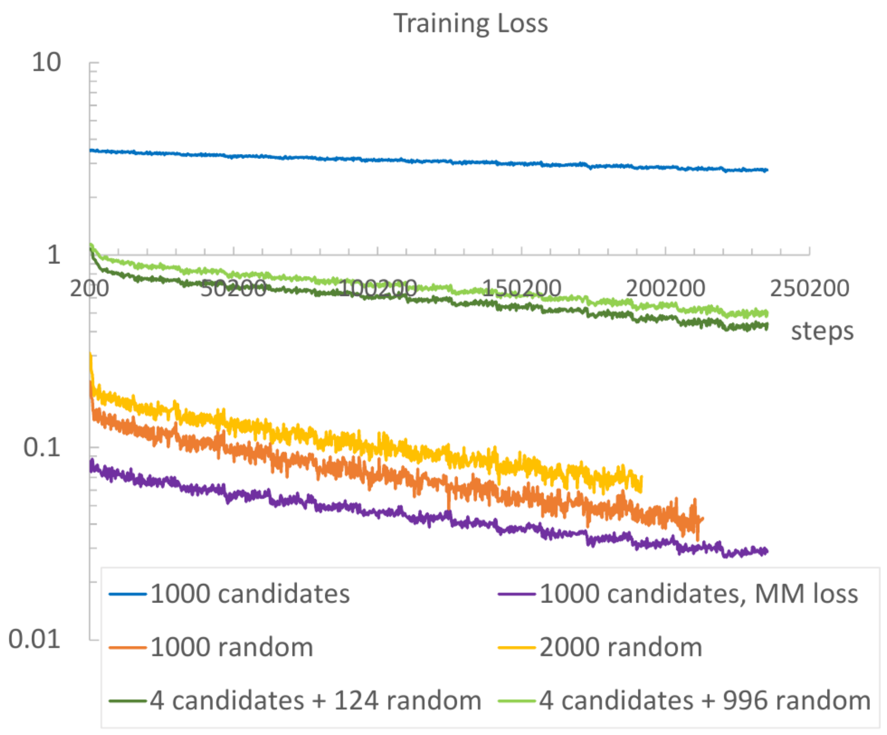


Figure B.2: Evolution of the loss on the training set, as training of BM25→CODER(RepBERT) progresses. Different curves correspond to a different type and number of documents used as negatives during training. While training loss is decreasing in all cases, only using numerous retrieved candidates as negatives, combined with a list-wise KL-divergence loss, results in a significant improvement of performance over the base method (see Figure 2.4) on the validation and test sets.

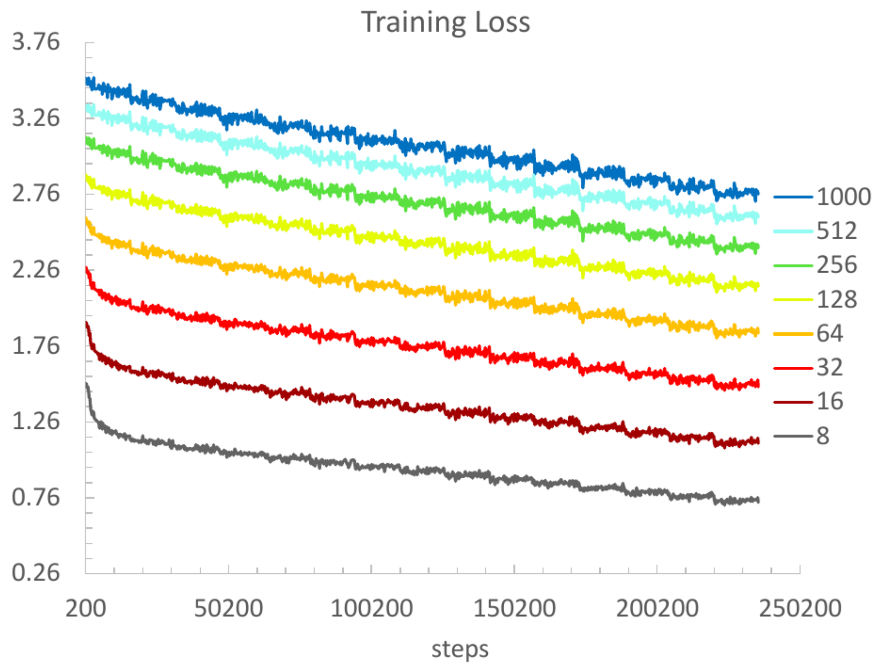


Figure B.3: Evolution of the loss on the training set, as training of BM25→CODER(RepBERT) progresses. Different curves correspond to a different number of BM25 candidates used as negatives during training. While training loss is decreasing in all cases, only using a large number of candidates results in a significant improvement of performance over the base method on the validation and test sets (see Figure 2.4).

## B.2 Inference-time reranking with reciprocal nearest neighbors

Prior work on rNN reranking considered the entire gallery of images (collection $\mathcal{C}$) as a reranking context for each probe, i.e. $N = |\mathcal{C}|$. With $|\mathcal{C}|$ in the order of tens of millions, this is intractable for the task of web retrieval using transformer LMs, and a smaller context size must be used instead. To investigate the importance of the context size, we therefore initially fix the number of in-context candidates per query to a large number within reasonable computational constraints ($N = 1000$) and optimize the hyperparameters of reciprocal nearest neighbors (e.g. $k$, $k_{\text{exp}}$, $\lambda$, $\tau$, $f_w$) on the MS MARCO `dev.small` subset.

We first rerank candidates initially ranked by a CODER-optimized TAS-B retriever, denoted as "CODER(TAS-B)". To determine an appropriate size of reranking context, we first sort candidates by their original relevance score (geometric similarity) and then recompute query similarity scores with a growing number of in-context candidates (selected in the order of decreasing geometric similarity from the query), while measuring changes in ranking effectiveness.

Figure 4.5 shows that rNN-based reranking slightly improves effectiveness compared to ranking purely based on geometric similarity, with the peak improvement registered around a context size of 60 candidates. This behavior is consistent when evaluating rNN-based raranking using the same hyperparameters on different query sets: MS MARCO `dev` (Fig. B.4), which is an order of magnitude larger, and TREC DL 2020 (Fig. B.5) and TREC DL 2019 (Fig. B.6), where the improvement is larger (possibly because it can be measured more reliably due to the greater annotation depth). In all cases performance clearly saturates as the number of candidates grows (somewhat slower for TREC DL 2019). The same behavior as described above is observed when reranking the original TAS-B model's results using the same hyperparameters chosen for the CODER-trained version, with the performance benefit being approximately twice as large (Fig. B.7).

While it is expected that progressively increasing the context size will increase performance, as there is a greater chance to include the ground-truth passage(s) which may have been initially ranked lower (i.e. embedded farther from the query), the peak and subsequent degradation or saturation is a novel finding. We hypothesize that it happens because the more negative candidates are added in the context, the higher the chance that they disrupt the reciprocal neighborhood relationship between query and positive document(s) (see Figure 4.1).

We can therefore conclude that we may use a relatively small number $N$ of context candidates for computing reciprocal nearest neighbors similarities, which is convenient because computational complexity scales with $O(N^2)$. For a context of 60 candidates, a CPU processing delay of only about 5 milliseconds per query is introduced (Figure 4.4). These results additionally indicate that the context size should best be treated as a rNN hyperparameter to be jointly optimized with the rest, which is reasonable, as it is expected to depend on the average rank that ground-truth documents tend to receive.

After optimizing rNN-related hyperparameters (including the context size) on MS MARCO `dev.small` for CODER(TAS-B), we evaluate rNN reranking on the other evaluation sets (including its $\times 8$ larger superset MS MARCO `dev`) and present the results in Table 4.2. We observe that a similarity based on reciprocal nearest neighbors can indeed improve ranking

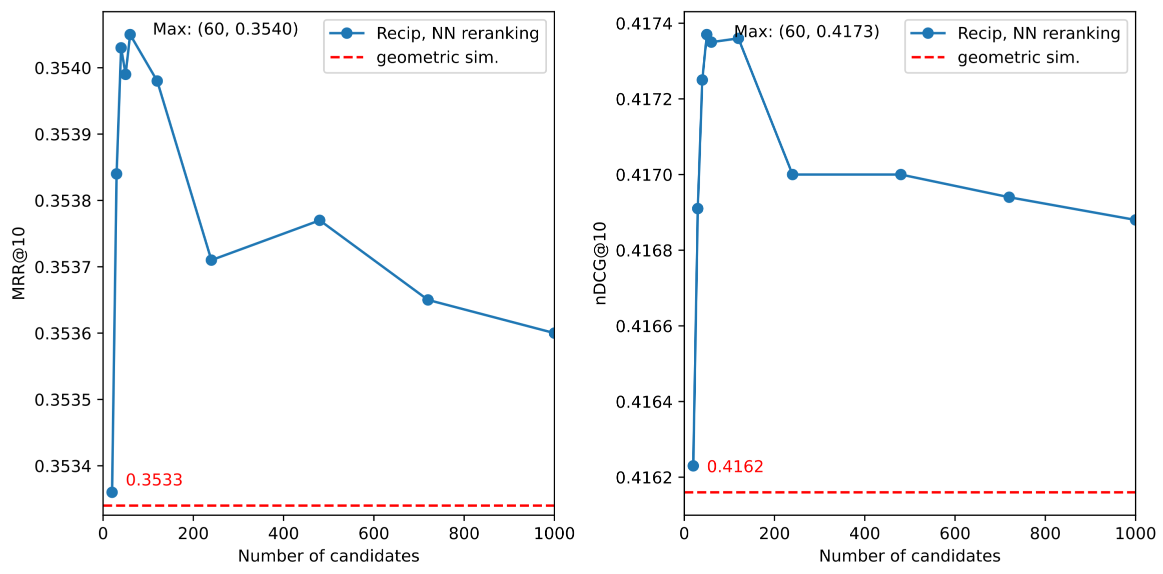


Figure B.4: Performance of reciprocal nearest neighbors-based reranking of CODER(TAS-B) results on MS MARCO dev, as the number of candidates in the ranking context grows. Hyperparameters are the same as in Fig. 4.5. Performance is slightly improved compared to ranking exclusively based on geometric similarity and peaks at 60 in-context candidates.

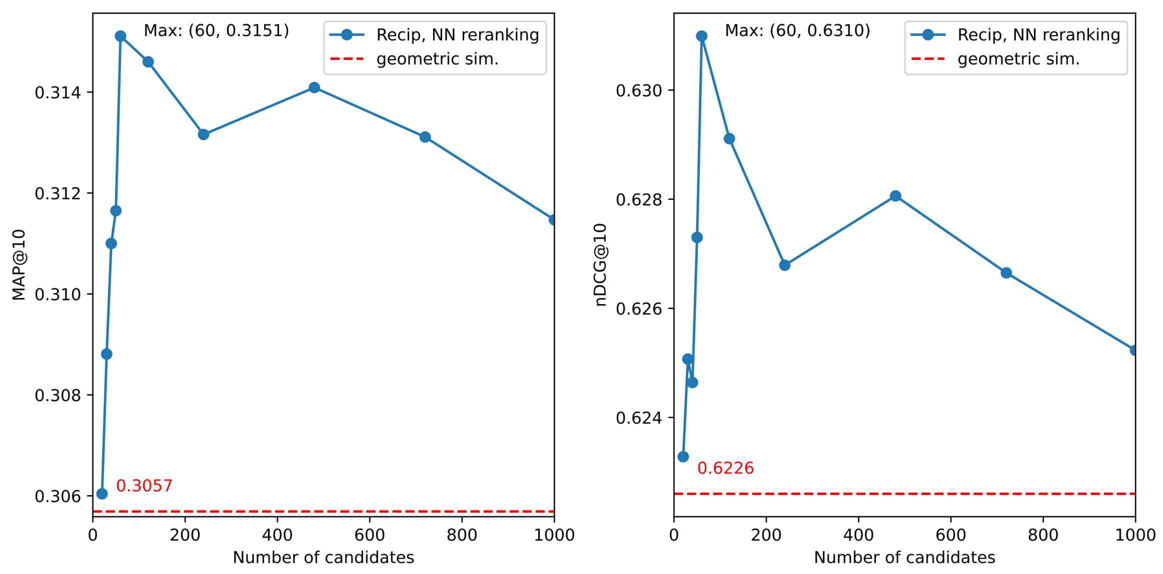


Figure B.5: Performance of reciprocal nearest neighbors-based reranking of CODER(TAS-B) results on TREC DL 2020, as the number of candidates in the ranking context grows. Hyperparameters are the same as in Fig. 4.5. Performance is improved compared to ranking exclusively based on geometric similarity and peaks at 60 in-context candidates.

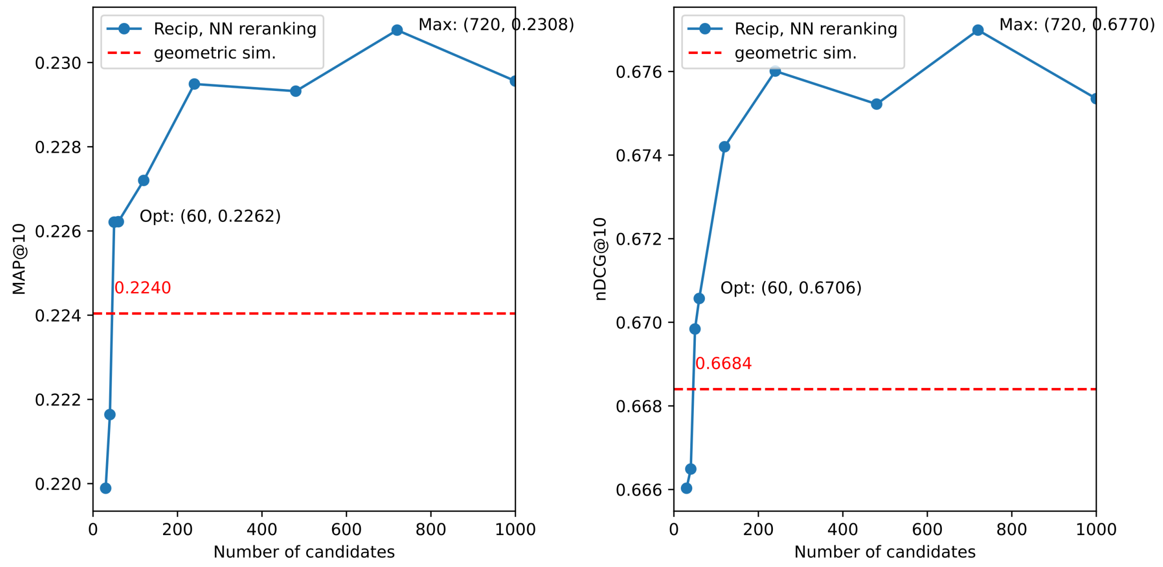


Figure B.6: Performance of reciprocal nearest neighbors-based reranking of CODER(TAS-B) results on TREC DL 2019, as the number of candidates in the ranking context grows. Hyperparameters are the same as in Fig. 4.5. Performance is improved compared to ranking exclusively based on geometric similarity but does not clearly saturate.

effectiveness compared to using purely geometric similarity. The improvement is more pronounced on the TREC DL datasets (+0.011 nDCG@10), where a greater annotation depth and multi-level relevance labels potentially allow to better differentiate between methods.

Additionally, we find that rankings from TAS-B – whose embeddings are relatively similar to CODER(TAS-B) – also improve, despite the fact that hyperparameters were chosen based on the CODER(TAS-B) model (also see Figure B.7).

The strongest dense retrieval models we evaluate, CoCondenser and CODER(CoCondenser), also show improved performance, again measured primarily on TREC DL: the former improves by +0.009 nDCG@10 on TREC DL 2020 and the latter by 0.009 nDCG@10 on TREC DL 2019. Notably, reranking effectiveness when using the exact same hyperparameters as for CODER(TAS-B) and TAS-B is only very slightly worse.

By contrast, when transferring hyperparameters selected for MS MARCO to reranking candidates on the TripClick dataset, we find that performance deteriorates with respect to geometric similarity. Therefore, we can conclude that rNN hyperparameters predominantly depend on the dataset, and to a much lesser extent on the dense retriever.

After optimizing hyperparameters on TripClick `HEAD Val`, we evaluate on `HEAD Test`, using both RAW (binary) as well as DCTR (multi-level) relevance labels; we present the results in Table 4.3. Also for this dataset, which differs substantially in characteristics from MS MARCO, we again observe that using reciprocal nearest neighbors to compute the similarity metric can slightly improve ranking effectiveness for all examined retrieval methods. We also observe the same saturation behavior with respect to the ranking context size, i.e. the number of candidates considered when reranking (Fig. B.8).

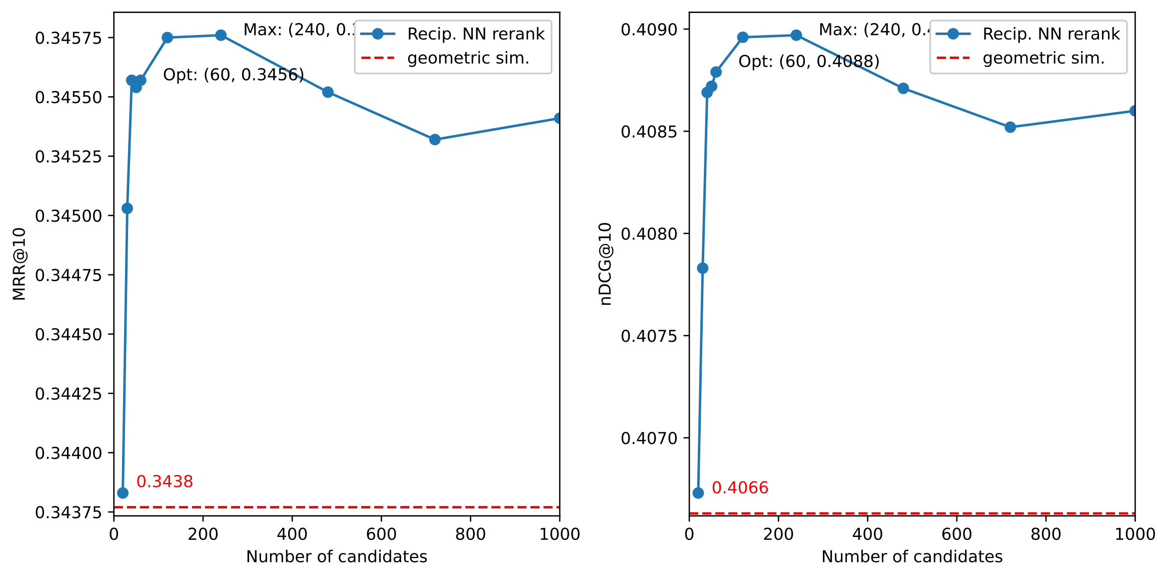


Figure B.7: Performance of reciprocal nearest neighbors-based reranking of TAS-B results on MS MARCO dev, as the number of candidates in the ranking context grows. Hyperparameters are the same as in Fig. 4.5. Performance is improved compared to ranking exclusively based on geometric similarity and peaks at approx. 60 in-context candidates.

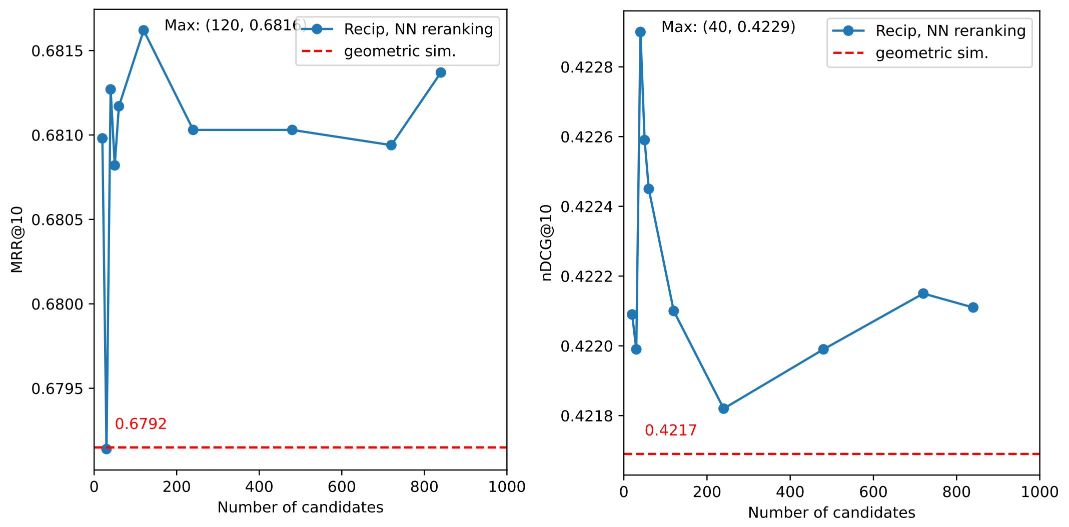


Figure B.8: Performance of reciprocal nearest neighbors-based reranking of CODER(RepBERT) results on TripClick `HEAD Test`, as the number of candidates in the ranking context grows.

## B.3 Evidence-based label smoothing

| Test: | DCTR Head | | | RAW Head | | |
|---|---|---|---|---|---|---|
| **Model** | **MRR@10** | **nDCG@10** | **Recall@10** | **MRR@10** | **nDCG@10** | **Recall@10** |
| TAS-B | 0.278 | 0.139 | 0.130 | 0.339 | 0.188 | 0.113 |
| CODER(TAS-B) | 0.279 | 0.140 | 0.130 | 0.338 | 0.191 | 0.115 |
| CODER(TAS-B) + geom. smooth labels | 0.285 | 0.143 | **0.134** | 0.344 | **0.195** | **0.116** |
| CODER(TAS-B) + rNN smooth labels | **0.288** | **0.144** | **0.134** | **0.347** | **0.195** | **0.116** |
| **CODER(TAS-B) + mixed rNN/geom. smooth lab.** | 0.284 | 0.142 | 0.132 | 0.342 | 0.192 | 0.115 |
| CoCondenser | 0.242 | 0.114 | 0.105 | 0.293 | 0.157 | 0.092 |
| CODER(CoCondenser) | 0.251 | 0.117 | 0.107 | 0.306 | 0.161 | 0.093 |
| **CODER(CoCondenser) + mixed rNN/geom. smooth lab.** | 0.250 | 0.117 | 0.107 | 0.304 | 0.162 | 0.094 |

Table B.1: Performance of models trained on MS MARCO but zeroshot-evaluated on TripClick Test. **Bold**: overall best, Underline: best in model class.

| Val: | DCTR Head | | | RAW Head | | |
|---|---|---|---|---|---|---|
| **Model** | **MRR@10** | **nDCG@10** | **Recall@10** | **MRR@10** | **nDCG@10** | **Recall@10** |
| TAS-B | 0.299 | 0.145 | 0.136 | 0.355 | 0.200 | 0.118 |
| CODER(TAS-B) | **0.300** | 0.146 | 0.140 | 0.353 | 0.203 | 0.121 |
| CODER(TAS-B) + geom. smooth labels | 0.297 | **0.147** | 0.140 | 0.355 | 0.204 | 0.121 |
| CODER(TAS-B) + rNN smooth labels | **0.300** | **0.147** | **0.141** | **0.357** | **0.205** | **0.122** |
| **CODER(TAS-B) + mixed rNN/geom. smooth lab.** | 0.299 | **0.147** | **0.141** | 0.355 | 0.204 | **0.122** |
| CoCondenser | 0.247 | 0.115 | 0.105 | 0.308 | 0.167 | 0.097 |
| CODER(CoCondenser) | 0.254 | 0.120 | 0.111 | 0.314 | 0.173 | 0.101 |
| **CODER(CoCondenser) + mixed rNN/geom. smooth lab.** | 0.254 | 0.118 | 0.109 | 0.311 | 0.169 | 0.098 |

Table B.2: Performance of models trained on MS MARCO but zeroshot-evaluated on TripClick Val. **Bold**: overall best, Underline: best in model class.

Query: *"iatrogenic infection definition"*

| rank | docID | text |
|---|---|---|
| 1 | 1423013* | Iatrogenic infection: Related Diseases. Iatrogenic infection: Iatrogenic infection is listed as a type of (or associated with) the following medical conditions in our database: 1 Hospital or surgery-related conditions.hese medical condition or symptom topics may be relevant to medical information for Iatrogenic infection: 1 Iatrogenic. 2 Iatrogenic disease. 3 Iatrogenic disorder. 4 Infection (1293 causes). 5 Infection symptoms (1293 causes) |
| 2 | 1423019 | These medical disease topics may be related to Iatrogenic infection: 1 radiation therapy-orchemotherapy. 2 fibromyalgia. 3 dissociative identity disorder. 4 bipolar disorder. 5 somatoform disorder.hese medical condition or symptom topics may be relevant to medical information for Iatrogenic infection: 1 Iatrogenic. 2 Iatrogenic disease. 3 Iatrogenic disorder. 4 Infection (1293 causes). 5 Infection symptoms (1293 causes) |
| 3 | 1423015 | Confidence votes 407. An iatrogenic infection is one actually caused by accidental medical actions. Iatrogenic means a complication as a result of treatment.Iatrogenic means that was caused by a doctor, or by a treatment prescribed by a doctor.plit and merge into it. Answer by Vtel57. Confidence votes 407. An iatrogenic infection is one actually caused by accidental medical actions. Iatrogenic means a complication as a result of treatment. Iatrogenic means that was caused by a doctor, or by a treatment prescribed by a doctor |
| 4 | 741411 | Causes. Iatrogenic infection is complex because it has so many causes, including chance, negligence, medical error, and interactions of prescription drugs. Nosocomial infections, another leading cause of iatrogenic illness, are those that occur during hospitalization or through treatment in another health care setting. Vectors for infection in these facilities include vomit, blood, urine, and feces. Some microorganisms can be spread through the air |
| 5 | 769742 | These diseases could be caused by a number of things, and in some cases they are more of an effect or symptom than a full-on disease. A complication after surgery or another medical procedure could be classified as an iatrogenic disease |
| 6 | 1423014 | Disease Topics Related To Iatrogenic conditions. Research the causes of these diseases that are similar to, or related to, Iatrogenic conditions: 1 Radiation therapyorchemotherapy. 2 Fibromyalgia. [...] |
| 7 | 1499919 | caused by treatment or diagnostic procedures. An iatrogenic disorder is a condition that is caused by medical personnel or procedures or that develops through exposure to the environment of a health care facility. See also nosocomial. iatrogenesis, iatrogeny, |
| 8 | 1423010 | iatrogenic. adjective Referring to a physical or mental condition caused by a physician or health care provider, eg, iatrogenic disease, due to exposure to pathogens, toxins or injurious treatment or procedures.aused by treatment or diagnostic procedures. An iatrogenic disorder is a condition that is caused by medical personnel or procedures or that develops through exposure to the environment of a health care facility |
| 9 | 1499918 | iatrogenic adjective Referring to a physical or mental condition caused by a physician or health care provider, eg, iatrogenic disease, due to exposure to pathogens, toxins or injurious treatment or procedures |

Table B.3: Top documents selected by evidence-based label smoothing to receive higher than zero target relevance probability. Although only the 1st document (ID: 1423013) is annotated as relevant in the dataset, we observe that most selected documents are relevant (false negatives), and in fact more informative and relevant than the ground truth. Documents in ranks 2 and 6, albeit non-relevant, scored highly due to their pronounced similarity to a non-relevant part of the ground-truth document.

Query: *"how to calculate demolition costs"*

| rank | docID | text |
|---:|---:|---|
| 1 | 2513851* | Instructions. Mesure the square footage of the structure you wish to demolish using a tape measure. Multiply the length and the width of the structure to arrive at its area, which is represented in units squared.For example, if your structure is 10 feet wide and 14 1/2 feet long, you will arrive at the sum of 145 feet. You will then have to pay for 145 square feet of demolition.View the structure and determine its number of floors and the building material it's made of. Additional floors will add cost to demolition, and various building materials will have different demolition costs per square foot. [...] |
| 2 | 2221219 | Instructions. Mesure the square footage of [...] Additional floors will add cost to demolition, and various building materials will have different demolition costs per square foot. |
| 3 | 2600307 | Multiply the length and the width of the structure to arrive at its area, which is represented in units squared. For example, if your structure is 10 feet wide and 14 1/2 feet long, you will arrive at the sum of 145 feet. You will then have to pay for 145 square feet of demolition.View the structure and determine its number of floors and the building material it's made of. Additional floors will add cost to demolition, and various building materials will have different demolition costs per square foot.rovide the structural information and ask for a quote from each demolition expert. Ask if they add hauling and landfill fees into their quote. An average cost of residential demolition was \$6 to \$15 per square foot, as of 2010. |
| 4 | 7171531 | Calculating the cubic yards of demolition debris is simple and involves converting the cubic footage of the structure to cubic yards while also accounting for the air space in the building (0.33). For our 2,000 sq. ft. house example, let's assume our home is 2 stories and the dimensions are 40 ft. x 25 ft. |
| 5 | 194518 | Measure the square footage of your driveway by multiplying its length by its width. For example, the square footage of a 40 ft. long x 20 ft. wide driveway is 800 square feet. In this example, a typical demolition cost would range from \$800 to \$2,400.ther cost factors to consider include permits and inspections. The national average cost to demolish a concrete driveway is \$1,500, but the price can top \$3,000 in some cases. |
| 6 | 5517379 | There's not a standard per-square-foot-cost to go by, so be sure to check with at least one other demolition contractor in your area to see if it can beat the quoted cost of a competing company. EXAMPLE: A 1,500 square foot house will typically cost \$6,000 to \$22,500 to demolish. The wide gap in price is due to the multitude of factors that influence demolition costs (e.g., location, asbestos abatement, etc.). |
| 7 | 783054 | Measure the square [...] to \$2,400. |
| 8 | 4724391 | Measure the square [...] to \$2,400. For curvy and/or multi-width driveways, the calculation is a bit more complicated. |
| 9 | 5567534 | Measure the square [...] to \$2,400. |

Table B.4: Top documents selected by evidence-based label smoothing to receive higher than zero target relevance probability. Although only the 1st document (ID: 2513851) is annotated as relevant in the dataset, we observe that most selected documents are relevant (false negatives). Documents in ranks 1, 2 and 5, 7, 8, 9 are near-duplicates.

Query: *"what cause you to lose your taste and smell"*

| rank | docID | text | Relev. |
|---|---|---|---|
| 1 | 3857566 | There are many causes behind loss of smell and taste. One of the main causes is aging, which brings on degeneration of nerve cells that control smell and taste buds. Other causes include excessive smoking, nutritional deficiencies, certain nervous system diseases, radiation therapy, fever, blocked nasal passages, sinusitis, viral or upper respiratory infections, and gum diseases. | True |
| 2 | 3857574 | Another cause of loss of smell and taste may be upper viral or respiratory infections. Sinusitis or blocked nasal passages cause a blockage in air flow; this reduces the amount of aromas reaching the smell receptors. | True |
| 3 | 8173187 | Anything that interferes with these processes, such as nasal congestion, nasal blockage, or damage to the nerve cells themselves, can lead to loss of smell. The ability to smell also affects our ability to taste. Without the sense of smell, our taste buds can only detect a few flavors, and this can affect your quality of life. Anosmia Causes Nasal congestion from a cold, allergy, sinus infection, or poor air quality is the most common cause of anosmia. | True |
| 4 | 3857571 | This may be due to a degeneration of the nerve cells which control smell, together with a loss of sensitivity in one's taste buds. Men, regardless of age have a lower ability to distinguish between odors. Smoking causes damage to the nasal membranes and reduces one's ability to identify odors. Foods will also become tasteless as a result of smoking and one may even lose his/her ability to smell aromas. Certain nervous system diseases or radiation treatment for cancer patients may result in a lack of taste and smell. | True |
| 5 | 732708 | Sometimes loss of taste and smell contributes to depression. Loss of taste and smell also might tempt you to use excess salt or sugar on your food to enhance the taste, which could be a problem if you have high blood pressure or diabetes. If you're experiencing loss of taste and smell, consult your doctor. Although you can't reverse age-related loss of taste and smell, some causes of impaired taste and smell are treatable. For example, your doctor might adjust your medications if they're contributing to the problem. | False |
| 6 | 7111647* | Sometimes loss of taste and smell contributes to depression. Loss of taste and smell also might tempt you to use excess salt or sugar on your food to enhance the taste, which could be a problem if you have high blood pressure or diabetes. If you're experiencing loss of taste and smell, consult your doctor. Although you can't reverse age-related loss of taste and smell, some causes of impaired taste and smell are treatable. For example, your doctor might adjust your medications if they're contributing to the problem. Many nasal and sinus conditions and dental problems can be treated as well. | False* |
| 7 | 4993775 | 1 A gradual loss of smell and taste may be due to a cold/flu or sinus infection. 2 If after the cold or sinus infection has cleared up, your sense of smell and taste has not returned, it is important that you visit your GP so that he/she can identify any underlying cause for your loss of smell or taste. Smoking causes damage to the nasal membranes and reduces one's ability to identify odors. 2 Foods will also become tasteless as a result of smoking and one may even lose his/her ability to smell aromas. 3 Certain nervous system diseases or radiation treatment for cancer patients may result in a lack of taste and smell. | True |
| 8 | 732707 | Various other factors also can contribute to loss of taste and smell, however, including: 1 Nasal and sinus problems, such as allergies, sinusitis or nasal polyps. 2 Certain medications, including beta blockers and angiotensin-converting enzyme (ACE) inhibitors. 3 Dental problems. 4 Cigarette smoking. 5 Head or facial injury. | True |
| 9 | 4453402 | Nose or sinus problems might make you lose your sense of smell, for a little while or even a long time. Your sinuses might be swollen or polyps (tiny growths) might block your nose passages. Infections (like colds or flu) or a head injury might also make you lose your ability to smell. Parkinson's disease or Alzheimer's disease can make people lose their sense of smell. Infection or inflammation in your mouth can cause loss of taste. (Inflammation means redness and swelling.) Head injury and Bell's palsy can also affect the ability to taste. | True |

Table B.5: Top documents retrieved by a CODER(TAS-B) model trained through EB label smoothing. Here, the only *non-relevant* documents are in fact the only document annotated as relevant in the dataset (marked with *, ID: 7111647) and its duplicate in rank 5 - all other documents are actually more relevant and informative. Yet, because in the original model's results the ground-truth positive was ranked 3rd, its demotion to rank 6 would decrease MRR by half, although fully justified.

# Bibliography

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268 [cs]*, October 2018. URL http://arxiv.org/abs/1611.09268. arXiv: 1611.09268.

[2] Navid Rekabsaz, Oleg Lesota, Markus Schedl, Jon Brassey, and Carsten Eickhoff. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2507–2513. Association for Computing Machinery, New York, NY, USA, July 2021. ISBN 978-1-4503-8037-9. URL https://doi.org/10.1145/3404835.3463242.

[3] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv:2007.00808 [cs]*, October 2020. URL http://arxiv.org/abs/2007.00808. arXiv: 2007.00808.

[4] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512, Virtual Event Canada, July 2021. ACM. ISBN 978-1-4503-8037-9. doi: 10.1145/3404835.3462880. URL https://dl.acm.org/doi/10.1145/3404835.3462880.

[5] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and A. Hanbury. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *SIGIR*, 2021. doi: 10.1145/3404835.3462891.

[6] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. Reduce, reuse, recycle: Green information retrieval research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2825–2837, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3531766. URL https://doi.org/10.1145/3477495.3531766.

[7] Gizem Gezici, Aldo Lipani, Yucel Saygin, and Emine Yilmaz. Evaluation metrics for measuring bias in search engine results. *Information Retrieval Journal*, 24(2):85–113, 2021.

[8] Navid Rekabsaz, Simone Kopeinik, and Markus Schedl. Societal biases in retrieved contents: Measurement framework and adversarial mitigation for bert rankers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[9] Navid Rekabsaz and Markus Schedl. Do neural ranking models intensify gender bias? In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2065–2068, 2020.

[10] Amin Bigdeli, Negar Arabzadeh, Shirin Seyedsalehi, Morteza Zihayat, and Ebrahim Bagheri. A light-weight strategy for restraining gender biases in neural rankers. In Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty, editors, *Advances in Information Retrieval*, pages 47–55, Cham, 2022. Springer International Publishing. ISBN 978-3-030-99739-7.

[11] Klara Krieg, Emilia Parada-Cabaleiro, Markus Schedl, and Navid Rekabsaz. Do perceived gender biases in retrieval results affect relevance judgements? In *Proceedings of the European Conference on Information Retrieval, Workshop on Algorithmic Bias in Search and Recommendation (ECIR-BIAS 2022)*, pages 104–116, Cham, 2022. Springer.

[12] Amin Bigdeli, Negar Arabzadeh, Morteza Zihayat, and Ebrahim Bagheri. Exploring gender biases in information retrieval relevance judgement datasets. In Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani, editors, *Advances in Information Retrieval*, pages 216–224, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72240-1.

[13] Alessandro Fabris, Alberto Purpura, Gianmaria Silvello, and Gian Antonio Susto. Gender stereotype reinforcement: Measuring the gender bias conveyed by ranking algorithms. *Information Processing & Management*, 2020.