Impact Analysis of Hard Negative Sample Rankings in ANCE Training

Hojae Son
University of Massachusetts Amherst
Amherst, MA, USA
hojaeson@umass.edu

Deepesh Suranjandass
University of Massachusetts Amherst
Amherst, MA, USA
dsuranjandas@umass.edu

ABSTRACT

We investigate the impact of hard negative sampling strategies in the ANCE [11] (Approximate Nearest Neighbor Negative Contrastive Learning) framework by analyzing how the ranking positions of negative samples affect model convergence and generalization. Our hypothesis suggests that the degree of negative sample hardness significantly influences training dynamics and model performance. Using a subset of MS MARCO, we conduct experiments comparing different ranking segments for negative sampling to understand their impact on training efficiency and model effectiveness.

ACM Reference Format:
Hojae Son and Deepesh Suranjandass. 2024. Impact Analysis of Hard Negative Sample Rankings in ANCE Training. In *Proceedings of Information Retrieval Course Project (Project Final Report)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 BACKGROUND

ANCE (Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval) [11] suggests an effective approach for optimizing indexer as well as fine-tuning Language model interactively. The mechanism of ANCE is built on two independent processes.One pipeline is an indexer based on the Faiss library [1] that retrieves the top K vectors given a query vector. As the indexer looks up the similar vectors to query vector, the query, and documents need to be represented with vectors to comprehend their similarities with high dimensions. Thereby, another pipeline, Trainer, takes the responsibility of translating natural language into vector embedding. ANCE uses a language model, Roberta [5], as an encoder to carry out the task of converting English passages to vector embeddings with contrastive loss [2] to carry out the abilities to differentiate the vectors during the training phase.

1.1 Warm Up

At the initial phase, ANCE uses an initial set of negative sample documents using BM25, a traditional lexical model as there is no negative sample retriever established. As the target of ANCE is fine-tuning a language model, Roberta [5] model, it has a certain Warm-Up phase with the relevant / non-relevant query-document pairs from BM25 model. We use same setting to warm up the model increasing learning rate for 5,000 steps to initial learning rate 1e-6.

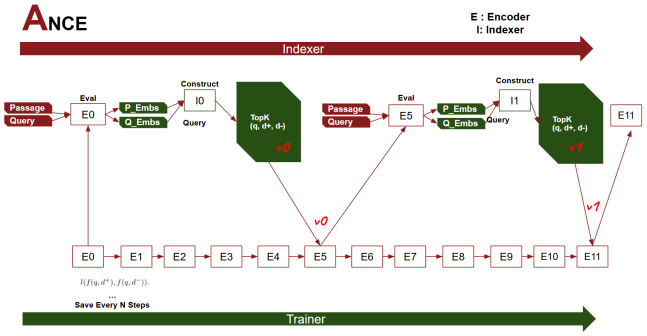

Figure 1: ANCE pipeline

1.2 Indexer & Trainer

As Figure 1 illustrates the overview of ANCE, the following items are brief explanation of Indexer & Trainer pipelines

1.2.1 Indexer. With a Warmed-Up model, indexer pipeline starts inferencing the entire MSMarco passage & query dataset to encode language data into vector embeddings. With the acquired and inferenced passage embeddings, Faiss indexer builds an index. With the query vector embeddings, the index looks up and retrieves Top K similar vector passages within the vector space. For a note, as MSMarco query dataset contains 502,939 queries, ANCE authors use a factor parameter to use the portion of queries at a time. For example, if the factor is 5, Indexer looks up 20% of queries to retrieve Top K vectors. After retrieving the vectors, is uses the vectors as negative samples for each query. As MSmarco dataset only contains a single positive passage to each query, the retrieved passage vectors are negative samples except for a positive passage and these vectors are considered as hard negative samples. The main contribution of ANCE comes with the hard negative samples as the encoder should learn the boundary between positive and negative samples and comprehend the subtle difference. Indexer pipeline saves the generated nagative samples in hard-disk for Trainer pipeline to pick up the better set of dataset to learn. ANCE authors call this process as a single refreshing.

1.2.2 Trainer. Starting from the pre-trained (warm-up) version of model encoder, encoder loads the latest dataset which consists of set of query, a positive passage, a set of negative passages. During

Project Final Report , December 2024, University of Massachusetts Amherst

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

training, the trainer loads triplet of dataset (query, pos passage, negative passage) and then, every n steps, it saves the checkpoints for indexer pipeline to pick-up the latest checkpoints as these pipelines are independent asynchronous processes.

1.2.3 *Summary of Architecture.*

- Asynchronous Inference & Indexing:
  - Inference the entire passage
  - Build index with passage
  - Retrieve hard negative samples
  - Generate dataset with the retrieved samples
- Asynchronous Training:
  - Training encoder with latest dataset from Indexer
  - If new dataset comes, it uses new dataset

2 RELATED WORK

Our work builds upon several key areas in information retrieval and negative sampling strategies.

2.1 Dense Retrieval Models

ANCE [11] introduced an innovative approach to dense retrieval using asynchronous negative sampling. Their method employs periodic index refreshing and dynamic negative mining, demonstrating significant improvements over traditional approaches. The dual-encoder architecture with BERT embedding has shown promising results in capturing semantic relationships between queries and documents.

2.2 Negative Sampling Techniques

Research in negative sampling has evolved significantly:

- **Traditional Methods:**
  - DPR [3] introduced in-batch negative sampling
  - RocketQA [9] demonstrated cross-batch negative mining
  - NPRINC [6] showed improved negative contrast methods
- **Advanced Strategies:**
  - STAR [12] explored hard negative mining impact
  - Metric learning approaches in ADORE [13]
  - Analysis of sampling strategies in RepCONC [4]

2.3 Training Dynamics

Recent research has focused on understanding and improving training efficiency:

- Studies on optimal batch composition [8]
- Investigation of loss function designs [7]
- Analysis of convergence patterns in dense retrievers

Our work extends these findings by specifically investigating how the selection of negative samples based on their ranking positions affects model performance and training dynamics. While previous work has focused on finding hard negatives, we explore the balance between hard and easy negatives, providing insights into optimal training strategies.

3 PROBLEM STATEMENT

ANCE [11] authors propose an influential approach fine-tuning encoder with hard negative sampling with indexer. In this project, we identify one potential spot to improve. Since Indexer uses the evaluation mode of the encoder, it takes one third of memory usage to inference over Trainer side. With inferencing the entire passage, 8,841,823 of MSMarco passages takes long hours. For example, we allocate the GPU resources 2:1 on the trainer and indexer each, one of the allocating ratio from the [11] with 4 A40 GPUs and 256 mini-batch size, one iteration takes about 7 seconds which means a single refreshing takes 16 hours.

$$\frac{8,841,823}{4 \times 256} \times 7 = 60,527.91$$

On the other hand, trainer can only load 16 mini batch size because the encoder needs to back-propagate loss and store the gradients which lead more memory space along with smaller mini batch size.

$$\frac{8,841,823}{4 \times 16} \times 7 = 967,074.39$$

With the estimation above, it takes about 11.19 days to iterate the whole dataset for the Trainer. Therefore, it's crucial that the trainer actually see what features during the training phase. Additionally, there is a potential risk with this huge gap of timing, indexer takes the encoder that actually sees a small part of the dataset. To address this, in our project, we propose controlling the indexer to manage the selection of negative samples and analyze how loss changes. This approach aims to get better strategies to establish the Trainer and Indexer and optimize training efficiency while reducing computational overhead. For a query-document pair (q, d), relevance score is computed as:

$$s(q, d) = E_Q(q)^T \cdot E_D(d) \tag{1}$$

During training, for each query q, ANCE retrieves top-k documents as candidates for negative sampling:

$$D_k(q) = \{d_1, ..., d_k\} \text{ ranked by } s(q, d) \tag{2}$$

The training process involves two interlinked components:

- Trainer: Updates encoder parameters using selected negative samples
- Inferencer: Generates document rankings using latest encoders

4 PROPOSED APPROACH

Our approach for configuring the selection of negative samples mainly lies on Indexer. We retrieve top 200 document vector embeddings and vary the selection of negative samples as follows.

- Top 20
- Top 20 + Bottom 20
- Bottom 20
- Randomized 20

During the fine-tuning phase and analyzing different segments within these retrieved documents. We formulate distinct strategies for top-ranked documents, bottom-ranked documents and randomized documents.

Our implementation builds upon:

(1) ANCE (passage) codebase [10] as baseline
(2) Modified negative sampling approach based on rank positions

4.1 Implementation Strategy

We segment retrieved documents based on their ranking positions:

Algorithm 1 Modify GenerateNegativePassageID

```
procedure GenerateNegativePassageID(query, top_k)
    Retrieve top-k documents for query
    Extract ranking scores and positions
    if using_top_negatives then
        segment ← documents[:20]
    else if using_bottom_negatives then
        segment ← documents[-20:]
    else if using_top+bottom_negatives then
        segment
        ← documents[:20] + documents[-20:]
    else if using_random_negatives then
        segment ← random_segment
    end if
    return segment
end procedure
```

- Top-ranked segment (ranks 0-20):
  - Highest semantic similarity to query
  - Potentially hardest negative examples
  - Close to decision boundary
  - Critical for fine-grained discrimination
- Bottom-ranked segment (ranks 180-200):
  - Lower semantic similarity
  - Relatively easier negatives
  - Clear semantic distinction
  - Baseline confidence calibration
- Top + Bottom-ranked segment (ranks 0-20 + ranks 180-200)
- Randomly ranked segment (ranks 0-20 + ranks 180-200)

5 EXPERIMENTS

5.1 Dataset

Given the limited timeline, we create a subset of the MS MARCO dataset. First, we randomly shuffle the raw passage dataset (collections.tsv) and sample 10% of the passages. During preprocessing, we include only query-relevant documents within the subset of the raw collection. Using a subset increases the likelihood of including positive documents, which improves precision. Thus, it is essential to monitor the trends in loss and accuracy throughout the training phase.

- Training Set:
  - subset of 884k passages
  - subset of 502k training queries
- Development Set:
  - subset of 6,980 queries
  - Enables direct comparison with ANCE

Dataset available at: https://microsoft.github.io/msmarco/

5.2 Experimental Setup

Our implementation consists of two main components:

5.2.1 Data Processing Pipeline. We implement a rank-aware data loading system that supports selective sampling based on document positions:

- Query Processing:
  - Tokenization and encoding
  - Embedding generation
  - Score computation
- Document Processing:
  - Rank-based segmentation
  - Selective sampling
  - Batch construction
- Training Pipeline:
  - Dynamic negative selection
  - Rank-aware batch formation
  - Efficient data loading

Algorithm 2 Dataset and Negative Sample Processing

```
procedure PrepareDataset
    Input: Full MS MARCO collection
    Load collection from TSV file
    Create 10% subset using random sampling
    Preserve query distribution in subset
    return processed_subset
end procedure
```

5.2.2 Finetuning Configuration for Encoder. We use a pretrained model from the ANCE repository, allocating 30 GB of the available 48 GB from the 4 A40 GPUs for the encoder and 13 GB for the inferencer, with the following settings:

- Learning rate: 1e-6 (following original paper)
- Base Model: Roberta (125M)
- Model type: rdot_nll
- Max sequence length: 512
- Batch size per GPU: 16
- Gradient accumulation steps: 8
- Optimizer: LAMB
- Training(Encoding) steps:
  - Warmup: 5000 steps
  - Checkpointing: every 2000 steps
  - Logging: every 300 steps
- Inference settings:
  - Batch size per GPU: 256
  - Top-k candidates for training: 200
  - Number of negative samples: 20
  - Evaluation metric: Top-k NDCG/MRR

5.3 Evaluation Framework

We focus on three key aspects:

5.3.1 Ranking Quality.

- NDCG@k:

$$nDCG@k = \frac{\sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)}}{IDCG@k} \tag{3}$$

where rel_i is document relevance at position i

- InfoNCE Loss:

$$\mathcal{L} = -\log \frac{\exp(s(q, d^+)/\tau)}{\exp(s(q, d^+)/\tau) + \sum_{d^- \in D_{neg}} \exp(s(q, d^-)/\tau)} \quad (4)$$

5.3.2 *Training Analysis.*

- Model Behavior:
  - Convergence patterns by rank segment
  - Loss stability characteristics
  - Gradient magnitude analysis
  - Parameter update distributions
- Performance Impact:
  - Query type stratification
  - Hard vs easy query analysis
  - Generalization metrics
  - Error pattern analysis

5.3.3 *Resource Utilization.*

- Computational Efficiency:
  - Training time comparison
  - Memory usage patterns
  - GPU utilization metrics
  - Scaling characteristics

5.4 Baselines

Our baseline implementation involves a crucial warm-up phase followed by negative sampling experiments:

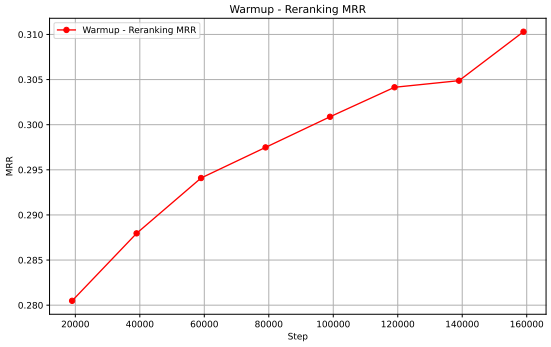

Figure 2: MRR Progression During Warm-up Phase

- **Warm-up Phase (0-150,000 steps):**
  - Pretrained ANCE BM25 model initialization
  - Learning rate: 2e-4 with standard top-20 negatives
  - Progressive MRR improvement from 0.280 to 0.310 (Fig. 2)
  - Establishes stable foundation for strategy comparison
- **Configuration Details:**
  - Training: Triplet loss, LAMB optimizer, batch size 32
  - Inference: Batch size 256, top-200 retrieval pool

After warm-up completion at 150,000 steps, we implement four sampling strategies: Top 20, Bottom 20, Top 20 + Bottom 20, and Random 20, all starting from this common baseline.

5.5 Results

Our experimental results after the warm-up phase demonstrate distinct patterns across different negative sampling strategies.

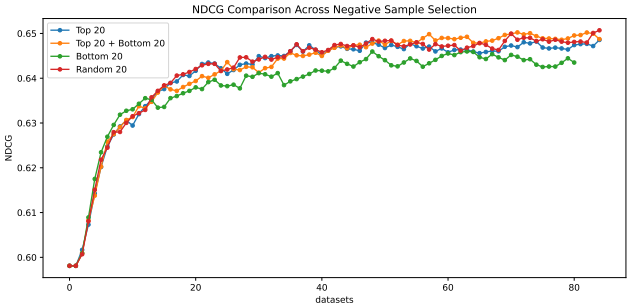

Figure 3: NDCG Comparison Across Different Negative Sample Selection Strategies

Figure 3 reveals several key performance patterns:

- **Initial Phase (0-20 datasets):**
  - All strategies show rapid NDCG improvement from 0.60 to 0.63-0.64
  - Similar learning trajectories suggest effective base architectures
  - Quick adaptation to respective sampling strategies
- **Stabilization Phase (20-80 datasets):**
  - Top 20 + Bottom 20 achieves and maintains highest NDCG (0.65)
  - Bottom 20 stabilizes at lower performance (0.64)
  - Random and Top 20 show similar intermediate performance

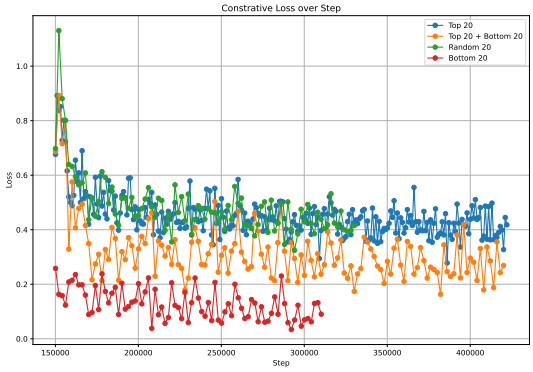

Figure 4: Contrastive Loss Over Training Steps

The contrastive loss behavior (Figure 4, starting at 150,000 steps after warm-up) shows distinct characteristics:

- **Loss Patterns:**
  - Bottom 20: Maintains lowest loss (0.2), indicating simpler learning tasks

  - Top 20: Higher loss (0.4-0.5) suggests challenging negative examples
  - Top 20 + Bottom 20: Balanced loss (0.3-0.4) with good stability
- **Training Stability:**
  - Initial adjustment period shows expected volatility
  - Stabilization occurs after approximately 200,000 steps
  - Different equilibrium points reflect strategy characteristics

5.6 Analysis

Our comprehensive analysis reveals three primary aspects:

1. Strategy Effectiveness:

- **Combined Strategy (Top 20 + Bottom 20):**
  - Achieves highest NDCG through balanced learning signals
  - Moderate loss indicates optimal challenge level
  - Better generalization from diverse negative samples
- **Individual Strategies:**
  - Bottom 20: Low loss but suboptimal NDCG suggests underfitting
  - Top 20: Higher loss indicates challenging but effective learning
  - Random: Competitive performance suggests natural balance

2. Training Dynamics:

- **Loss Characteristics:**
  - Top 20's higher loss reflects beneficial challenging examples
  - Bottom 20's low loss indicates insufficient learning pressure
  - Combined strategy shows optimal stability-challenge balance
- **Performance Patterns:**
  - Consistent improvement in early stages across strategies
  - Strategy differences emerge in later training phases
  - Long-term stability varies by negative sample difficulty

3. Practical Implications:

- Mixed strategy provides best balance of stability and performance
- Random sampling offers competitive alternative with implementation simplicity
- Strategy choice significantly impacts training efficiency and outcomes

4. Study Limitations:

- **Dataset Scope:**
  - Analysis conducted on 10% subset of MS MARCO
  - Metrics may not directly translate to full dataset performance
  - Relative rankings and patterns more indicative than absolute values
- **Performance Interpretation:**
  - NDCG and MRR values specific to subset dynamics
  - Loss patterns demonstrate relative strategy effectiveness
  - Full dataset behavior may show different convergence characteristics

These findings demonstrate that while hard negative samples are important for discriminative learning, combining them with easier negatives provides optimal training dynamics and model performance. The balanced approach prevents overfitting while maintaining sufficient learning challenge, leading to better generalization and stability.

6 OVERLAP STATEMENT

This project represents original research designed specifically for this course. It does not overlap with any other current work or coursework.

7 MISCELLANEOUS

- Using ANCE's original codebase as foundation
- Regular progress tracking and documentation
- Code modifications will be open-sourced
- Weekly development meetings
- Performance regression testing
- Continuous integration setup

REFERENCES

[1] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[2] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.

[3] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2004.04906* (2020).

[4] Xiao Li, Ye Wang, Mingxuan Ji, Yue Zhang, and Fei Wu. 2021. Representation Contrastive Learning for Dense Text Retrieval. *arXiv preprint arXiv:2102.04723* (2021).

[5] Y Liu, M Ott, N Goyal, J Du, M Joshi, D Chen, O Levy, M Lewis, L Zettlemoyer, and V Stoyanov. [n. d.]. RoBERTa: A robustly optimized BERT pretraining approach, 2019, CoRR.

[6] Ji Lu, Qingyun Wang, and Andrew McCallum. 2020. Neural Passage Retrieval with Improved Negative Contrast. *arXiv preprint arXiv:2010.12523* (2020).

[7] Yi Lu, Jinfeng Zhang, Rodrigo Nogueira, Yue Wang, and Jimmy Lin. 2021. A Study of IR Loss Functions for Dense Text Retrieval. *arXiv preprint arXiv:2102.08815* (2021).

[8] Chen Qu, Liu Yang, Minghui Chen, W. Bruce Croft, and Mohit Iyyer. 2021. Understanding Training Dynamics in Dense Retrieval. *arXiv preprint arXiv:2105.13425* (2021).

[9] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2010.08191* (2020).

[10] Microsoft Research. 2020. Approximate Nearest Neighbor Negative Contrastive Estimation (ANCE). https://github.com/microsoft/ANCE. Accessed: YYYY-MM-DD.

[11] Lee Xiong, Chenyan Xiong, Ye Li, Ke-Fei Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[12] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. Learning to Retrieve: How to Train a Dense Retrieval Model Effectively and Efficiently. *arXiv preprint arXiv:2010.10469* (2020).

[13] Xiang Zhang, Shizhu Yang, Jiaxin Zhang, and Xiao Li. 2020. ADORE: Adaptive Online Retrieval for Dense Text Embeddings. *arXiv preprint arXiv:2012.12593* (2020).