Noisy Pair Corrector for Dense Retrieval

Hang Zhang†1, Yeyun Gong, Xingwei He†2, Dayiheng Liu†1, Daya Guo†3, Jiancheng Lv†1, Jian Guo†4

1 College of Computer Science, Sichuan University
2 The University of Hong Kong 3 Sun Yat-sen University 4 IDEA Research, China
hangzhang_scu@foxmail.com, hexingwei15@gmail.com

Abstract

Most dense retrieval models contain an implicit assumption: the training query-document pairs are exactly matched. Since it is expensive to annotate the corpus manually, training pairs in real-world applications are usually collected automatically, which inevitably introduces mismatched-pair noise. In this paper, we explore an interesting and challenging problem in dense retrieval, how to train an effective model with mismatched-pair noise. To solve this problem, we propose a novel approach called Noisy Pair Corrector (NPC), which consists of a detection module and a correction module. The detection module estimates noise pairs by calculating the perplexity between annotated positive and easy negative documents. The correction module utilizes an exponential moving average (EMA) model to provide a soft supervised signal, aiding in mitigating the effects of noise. We conduct experiments on text-retrieval benchmarks Natural Question and TriviaQA, code-search benchmarks StaQC and SO-DS. Experimental results show that NPC achieves excellent performance in handling both synthetic and realistic noise.

1 Introduction

With the advancements in pre-trained language models (Devlin et al., 2019; Liu et al., 2019), dense retrieval has developed rapidly in recent years. It is essential to many applications including search engine (Brickley et al., 2019), open-domain question answering (Karpukhin et al., 2020a; Zhang et al., 2021), and code intelligence (Guo et al., 2021). A typical dense retrieval model maps both queries and documents into a low-dimensional vector space and measures the relevance between them by the similarity between their respective representations (Shen et al., 2014). During training, the model utilizes query-document pairs as labelled training data (Xiong et al., 2021) and samples negative documents for each pair. Then the model learns to minimize the contrastive loss for obtaining a good representation ability (Zhang et al., 2022b; Qu et al., 2021).

Recent studies on dense retrieval have achieved promising results with hard negative mining (Xiong et al., 2021), pretraining (Gao and Callan, 2021a), distillation (Yang and Seo, 2020), and adversarial training (Zhang et al., 2022a). All methods contain an implicit assumption: each query is precisely aligned with the positive documents in the training set. In practical applications, this assumption becomes challenging to satisfy, particularly when the corpora is automatically collected from the internet. In such scenarios, it is inevitable that the training data will contain mismatched pairs, incorporating instances such as user mis-click noise in search engines or low-quality reply noise in Q&A communities. As shown in Fig. 1, the examples are from StaQC benchmark (Yao et al., 2018), which is automatically collected from StackOverflow. The document, i.e., code solution, can not answer the query but is incorrectly annotated as a positive doc-

Figure 1: Two examples from StaQC training set. In the bottom example, the given code is mismatched with the query, since it can not answer the query.

---

Findings of the Association for Computational Linguistics: EMNLP 2023, pages 11439–11451
December 6-10, 2023 ©2023 Association for Computational Linguistics
We reveal and extensively explore a long-neglected problem in dense retrieval, i.e., mismatched-pair noise, which is ubiquitous in the real world. (2) We propose a simple yet effective method for training dense retrievers with mismatched-pair noise. (3) Extensive experiments on four datasets and comprehensive analyses verify the effectiveness of our method against synthetic and realistic noise. Code is available at https://github.com/hangzhang-nlp/NPC.

2 Preliminary

Before describing our model in detail, we first introduce the basic elements of dense retrieval, including problem definition, model architecture, and model training.

Given a query \( q \), and a document collection \( \mathbb{D} \), dense retrieval aims to find document \( d^+ \) relevant to \( q \) from \( \mathbb{D} \). The training set consists of a collection of query-document pairs, donated as \( C = \{ (q_1, d_1^+), \ldots, (q_N, d_N^+) \} \), where \( N \) is the data size. Typical dense retrieval models adopt a dual encoder architecture to map queries and documents into a dense representation space. Then the relevance score \( f(q, d) \) of query \( q \) and document \( d \) can be calculated with their dense representations:

\[
f_{\theta}(q, d) = \text{sim}(E(q; \theta), E(d; \theta)),
\]

where \( E(\cdot; \theta) \) denotes the encoder module parameterized with \( \theta \), and \( \text{sim} \) is the similarity function, e.g., euclidean distance, cosine distance, inner-product. Existing methods generally leverage the approximate nearest neighbor technique (ANN) (Johnson et al., 2019) for efficient search.

For training dense retrievers, conventional approaches leverage contrastive learning techniques (Karpukhin et al., 2020a; Zhang et al., 2022b). Given a training pair \( (q_i, d_i^+) \in C \), these methods sample \( m \) negative documents \( \{d_{i,1}^-, \ldots, d_{i,m}^-\} \) from a large document collection \( \mathbb{D} \). The retriever’s objective is to minimize the contrastive loss, pushing the similarity of positive pairs higher than negative pairs. Previous work (Xiong et al., 2021) has verified the effectiveness of the negative sampling strategy. Two commonly employed strategies are “In-Batch Negative” and “Hard Negative” (Karpukhin et al., 2020a; Qu et al., 2021).

We propose a Noisy Pair Corrector (NPC) framework to solve the false-positive problem. NPC consists of noise detection and correction modules. At each epoch, the detection module estimates whether a query-document pair is mismatched by the perplexity between the annotated document and easy negative documents. Then the correction module provides a soft supervised signal for both estimated noisy data and clean data via an exponential moving average (EMA) model. Both modules are plug-and-play, which means NPC is a general training paradigm that can be easily applied to almost all retrieval models.

The contributions of this paper are as follows: (1)
We propose NPC framework to learn retrievers with mismatched-pair noise. As shown in Fig. 3, NPC consists of two parts: (a) the noise detection module as described in Sec. 3.1, and (b) the noise correction module as described in Sec. 3.2.

3 Method

3.1 Noise Detection

The noise detection module is meant to detect mismatched pairs in the training set. We hypothesize that: dense retrievers will first learn to distinguish correctly matched pairs from easy negatives, and then gradually overfit the mismatched pairs. Therefore, we determine whether a training pair is mismatched by the perplexity between the annotated document and easy negative documents.

Specifically, given a retriever \( \theta \) equipped with preliminary retrieval capabilities, and an uncertain pair \((q_i, d_i)\), we calculate the perplexity as follows:

\[
PPL(q_i, d_i, \theta) = -\log \frac{e^{r_f(q_i, d_i)}}{e^{r_f(q_i, d_i)} + \sum_{j=1}^m e^{r_f(q_i, d_{i,j})}},
\]

where \( \tau \) is a hyper-parameter, \( d_{i,j} \) is the negative document randomly sampled from the document collection \( \mathbb{D} \). Note that \( d_{i,j} \) is a randomly selected negative document, not a hard negative. We discuss this further in Appendix C. In practice, we adopt the “In-Batch Negative” strategy for efficiency.

After obtaining the perplexity of each pair, an automated method is necessary to differentiate between the noise and the clean data. We note that there is a bimodal effect between the distribution of clean samples and the distribution of noisy samples. An example can be seen in Figure 4(b). Motivated by this, we fit the perplexity distribution over all training pairs with a two-component Gaussian Mixture Model (GMM):

\[
p(PPL | \theta) = \sum_{k=1}^K \pi_k \phi(PPL | k),
\]

where \( \pi_k \) and \( \phi(PPL | k) \) are the mixture coefficient and the probability density of the \( k \)-th component, respectively. We optimize the GMM with the Expectation-Maximization algorithm (Dempster et al., 1977).

Based on the above hypothesis, we treat training pairs with higher PPL as noise and those with lower PPL as clean data. So the estimated clean flag can be calculated as follows:

\[
\hat{y}_i = \mathbb{I}(p(\kappa | PPL(q_i, d_i, \theta)) > \lambda),
\]

where \( \hat{y}_i \in \{1, 0\} \) denotes whether we estimate the pair \((q_i, d_i)\) to be correctly matched or not, \( \kappa \) is the GMM component with the lower mean, \( \lambda \) is the threshold. \( p(\kappa | PPL(q_i, d_i, \theta)) \) is the posterior probability over the component \( \kappa \), which can be intuitively understood as the correctly annotated con-
fidence. We set \( \lambda \) to 0.5 in all experiments. Note that before noise detection, the retriever should equip with preliminary retrieval capabilities. This can be achieved by initializing it with a strong unsupervised retriever or by pre-training it on the entire noise dataset.

### 3.2 Noise Correction

Next, we will introduce how to reduce the impact of noisy pairs after obtaining the estimated flag set \( \{ \hat{y}_i \}_{i=1}^N \). One quick fix is to discard the noise data directly, which is sub-optimal since it wastes the query data in noisy pairs. In this work, we adopt a self-ensemble teacher to provide rectified soft labels for noisy pairs. The teacher is an exponential moving average (EMA) of the retriever, and the teacher is trained with a weight-averaged consistency target on noisy data.

Specifically, given a retriever \( \theta \), the teacher \( \theta^* \) is updated with an exponential moving average strategy as follows:

\[
\theta_i^* = \alpha \theta_{i-1}^* + (1 - \alpha) \theta_i,
\]

where \( \alpha \) is a momentum coefficient. Only the parameters \( \theta \) are updated by back-propagation.

For a query \( q_i \) and the candidate document set \( D_{q_i} \), where \( D_{q_i} = \{ d_{i,j} \}_{j=1}^m \) could consist of annotated documents, hard negatives and in-batch negatives, we first get teacher’s and retriever’s similarity scores, respectively. Then, the retriever \( \theta \) is expected to keep consistent with its smooth teacher \( \theta^* \). To achieve this goal, we update the retriever \( \theta \) by minimizing the KL divergence between the student’s distribution and the teacher’s distribution.

To be concrete, the similarity scores between \( q_i \) and \( D_{q_i} \) are normalized into the following distributions:

\[
p_{\phi}(d_{i,j}|q_i; D_{q_i}) = \frac{e^{\tau f_{\phi}(q_i,d_{i,j})}}{\sum_{j=1}^m e^{\tau f_{\phi}(q_i,d_{i,j})}}, \phi \in \{ \theta, \theta^* \}, \tag{6}
\]

Then, the consistency loss \( L_{cons} \) can be written as:

\[
L_{cons} = KL(p_{\theta}(\cdot|q_i; D_{q_i}), p_{\theta^*}(\cdot|q_i; D_{q_i})), \tag{7}
\]

where \( KL(\cdot) \) is the KL divergence, \( p_{\theta}(\cdot|q_i; D_{q_i}) \) and \( p_{\theta^*}(\cdot|q_i; D_{q_i}) \) denote the conditional probabilities of candidate documents \( D_{q_i} \) by the retriever \( \theta \) and the teacher \( \theta^* \), respectively.

For the estimated noisy pair, the teacher corrects the supervised signal into a soft label. For the estimated clean pair, we calculate the contrastive loss and consistency loss. So the overall loss is formalized:

\[
L = \hat{y}_i L_{cont} + L_{cons}, \tag{8}
\]

where \( \hat{y}_i \in \{1, 0\} \) is estimated by the noise detection module.

### 3.3 Overall Procedure

NPC is a general training framework that can be easily applied to most retrieval methods. Under the classical training process of dense retrieval, We first warmup the retriever with the typical contrastive learning method to provide it with basic retrieval abilities, and then add the noise detection module before training each epoch and the noise correction module during training. The detail is presented in Algorithm 1.

---

**Algorithm 1 Noisy Pair Corrector (NPC)**

**Require:** Retriever \( \theta \); Noisy Training dataset \( C \).

1. Warm up the retriever \( \theta \).
2. Initial EMA model \( \theta^* \) with \( \theta \).
3. for \( i = 1 : num\_epoch \) do
   4. Calculate PPL of training pairs with random negatives using Eq.2;
   5. Fit PPL distribution with GMM;
   6. Get the estimated flag set \( \{ \hat{y}_i \} \) using Eq.4;
   7. for \( i = 1 : num\_batch \) do
      8. Sample negatives with “In-Batch Negative” or “Hard Negative” strategy;
      9. Calculate rectified soft labels with EMA model \( \theta^* \);
     10. Train \( \theta \) by optimizing Eq.8;
   11. Update EMA model \( \theta^* \) using Eq.5;
   12. end for
6. end for

---

4 Experiments

4.1 Datasets

To verify the effectiveness of NPC in robust dense retrieval, we conduct experiments on four commonly-used benchmarks, including Natural Questions (Kwiatkowski et al., 2019), Trivia QA (Joshi et al., 2017), StaQC (Yao et al., 2018) and SO-DS (Heyman and Van Cutsem, 2020).

StaQC is a large dataset that collects real query-code pairs from Stack Overflow*. The dataset has

*https://stackoverflow.com/
Table 1: Retrieval performance on StaQC and SO-DS, which are realistic-noisy datasets. The results of the first block are mainly borrowed from published papers (Heyman and Van Cutsem, 2020; Li et al., 2022). If the results are not provided, we mark them as “-”.

<table>
<thead>
<tr>
<th>Methods</th>
<th>StaQC</th>
<th>SO-DS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@3</td>
<td>R@10</td>
</tr>
<tr>
<td>BM25\textsubscript{desc} (Heyman and Van Cutsem, 2020)</td>
<td>8.0</td>
<td>13.3</td>
</tr>
<tr>
<td>NBOW (Heyman and Van Cutsem, 2020)</td>
<td>10.9</td>
<td>16.6</td>
</tr>
<tr>
<td>USE (Heyman and Van Cutsem, 2020)</td>
<td>12.8</td>
<td>20.3</td>
</tr>
<tr>
<td>CodeBERT (Feng et al., 2020)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GraphCodeBERT (Guo et al., 2021)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CodeRetriever (In-Batch Negative) (Li et al., 2022)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CodeRetriever (Hard Negative) (Li et al., 2022)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UniXcoder (In-Batch Negative) (Guo et al., 2022)</td>
<td>29.98</td>
<td>47.47</td>
</tr>
<tr>
<td>UniXcoder (Hard Negative) (Guo et al., 2022)</td>
<td>31.18</td>
<td>48.38</td>
</tr>
<tr>
<td>NPC (In-Batch Negative)</td>
<td>33.07</td>
<td>50.35</td>
</tr>
<tr>
<td>NPC (Hard Negative)</td>
<td>34.38</td>
<td>52.20</td>
</tr>
</tbody>
</table>

been widely used on code summarization (Peddamail et al., 2018) and code search (Heyman and Van Cutsem, 2020). SO-DS mines query-code pairs from the most upvoted Stack Overflow posts, mainly focuses on the data science domain. Following previous works (Heyman and Van Cutsem, 2020; Li et al., 2022), we resort to Recall of top-k (R@k) and Mean Reciprocal Rank (MRR) as the evaluation metric. StaQC and SO-DS are constructed automatically without human annotation. Therefore, there are numerous mismatched pairs in training data.

Natural Questions (NQ) collects real queries from the Google search engine. Each question is paired with an answer span and golden passages from the Wikipedia pages. Trivia QA (TQ) is a reading comprehension dataset authored by trivia enthusiasts. During the retrieval stage of both datasets, the objective is to identify positive passages from a large collection. Positive pairs in these datasets are assessed based on strict rule, i.e., whether passages contain answers or not (Karpukhin et al., 2020a). Consequently, we consider these datasets to be of high quality. Thus, we leverage them for simulation experiments to quantitatively analyze the impact of varying proportions of noise. Drawing inspiration from the setting in the noisy classification task (Han et al., 2018), we simulate the mismatched-pair noise by randomly pairing queries with unrelated documents.

4.2 Implementation Details

NPC is a general training paradigm that can be directly applied to almost all retrieval models. For StaQC and SO-DS, we adopt UniXcoder (Guo et al., 2022) as our backbone, which is the SoTA model for code representation. Following Guo et al. (2022), we adopt the cosine distance as a similarity function and set temperature $\lambda$ to 20. We update model parameters using the Adam optimizer and perform early stopping on the development set. The learning rate, batch size, warmup epoch, and training epoch are set to 2e-5, 256, 5, and 10, respectively. In the “Hard Negative” setting, we adopt the same strategy as Li et al. (2022).

For NQ and TQ, we adopt BERT (Devlin et al., 2019) as our initial model. Following Karpukhin et al. (2020a), we adopt inner-product as the similarity function and set temperature $\lambda$ to 1. The max sequence length is 16 for query and 128 for passage. The learning rate, batch size, warmup epoch, and training epoch are set to 2e-5, 512, 10, and 40, respectively. We adopt “BM25 Negative” and “Hard Negative” strategies as described in the DPR toolkit. For a fair comparison, we implement DPR (Karpukhin et al., 2020a) with the same hyperparameters. All experiments are run on 8 NVIDIA Tesla A100 GPUs. The implementation of NPC is based on Huggingface (Wolf et al., 2020).

4.3 Results

Results on StaQC and SO-DS: Table 1 shows the results on the realistic-noisy datasets StaQC and SO-DS. Both datasets contain a large number of real noise pairs. The first block shows the results of previous SoTA methods. BM25\textsubscript{desc} is a traditional
Table 2: Retrieval performance on Natural Questions and Trivia QA under the noise ratio of 20%, and 50%, respectively. The results of BM25∗ and DPR∗ are borrowed from Karpukhin et al. (2020a). If the results are not provided, we mark them as “-”.

<table>
<thead>
<tr>
<th>Noisy Methods</th>
<th>Natural Questions</th>
<th>Trivia QA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1</td>
<td>R@5</td>
</tr>
<tr>
<td>BM25∗</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DPR∗</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DPR (BM25 Negative)</td>
<td>27.07</td>
<td>47.79</td>
</tr>
<tr>
<td>coCondenser (BM25 Negative)</td>
<td>29.12</td>
<td>51.02</td>
</tr>
<tr>
<td>Co-teaching (BM25 Negative)</td>
<td>26.02</td>
<td>52.48</td>
</tr>
<tr>
<td>DPR-C (BM25 Negative)</td>
<td>43.69</td>
<td>66.62</td>
</tr>
<tr>
<td>NPC (BM25 Negative)</td>
<td>45.22</td>
<td>68.42</td>
</tr>
<tr>
<td>DPR (Hard Negative)</td>
<td>37.61</td>
<td>60.73</td>
</tr>
<tr>
<td>coCondenser (Hard Negative)</td>
<td>40.71</td>
<td>63.41</td>
</tr>
<tr>
<td>RocketQA (Hard Negative)</td>
<td>43.32</td>
<td>64.25</td>
</tr>
<tr>
<td>Co-teaching (Hard Negative)</td>
<td>31.78</td>
<td>56.32</td>
</tr>
<tr>
<td>DPR-C (Hard Negative)</td>
<td>51.85</td>
<td>73.06</td>
</tr>
<tr>
<td>NPC (Hard Negative)</td>
<td>51.85</td>
<td>73.06</td>
</tr>
</tbody>
</table>

From the results, we can see that our proposed NPC consistently performs better than the evaluated models across all metrics. Compared with the strong baseline UniXcoder which ignores the mismatched-pair problem, NPC achieves a significant improvement with both “in-batch negative” and “hard negative” sampling strategies. It indicates that the mismatched-pair noise greatly limits the performance of dense retrieval models, and NPC can mitigate this negative effect. We also show some noisy examples detected by NPC in Appendix A.

Results on NQ and TQ: Table 2 shows the results on the synthetic-noisy datasets NQ and TQ under the noise ratio of 20%, and 50%. We compare NPC with BM25 (Yang et al., 2017) and DPR (Karpukhin et al., 2020a). BM25 is an unsupervised sparse retriever based on the exact term matching of queries and code descriptions. NBOW is an unsupervised retriever that leverages pre-trained word embedding of queries and code descriptions for retrieval. USE is a simple dense retriever based on transformer. CodeBERT, GraphCodeBERT are pre-trained models for code understanding using large-scale code corpus. CodeRetriever is a pre-trained model dedicated to code retrieval, which is pre-trained with unimodal and bimodal contrastive learning on a large-scale corpus. UniXcoder is also a pretrained model that utilizes multi-modal data, including code, comment, and AST, for better code representation. The results are implemented by ourselves for a fair comparison with NPC. The bottom block shows the results of NPC using two negative sampling strategies.

Co-teaching (Han et al., 2018) uses the samples with small loss to iteratively train two networks, which is widely used in the noisy label classification task. We implement baselines using two negative sampling strategies. Besides, we evaluate DPR on clean datasets by discarding the synthetic-noisy pairs, denoted by DPR-C. DPR-C is a strong
Table 3: Ablation studies on StaQC dev set and NQ dev set under noise ratio of 50%.

<table>
<thead>
<tr>
<th>Methods</th>
<th>NQ</th>
<th>StaQC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>De</td>
<td>Co</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Baseline that is not affected by mismatched pairs.

We can observe that (1) As the noise ratio increases, DPR, coCondenser, and RocketQA experience a significant decrease in performance. At a noise rate of 50%, they perform worse than unsupervised BM25. (2) Despite Co-teaching having good noise resistance, its performance is still low. This indicates that methods for dealing with label noise in classification are not effective for retrieval. (3) NPC outperforms baselines by a large margin, with only a slight performance drop when the noise increases. Even comparing DPR-C, NPC still achieves competitive results.

4.4 Analysis

Ablations of Noise Detection and Noise Correction: To get a better insight into NPC, we conduct ablation studies on the realistic-noisy dataset StaQC and the synthetic-noisy dataset NQ under the noise ratio of 50%. The results are shown in Table 3. “De” and “Co” refer to noise detection and noise correction, respectively. “HN” indicates whether to perform “Hard Negative” strategy. For both synthetic noise and realistic noise, we can see that the noise detection module brings a significant gain, no matter which negative sampling strategy is used. Correction also enhances the robustness of the retriever since it provides rectified soft labels which can lead the model output to be smoother. The results show that combining the two obtains better performance compared with only using the detection module or correction module.

Impact of Warmup Epoch: According to the foregoing, NPC starts by warming up. In Table 4, we pre-training the retriever on the noisy dataset for warming up, and show the performance of NPC with different various epoch numbers $n$. In this experiment, we adopt “Hard Negative” sampling strategy. We find that NPC achieves good results when the warmup epoch is relatively small ($1 - 10$). However, when the warmup epoch is too large, the performance will degrade. We believe that a prolonged warmup causes overfitting to noise samples.

Impact of Iterative Detection: In the training of NPC, we perform iterative noise detection every epoch. A straightforward approach is to detect the noise only once after warmup and fix the estimated flag set $\{\hat{y}_i\}$. To study the effectiveness of iterative detection, we conducted an ablation study. The results are shown in Table 5. We can see that the model performance degrades after removing iterative detection.

Ablations of PPL: We distinguish noise pairs according to the perplexity between the annotated positive document and easy negatives. When calculating the perplexity, “Hard Negative” will cause trouble for detection. We construct ablation experiments to verify this, and the results are shown in Table 5. We can see that the perplexity with “Hard Negative” results in performance degradation.

Visualization of Perplexity Distribution: In Fig. 4, we illustrate the perplexity distribution of
Table 5: Ablation studies of iterative noise detection and perplexity variants

<table>
<thead>
<tr>
<th>Setting</th>
<th>R@1</th>
<th>R@5</th>
<th>R@20</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPC</td>
<td>50.07</td>
<td>69.93</td>
<td>80.07</td>
<td>85.89</td>
</tr>
<tr>
<td>-w/o iterative detection</td>
<td>47.29</td>
<td>68.39</td>
<td>78.79</td>
<td>85.38</td>
</tr>
<tr>
<td>-ppl with HN</td>
<td>42.81</td>
<td>65.06</td>
<td>75.22</td>
<td>83.09</td>
</tr>
</tbody>
</table>

Figure 5: Retrieval performance of DPR and NPC on NQ dev set under different noise ratios.

5 Related Work

5.1 Dense Retrieval

Dense retrieval has shown better performance than traditional sparse retrieval methods (Lee et al., 2019; Karpukhin et al., 2020a). The studies of dense retrieval can be divided into two categories, (1) unsupervised pre-training to get better initialization and (2) more effective fine-tuning on labeled data. In the first category, some researchers focus on how to generate contrastive pairs automatically from a large unsupervised corpus (Lee et al., 2019; Chang et al., 2019; Ma et al., 2022; Li et al., 2022). Another line of research enforces the model to produce an information-rich CLS representation (Gao and Callan, 2021a,b; Lu et al., 2021). As for effective fine-tuning strategies (He et al., 2022b), recent studies show that negative sampling techniques are critical to the performance of dense retrievers. DPR (Karpukhin et al., 2020b) adopts in-batch negatives and BM25 negatives; ANCE (Xiong et al., 2021), RocketQA (Qu et al., 2021), and AR2 (Zhang et al., 2022a) improve the hard negative sampling by iterative replacement, denoising, and adversarial framework, respectively. Several works distill knowledge from ranker to retriever (Izacard and Grave, 2020; Yang and Seo, 2020; Ren et al., 2021; Zeng et al., 2022). Some studies incorporate lexical-aware sparse retrievers to convey lexical-related knowledge to dense retrievers, thereby enhancing the dense retriever’s ability to recognize lexical matches (Shen et al., 2023; Zhang et al., 2023).

Although the above methods have achieved promising results, they are highly dependent on correctly matched data, which is difficult to satisfy in real scenes. The mismatched-pair noise problem has seldom been considered. Besides, some studies utilize large-sized generative models (He et al., 2023) to guide retrievers, which achieve impressive performance without paired data (Sachan et al., 2022, 2021; Gao et al., 2022; He et al., 2022a). Although these models exhibit some robustness to noisy data, their success depends on the availability of strong generative models. Moreover, their applicability will be limited in domains where generative models do not perform well.

5.2 Denoising Techniques

One related task to our work is Noisy Label. Numerous methods have been proposed to solve this problem, and most of them focus on the classification task (Han et al., 2020). Some works design robust loss functions to mitigate label noise (Ghosh et al., 2017; Ma et al., 2020). Another line of work aims to identify noise from the training set with the memorization effect of neural networks (Silva et al., 2022; Liang et al., 2022; Bai et al., 2021). These studies mainly focus on classification. NPC studies the mismatched noise problem in dense retrieval rather than the noise in category annotations, which is more complex to handle. Several pre-training approaches noticed the problem of mismatched noisy pairs. ALIGN (Jia et al., 2021) and CLIP (Radford et al., 2021) claim that utilizing
large-scale image-text pairs can ignore the existence of noise. E5 (Wang et al., 2022) employs a consistency-based rule to filter the pre-training data. Although they slightly realized the existence of noisy pairs during pre-train, none of them give a specialized solution to solve it and extensively explored the characteristics of noisy text pairs. Some recent works (Huang et al., 2021; Han et al., 2023) study the noisy correspondence problem in cross-modal retrieval. Although the "mismatched-pair noisy" problem in cross-modal retrieval and text retrieval shares similarities, the specific settings and methods used in these two areas are notably distinct. It is challenging to directly apply these cross-modal retrieval works to document and code retrieval. Our NPC is the first systematic work to explore mismatched-pair noise in document/code retrieval.

6 Conclusion

This paper explores a neglected problem in dense retrieval, i.e., mismatched-pair noise. To solve this problem, we propose a generalized Noisy Pair Corrector (NPC) framework, which iteratively detects noisy pairs per epoch based on the perplexity and then provides rectified soft labels via an EMA model. The experimental results and analysis demonstrate the effectiveness of NPC in effectively handling both synthetic and realistic mismatched-pair noise.

Limitations

This work mainly focuses on training the dense retrieval models with mismatched noise. There may be two possible limitations in our study.

1) Due to the limited computing infrastructure, we only verified the robustness performance of NPC based on the classical retriever training framework. We leave experiments to combine NPC with more effective retriever training methods such as distillation (Ren et al., 2021), AR2 (Zhang et al., 2022a), as future work.

2) Mismatched-pair noise may also exist in other tasks, such as recommender systems. We will consider extending NPC to more tasks.

Acknowledgement

This work is supported by the Fundamental Research Funds for the Central Universities under Grant 1082204112364 and the Key Program of the National Science Foundation of China under Grant 61836006.

References

Yingbin Bai, Erkun Yang, Bo Han, Yanhua Yang, Jiayong Li, Yinian Mao, Gang Niu, and Tongliang Liu. 2021. Understanding and improving early stopping for learning with noisy labels. In NIPS.


Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. CoRR.


Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. 2022. Unixcoder: Unified cross-modal pre-training for code representation. In ACL.


Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In NIPS.


Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In International Conference on Machine Learning.


Xingjun Ma, Hanxun Huang, Yisen Wang, Simone Romano, Sarah Erfani, and James Bailey. 2020. Normalized loss functions for deep learning with noisy labels. In ICML.


A Qualitative Analysis

Table 9 lists some mismatched pairs detected by NPC in StaQC training set. We can see that these mismatched pairs are almost irrelevant and can be correctly detected by NPC. These examples are not well aligned, mainly due to the low-quality answers of the open community (cases 2 and 4), inappropriate data preprocessing in the collection phase (cases 2 and 3), and other reasons. It is well known that collecting and cleaning training data is expensive and complex work. Automatically constructed datasets in real-world applications often contain such mismatched-pair noise. Our method can mitigate the impact caused by such noise during training.

B Statistics of Datasets

The statistics of datasets are shown in Table 6.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
<th>Corpus size</th>
</tr>
</thead>
<tbody>
<tr>
<td>StaQC</td>
<td>203.7K</td>
<td>2.6K</td>
<td>2.7K</td>
<td>14.6K</td>
</tr>
<tr>
<td>SO-DS</td>
<td>12.1K</td>
<td>0.9K</td>
<td>1.1K</td>
<td>12.1K</td>
</tr>
<tr>
<td>NQ</td>
<td>79.2K</td>
<td>8.8K</td>
<td>3.6K</td>
<td>21 M</td>
</tr>
<tr>
<td>TQ</td>
<td>78.8K</td>
<td>8.8k</td>
<td>11.3K</td>
<td>21 M</td>
</tr>
</tbody>
</table>

Table 6: The statistics of datasets. Corpus size means the size of document corpus for evaluation.

C Discussion about Perplexity

We calculate the perplexity between the annotated document and easy negative documents during noise detection. We emphasize that the negative documents are randomly selected from the document collection D. It is not suitable to adopt “Hard Negative” sampling strategy when calculating the perplexity. Although hard negatives are important to train a strong dense retriever, they will cause trouble during noise detection. Specifically, it is expected that the retriever is confused only between false positive and negative documents and can confidently distinguish true positive and negative documents. But if we adopt “Hard Negative” when calculating the perplexity, the retriever will also be confused between true positive and hard negative documents, which will affect noise detection. We construct ablation experiments to verify this, and the results are shown in Table 5.

D Integration with stronger methods

We conducted comprehensive experiments that integrated NPC into coCondenser and RocketQA v2. The subsequent experiments were conducted on the NQ dataset with a 50% noise ratio. We first combined NPC with coCondenser which is a pre-trained model specialized for dense retrieval tasks. The results are shown in Table 7:

<table>
<thead>
<tr>
<th></th>
<th>R@1</th>
<th>R@5</th>
<th>R@20</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25 Negative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>coCondenser</td>
<td>18.28</td>
<td>36.37</td>
<td>52.01</td>
<td>65.92</td>
</tr>
<tr>
<td>coCondenser-C</td>
<td>44.75</td>
<td>68.91</td>
<td>80.89</td>
<td>87.31</td>
</tr>
<tr>
<td>coCondenser+NPC</td>
<td>47.31</td>
<td>70.38</td>
<td>81.58</td>
<td>87.38</td>
</tr>
<tr>
<td>Hard Negative</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>coCondenser</td>
<td>24.55</td>
<td>44.16</td>
<td>56.69</td>
<td>68.72</td>
</tr>
<tr>
<td>coCondenser-C</td>
<td>49.31</td>
<td>71.99</td>
<td>82.41</td>
<td>88.38</td>
</tr>
<tr>
<td>coCondenser+NPC</td>
<td>50.66</td>
<td>72.42</td>
<td>82.64</td>
<td>88.31</td>
</tr>
</tbody>
</table>

Table 7: Retrieval performance on NQ after combining NPC with coCondenser.

It’s evident that NPC significantly enhances the robustness of coCondenser against noise associated with mismatched pairs. This observation underscores the compatibility between NPC and pre-trained dense retrievers.

Furthermore, we combined NPC with RocketQA v2 which adopted a cross-encoder as a teacher and dynamically distilled knowledge to the dense retriever. The results are shown in Table 8:

<table>
<thead>
<tr>
<th></th>
<th>R@1</th>
<th>R@5</th>
<th>R@20</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>RocketQA v2</td>
<td>32.30</td>
<td>51.37</td>
<td>62.19</td>
<td>70.79</td>
</tr>
<tr>
<td>RocketQA v2-C</td>
<td>52.63</td>
<td>73.51</td>
<td>83.21</td>
<td>88.71</td>
</tr>
<tr>
<td>RocketQA v2+NPC</td>
<td>52.59</td>
<td>73.83</td>
<td>83.32</td>
<td>88.69</td>
</tr>
</tbody>
</table>

Table 8: Retrieval performance on NQ after combining NPC with RocketQA v2.

To combine NPC with RocketQA v2, we integrate the noise detection and the correction modules in each training epoch of RocketQA v2. From the table, we can find that although RocketQA v2 uses a powerful cross-encoder as a teacher, it is still limited by the noise of the training data and shows low performance. NPC can effectively harmonize with RocketQA v2 to mitigate the problems caused by mismatched pair noise. We will add these experiment results to the next version.
<table>
<thead>
<tr>
<th>Question</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Split words in a nested list into letters</td>
<td>» [list(l[0]) for l in mylist]</td>
</tr>
<tr>
<td>2 Dictionary in python problem</td>
<td>» s = problem.getSuccessors( getStartState())</td>
</tr>
<tr>
<td>3 Find the Common first name from Django Auth user Model</td>
<td>» import operator</td>
</tr>
<tr>
<td>4 Find all text files not containing some text string</td>
<td>» lst = [1,2,4,6,3,8,0,5]</td>
</tr>
<tr>
<td></td>
<td>» for n in lst[:]:</td>
</tr>
<tr>
<td></td>
<td>»» if n % 2 == 0:</td>
</tr>
<tr>
<td></td>
<td>»»» lst.remove(n)</td>
</tr>
<tr>
<td></td>
<td>» lst</td>
</tr>
</tbody>
</table>

Table 9: Some noisy pairs detected by NPC in StaQC training set.