DMRetriever: A Family of Models for Improved Text Retrieval in Disaster Management

Kai Yin[1] **Xiangjue Dong**[1] **Chengkai Liu**[1] **Allen Lin**[1]
Lingfeng Shi[1] **Ali Mostafavi**[1] **James Caverlee**[1]
[1]Texas A&M University
{kai_yin,xj.dong,liuchengkai,al001,lingfengs111,mostafavi,caverlee}@tamu.edu

Abstract

Effective and efficient access to relevant information is essential for disaster management. However, no retrieval model is specialized for disaster management, and existing general-domain models fail to handle the varied search intents inherent to disaster management scenarios, resulting in inconsistent and unreliable performance. To this end, we introduce DMRetriever, the first series of dense retrieval models (33M to 7.6B) tailored for this domain. It is trained through a novel three-stage framework of bidirectional attention adaptation, unsupervised contrastive pre-training, and difficulty-aware progressive instruction fine-tuning, using high-quality data generated through an advanced data refinement pipeline. Comprehensive experiments demonstrate that DMRetriever achieves state-of-the-art (SOTA) performance across all six search intents at every model scale. Moreover, DMRetriever is highly parameter-efficient, with 596M model outperforming baselines over 13.3× larger and 33M model exceeding baselines with only 7.6% of their parameters. All codes, data, and checkpoints are available at this repository.

1 Introduction

Disasters, both natural and man-made, inevitably disrupt communities, endanger lives, and cause severe economic losses (Fan et al., 2021; Liu et al., 2025). During such events, emergency responders, policymakers, and affected populations rely on information retrieval (IR) systems for decision-making, situational awareness, and coordination of relief efforts (Abbas and Miller, 2025). Timely and reliable information access is thus essential for effective disaster management (Priya et al., 2020).

Information needs during disaster events are highly diverse, spanning fact-checking critical information, monitoring social media, and consulting technical or policy documents (See §3.1 for full lists of search intents) (Yin et al., 2025). However, a recent analysis of 30 popular general-domain IR models finds that *none consistently achieve SOTA performance across all disaster management-specific search intents* (Yin et al., 2025). This limits their real-world applicability: some queries yield valuable insights, while others return irrelevant results, delaying decision-making and risking missed evidence that could compromise response efforts.

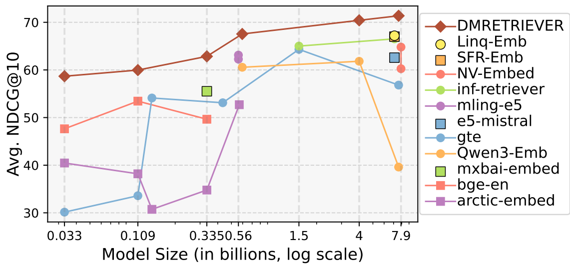

Figure 1: Average performance of DMRetriever. Breakdown for each search intent is in Table 2

Furthermore, real-world disaster management demands *scalable retrieval solutions*, from lightweight models for resource-constrained settings (e.g., on edge devices for rapid response) to larger models for performance-critical decision-making (e.g., in central command system). Hence, there is a great opportunity to **develop domain-specialized IR models of various scales that achieve both high performance and computational efficiency for disaster management.**

To this end, we introduce DMRetriever, a family of dense retrieval models at six scales (33M–7.6B parameters) to improve retrieval performance in disaster management (Figure 1).

The success of domain-specific IR models in fields such as biomedicine (Xu et al., 2024b), law (Chalkidis et al., 2020), and finance (Sarmah et al., 2024) shows the effectiveness of adding deep domain knowledge to improve performance. However, injecting disaster management knowledge into DMRetriever to improve performance

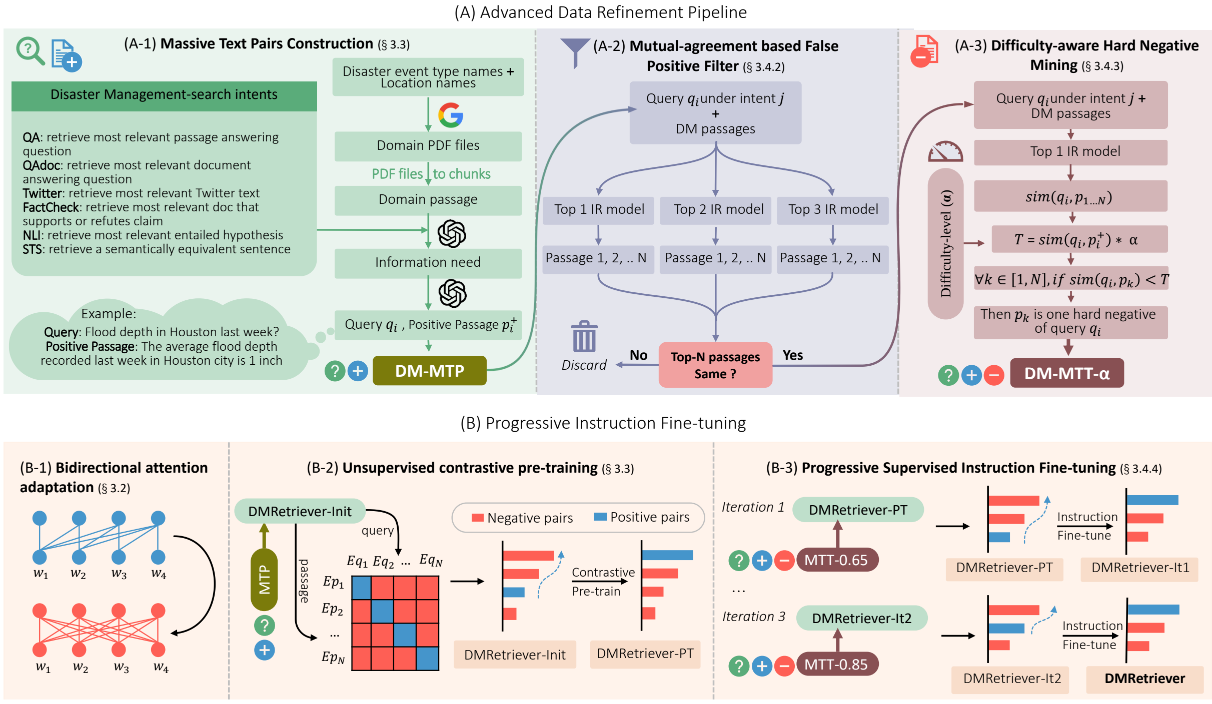

Figure 2: Overview of data refinement pipeline and progressive fine-tuning for DMRETRIEVER. MTP: **M**assive **T**ext **P**airs, MTT: **M**assive **T**ext **T**riplets. Both MTP and MTT in sub-figure (B) include DM (**D**isaster **M**anagement) and GD (**G**eneral **D**omain) variants, with constructions of GD variants omitted here and detailed in §3.3.1 and §3.4.3. DM passages include those extracted from PDFs and those generated by LLMs. DMRetriever-Init: backbone after bidirectional attention adaptation for decoder-only; original backbone for encoder-only. DMRetriever-PT: model after pre-training. DMRetriever-It2: model after iteration 2. Algorithm 1 details the full workflow.

poses **two challenges**: (1) the lack of high-quality, knowledge-rich data that covers diverse search intents during disasters, and (2) the absence of an efficient training method that can transfer such knowledge across models of different sizes.

To tackle the first challenge, we propose a **data refinement pipeline** (A in Figure 2) to generate *the first*, knowledge-rich data for disaster management. We first prompt a large language model (LLM)[1] with domain-specific passages to generate large-scale unlabeled query–passage pairs[2] (A-1 in Figure 2, §3.3.1). Given limitations of existing IR models in disaster management, we refine these noisy corpora with mutual-agreement filtering, which removes mislabeled pairs through model consensus (A-2 in Figure 2, §3.4.2). Recognizing the varied learning capacities of different-sized DMRETRIEVER models, we further propose difficulty-aware hard negative mining to generate negatives at varying difficulty levels (A-3 in Figure 2, §3.4.3), finally yielding a high-quality labeled dataset.

[1]We use GPT-4o-mini as LLM in this work.

[2]Following Xiao et al. (2024); Wang et al. (2022), we treat data without mined negatives as unlabeled and data with mined hard negatives as labeled.

For the second challenge, we propose a **Progressive instruction fine-tuning** (B in Figure 2) that enables DMRETRIEVER models of different sizes to effectively learn domain knowledge. For large variants initialized from decoder-only backbones (Table 1), we adapt their causal attention into bidirectional attention, allowing access to information from future tokens (B-1 in Figure 2, §3.2). Next, to better initialize DMRETRIEVER and inject domain knowledge, we perform large-scale contrastive pre-training on unlabeled query–passage pairs (B-2 in Figure 2, §3.3). Finally, to adapt DMRETRIEVER to diverse search intents, we perform instruction fine-tuning on labeled data[2], where for small variants that have limited learning capacity, the difficulty of negative samples is progressively increased during fine-tuning (B-3 in Figure 2, §3.4.4).

Extensive experiments show that DMRETRIEVER achieves new SOTA results on all six search intents at each scale (Figure 1). It is also highly parameter-efficient: 596M variant beats all XL-scale baselines while being over 13× smaller, and 33M variant surpasses all medium baselines with only 7.6% of their parameters. Ablation study further confirms that knowledge generated by data refinement pipeline and injected through three-stage training framework substantially enhances model performance in disaster management.

The contributions of this work are as follows:

(1) We introduce DMRetriever, a family of six dense retrieval models (33M to 7.6B) that achieve new SOTA performance on each search intent across all scales, with superior parameter efficiency over existing baselines.

(2) We propose a data refinement pipeline yielding the first large-scale training dataset for disaster management. It is generalizable to other domains with limited annotations.

(3) We introduce progressive instruction fine-tuning which enables models of varied sizes to effectively absorb domain knowledge and enhances their ability across different search intents.

2 Related Work

General-domain retrieval has been long dominated by bidirectional encoders such as BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020), with models like Sentence-BERT (Reimers and Gurevych, 2019) and SimCSE (Gao et al., 2021) improving performance through contrastive pre-training and supervised fine-tuning on MS MARCO (Nguyen et al., 2016). More recently, decoder-only models, from GPT-based embeddings (Brown et al., 2020) to models such as Qwen3-Embedding (Zhang et al., 2025), have leveraged instruction tuning and large-scale synthetic data to achieve SOTA results.

However, these general-domain models face clear limitations in disaster management, with no approach achieving SOTA performance across diverse search intents (Yin et al., 2025) due to distribution shift issue (Thakur et al., 2021). Prior work on disaster information access has mainly targeted on social media, emphasizing clustering of posts, event or situational classification, and named entity recognition (Imran et al., 2015; Alam et al., 2021; Yin et al., 2024). While valuable, these efforts remain narrow in scope and have yet to address the need for robust retrieval systems tailored to disaster management-related diver search intents.

3 DMRetriever

DMRetriever is a family of dense retrieval models at six scales (33M–7.6B) initialized from encoder- and decoder-only backbones (Table 1). This scalability supports real-world disaster management applications ranging from resource-constrained settings with smaller models to performance-critical scenarios with larger ones.

3.1 Preliminaries

In DMRetriever, queries and passages are encoded into dense vectors using a shared embedding model $\mathbf{E}(\cdot)$. Given a query q and a passage p, we prepend a search intent-specific instruction I_q to the query to improve search intent awareness (Wei et al., 2021; Ouyang et al., 2022; Lee et al., 2024a).

Following DisastIR (Yin et al., 2025), we consider six disaster management-specific search intents: question-answer (QA), Twitter (TW), Fact Checking (FC), Natural Language Inference (NLI), and Semantic Textual Similarity (STS). Table 5 details corresponding instruction formats.

The embeddings are then computed as $\mathbf{e}_q = \mathbf{E}(I_q \oplus q)$ and $\mathbf{e}_p = \mathbf{E}(p)$, where $\oplus$ denotes sequence concatenation. We apply mean pooling over the final hidden states to obtain fixed-length representations. The relevance score $\mathrm{sim}(q, p)$ is computed as the cosine similarity between embeddings, scaled by a temperature parameter τ:

$$\mathrm{sim}(q, p) = \cos(\mathbf{e}_q, \mathbf{e}_p)/\tau. \tag{1}$$

To effectively adapt DMRetriever to disaster management domain, we propose a three-stage training framework using high-quality training data produced from our advanced data refinement pipeline (Figure 2 and Algorithm 1).

3.2 Enabling Bidirectional Attention in Decoder-only Backbones

3.2.1 Bidirectional attention adaptation

For DMRetriever initialized from decoder-only backbones, we replace the causal attention mask with an all-ones matrix (B-1 in Figure 2) to enable the model to capture information from future tokens, inspired by the recent success of Lee et al. (2024a); BehnamGhader et al. (2024); Muennighoff et al. (2024). This replacement allows each token to attend to every other token in the sequence, converting it into a bidirectional model.

To make model aware of its bidirectional attention, we train model by using masked language model (MLM) training objective (Devlin et al., 2019). Given an input sequence, we randomly mask some of tokens and then train model to predict masked tokens based on past and future tokens.

Model	#Parameters	Backbone	Backbone Type	Hidden Size	#Layers
DMRetriever-33M	33M	MiniLM	encoder-only	384	12
DMRetriever-109M	109M	BERT-base-uncased	encoder-only	768	12
DMRetriever-335M	335M	BERT-large-uncased-WWM	encoder-only	1024	24
DMRetriever-596M	596M	Qwen3-0.6B	decoder-only	1024	28
DMRetriever-4B	4B	Qwen3-4B	decoder-only	2560	36
DMRetriever-7.6B	7.6B	Qwen3-8B	decoder-only	4096	36

Table 1: Model configurations of DMRetriever family across six model sizes. DMRetriever-7.6B is initialized from Qwen3-8B after removing its language modeling head, leaving 7.568B parameters (rounded to 7.6B).

3.2.2 MLM training data construction

We perform MLM training using a mix of general-domain and disaster management corpora. For the general domain, we use the English Wikipedia from Merity et al. (2016). For the disaster management domain, we use passages from DM-MTP (constructed in §3.3.1).

3.3 Unsupervised contrastive pre-training

3.3.1 Pre-training dataset construction

To robustly initialize DMRetriever for retrieval tasks and inject domain knowledge, we construct MTP (**M**assive **T**ext **P**airs; see Algorithm 2), which consists of two large-scale subsets: DM-MTP (disaster management domain) and GD-MTP (general-domain). Its composition is shown in Table 10.

In the absence of paired datasets for disaster management, DM-MTP represents the first large-scale collection of query–passage pairs in this domain. Inspired by Wang et al. (2023) and Yin et al. (2025), we propose a three-stage pipeline to generate it (A-1 in Figure 2, lines 1–7 in Algorithm 2). In the first stage, we crawl the web with queries formed by combining disaster event type names and location names, collect domain-specific PDFs, and process them into chunks, taking these as disaster-domain passages. Secondly, for each passage, a LLM generates information-need statement taking passage content and its designated search intent definition as input. Finally, the information-need statement is paired with the passage to prompt the LLM to generate a user query and a directly relevant passage, which serves as the positive passage for this query in DM-MTP. Appendix A details the whole DM-MTP construction process.

To enrich DMRetriever with diverse linguistic and semantic relevance patterns, GD-MTP is built by aggregating 15 publicly available datasets (Table 10). Query–positive passage pairs are created from unlabeled data using search intent-specific heuristics (Lines 8–12 in Algorithm 2). For example, in datasets associated with QA intents, user questions are treated as queries and their answers as positive passages. Heuristics for other search intents are described in Appendix B.

For negative sampling in MTP, instead of explicitly mining hard negatives, we adopt the in-batch negative strategy (Karpukhin et al., 2020), where passages from other pairs within the same mini-batch serve as negative examples.

3.3.2 Contrastive Pre-training

DMRetriever is pre-trained on MTP using contrastive learning to separate relevant query–passage pairs from irrelevant ones. For each mini-batch $\mathcal{B}$, we adopt InfoNCE loss (Chen et al., 2020) as pre-training objective, encouraging similarity between q_i and its positive passage p_i^+ to be higher than that with any other passage $p_j \in \mathcal{B}$, $j \neq i$ within the same batch (B-2 in Figure 2):

$$\mathcal{L}_{\text{cpt}} = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{\sum_{p_j \in \mathcal{B}} e^{\text{sim}(q_i, p_j)}}. \tag{2}$$

3.4 Difficulty-aware Progressive Supervised Instruction Fine-tuning

3.4.1 Fine-tuning dataset overview

To further enhance domain-specific retrieval capabilities, DMRetriever is fine-tuned on a labeled dataset, MTT (**M**assive **T**ext **T**riplets; Algorithm 3) comprising: DM-MTT for disaster management domain and GD-MTT for general domain. DM-MTT is obtained by applying false positive filtering (§3.4.2) and hard negative mining (§3.4.3) to query–passage pairs in DM-MTP, while pairs in GD-MTP undergo hard negative mining to produce GD-MTT. Its composition is listed in Table 11.

3.4.2 Mutual-agreement False Positive Filter

Previous studies (Alberti et al., 2019; Xu et al., 2024b; Wang et al., 2022) show that LLM synthetic data can be noisy, and they commonly adopt a consistency-based filtering (CBF for short) approach: discarding pairs in which LLM-generated positive passage is not ranked within the top-k results returned by a reference embedding model.

Given the limited capability of existing open-source IR models in disaster management, we propose a *mutual-agreement-based false positive filtering* method to more effectively remove low-quality pairs from DM-MTP (A-2 in Figure 2, lines 1–8 in Algorithm 3). For each query, we use the top three open-source IR models identified in DisastIR for its corresponding search intent (Table 4) to retrieve the top-N passages. A query is retained only if all three models return the same top-N passages; in that case, its top-1 passage is designated as the positive passage of this query.

3.4.3 Difficulty-aware Hard Negative Mining

Query–positive passage pairs further undergo hard negative mining to provide challenging negatives for enhancing DMRetriever's performance. Given that DMRetriever spans a wide range of model parameter sizes with different learning capacities, we propose a *difficulty-aware hard negative mining strategy* that caters to each model's learning ability by providing negatives at appropriate difficulty levels.

Inspired by Moreira et al. (2024), for a query q_i with a positive passage p_i^+, a passage p_k is treated as a hard negative $p_{i,k}^-$ if

$$\mathrm{sim}(q_i, p_k) < \mathrm{sim}(q_i, p_i^+) \times \alpha. \tag{3}$$

For each query q_i, we employ the best-performing IR model corresponding to its search intent (See Table 4) to retrieve the top 200 passages from the associated passage corpus. From this pool, we select the top-K passages that satisfy Equation 3, forming the hard negative set $\mathcal{H}_i$ (A-3 in Figure 2, Lines 10–15 in Algorithm 3).

The margin α serves as a tunable parameter that controls the difficulty of mined negatives, referred to as the *difficulty level*. Instead of applying a uniform difficulty level across all DMRetriever variants, we adjust α based on model size to better align with their learning capacities. This yields datasets tailored to specific difficulty levels, denoted as MTT-α.

3.4.4 Progressive Supervised Instruction Fine-tuning

Previous studies show that retrieval performance degrades when trained with negatives that are either too difficult or too simple (Merrick et al., 2024). To address this, for small-sized variants of DMRetriever (33M, 109M, and 335M) that have limited learning capacity after pre-training, we introduce *progressive instruction fine-tuning* (B-3 in Figure 2, Lines 9–13 in Algorithm 1), inspired by curriculum learning (Bengio et al., 2009). This approach gradually increases the difficulty of training samples across iterations,[3] where the best validation checkpoint from one iteration initializes the next. At each stage, the difficulty level α is raised to make negatives harder, with models progressively trained on MTT-0.65, 0.75, and 0.85.

In contrast, the larger variants (596M, 4B, and 7.6B), which already exhibit strong performance after pre-training, are fine-tuned only on MTT-0.95. α value selections are detailed in Appendix D.

To further adapt DMRetriever's embeddings to different search intents, we perform instruction fine-tuning, where each query q_i is prepended with an intent-specific instruction I_q, forming the input $\tilde{q}_i = [I_q; q_i]$. The model is then fine-tuned on MTT-α at each iteration using the InfoNCE loss:

$$\mathcal{L}_{\text{ft}} = -\log \frac{e^{\mathrm{sim}(\tilde{q}_i, p_i^+)}}{e^{\mathrm{sim}(\tilde{q}_i, p_i^+)} + \sum_{p_{i,k}^- \in \mathcal{H}_i} e^{\mathrm{sim}(\tilde{q}_i, p_{i,k}^-)}}. \tag{4}$$

4 Experiment Setup

4.1 DisastIR-DevLite and DisastIR-Test Construction

Prior works (Boteva et al., 2016; Yang et al., 2018; Thakur et al., 2021) typically construct validation sets by randomly splitting queries while keeping the passage corpus identical across validation and test sets, which makes repeatedly encoding the full passage corpus computationally expensive.

To address this, we introduce **DisastIR-DevLite** (See Algorithm 4), a lightweight validation set derived from DisastIR[4] by sampling queries and constructing a much smaller passage corpus tailored to them, while the remaining queries form **DisastIR-Test** with the full DisastIR passage corpus. Full construction details are provided in Appendix E.

DisastIR-DevLite contains only 3.9 % of the original passages (Table 6), yet remains challenging and informative for model development. Appendix F highlights its practicality, showing that

[3]An iteration refers to a curriculum stage in Lines 9–12 of Algorithm 1, each consisting of multiple training batches.

[4]DisastIR is the only publicly released IR benchmark for disaster management available at this repository.

Model	Scale	QA	QAdoc	TW	FC	NLI	STS	Avg.
Small Size (≤109M)								
thenlper-gte-small (Li et al., 2023)	33M	18.04	9.13	10.95	49.63	37.51	55.55	30.14
arctic-embed-m (Merrick et al., 2024)	109M	33.15	14.04	8.48	35.07	38.67	56.20	30.94
thenlper-gte-base (Li et al., 2023)	109M	9.18	5.42	37.91	60.45	42.52	46.07	33.59
arctic-embed-m-v1.5 (Merrick et al., 2024)	109M	25.76	30.41	17.95	47.97	42.88	64.16	38.19
arctic-embed-s (Merrick et al., 2024)	33M	38.58	28.81	21.33	47.21	39.85	66.96	40.46
bge-small-en-v1.5 (Xiao et al., 2024)	33M	56.91	51.19	25.15	55.17	32.87	64.54	47.64
bge-base-en-v1.5 (Xiao et al., 2024)	109M	51.50	52.78	46.72	59.93	41.16	68.63	53.45
DMRetriever-33M (ours)	33M	62.47†	57.03†	57.22†	60.81†	46.56†	67.57	58.61†
DMRetriever-109M (ours)	109M	**63.19**†	**59.55**†	**58.88**†	**62.48**†	**46.93**†	**68.79**†	**59.97**†
Medium Size (137M–335M)								
arctic-embed-m-long (Merrick et al., 2024)	137M	21.51	10.86	19.24	36.13	41.67	54.94	30.73
arctic-embed-l (Merrick et al., 2024)	335M	40.56	30.19	14.98	32.64	34.20	56.10	34.78
bge-large-en-v1.5 (Xiao et al., 2024)	335M	56.76	54.45	32.20	54.90	35.11	64.47	49.65
gte-base-en-v1.5 (Li et al., 2023)	137M	60.51	55.62	46.26	52.24	39.59	70.40	54.10
mxbai-embed-large-v1 (Lee et al., 2024b)	335M	64.24	62.63	39.94	58.12	40.18	68.01	55.52
arctic-embed-m-v2.0 (Merrick et al., 2024)	305M	61.22	62.20	47.01	57.79	42.29	64.51	55.84
DMRetriever-335M (ours)	335M	**67.44**†	**62.69**†	**62.16**†	**64.42**†	**49.69**†	**70.71**†	**62.85**†
Large Size (434M–1.5B)								
arctic-embed-l-v2.0 (Merrick et al., 2024)	568M	55.23	59.11	38.11	60.10	41.07	62.61	52.70
gte-large-en-v1.5 (Li et al., 2023)	434M	67.37	58.18	39.43	52.66	34.45	66.47	53.09
Qwen3-Embedding-0.6B (Zhang et al., 2025)	596M	66.10	52.31	62.38	64.89	50.30	67.39	60.56
mulling-e5-large-instruct (Wang et al., 2024)	560M	67.97	64.64	62.25	66.78	48.51	63.42	62.26
mulling-e5-large (Wang et al., 2024)	560M	66.99	64.01	62.81	59.87	50.93	74.12	63.12
gte-Qwen2-1.5B-instruct (Li et al., 2023)	1.5B	69.85	59.17	65.09	62.73	55.51	73.58	64.32
inf-retriever-v1-1.5b (Yang et al., 2025)	1.5B	69.41	64.29	62.99	65.39	54.03	73.92	65.01
DMRetriever-596M (ours)	596M	**72.44**†	**67.50**†	**65.79**†	**69.15**†	**55.71**†	**74.73**†	**67.55**†
XL Size (≥4B)								
Qwen3-Embedding-8B (Zhang et al., 2025)	7.6B	44.21	34.38	41.56	42.04	32.53	42.95	39.61
gte-Qwen2-7B-instruct (Li et al., 2023)	7.6B	70.24	47.41	63.08	31.62	53.71	74.88	56.82
NV-Embed-v1 (Moreira et al., 2024)	7.9B	68.06	62.70	56.02	59.64	48.05	67.06	60.26
Qwen3-Embedding-4B (Zhang et al., 2025)	4B	67.20	59.14	65.28	67.16	53.61	58.51	61.82
e5-mistral-7b-instruct (Wang et al., 2023)	7.1B	65.57	64.97	63.31	67.86	47.55	66.48	62.58
NV-Embed-v2 (Lee et al., 2024a)	7.9B	74.47	69.37	42.40	68.32	58.20	76.07	64.80
inf-retriever-v1 (Yang et al., 2025)	7.1B	72.84	66.74	66.23	65.53	51.86	75.98	66.53
SFR-Embedding-Mistral (Meng et al., 2024)	7.1B	71.41	67.14	69.45	70.31	50.93	72.67	66.99
Linq-Embed-Mistral (Kim et al., 2024)	7.1B	74.40	70.31	64.11	70.64	52.46	71.25	67.19
DMRetriever-4B (ours)	4B	75.32†	70.23†	70.55†	71.44†	57.63	77.38†	70.42†
DMRetriever-7.6B (ours)	7.6B	**76.19**†	**71.27**†	**71.11**†	**72.47**†	**58.81**†	**78.36**†	**71.37**†

Table 2: Experiment results across all six search intents at various scales in DisastIR-Test. **Bold** and underline denote the best and second-best scores within each size group. † indicates improvement over the best baseline is statistically significant with p-value < 0.05 evaluated using the one-tailed Wilcoxon signed-rank test.

Size	QA	QAdoc	TW	FC	NLI	STS	Avg
33M	57.72	55.83	57.40	59.17	41.55	64.71	56.06
109M	59.41	58.24	58.72	60.68	42.13	62.93	57.02
335M	64.10	61.36	62.09	63.33	44.86	66.55	60.38
596M	71.39	67.00	67.23	68.71	49.91	71.66	65.98
4B	72.35	68.68	**70.06**	71.09	53.69	74.47	68.39
7.6B	**73.47**	**69.77**	69.42	**71.44**	**53.91**	**75.56**	**68.93**

Table 3: Performance of DMRetriever-PT.

DevLite enables over 30× faster model development while maintaining reliable performance rankings (Kendall's $\tau > 0.90$), thus supporting rapid yet trustworthy experimentation.

4.2 Baselines and Metrics

Baselines are drawn from two sources: (1) models evaluated in DisastIR, and (2) the latest high-performing open-source models on the MTEB retrieval benchmark [5] (Muennighoff et al., 2022) that have not been evaluated on DisastIR.

Some baselines in DisastIR are fine-tuned with knowledge distillation (KD), which uses external supervision from a teacher model. In such cases, models are not trained solely on ground-truth data but are also guided by teacher's high-quality interpretation of it. We thus exclude these KD-based models as unfair comparisons, leaving 29 baselines ranging from 33M to 7.9B parameters. Detailed reasons for this exclusion are in Appendix G. Baseline implementations are detailed in Appendix H.

Model performance is evaluated using Normalized Discounted Cumulative Gain at rank 10 (NDCG@10) on DisastIR-Test. DMRetriever training setting is detailed in Appendix I.

[5] As of August 15, 2025.

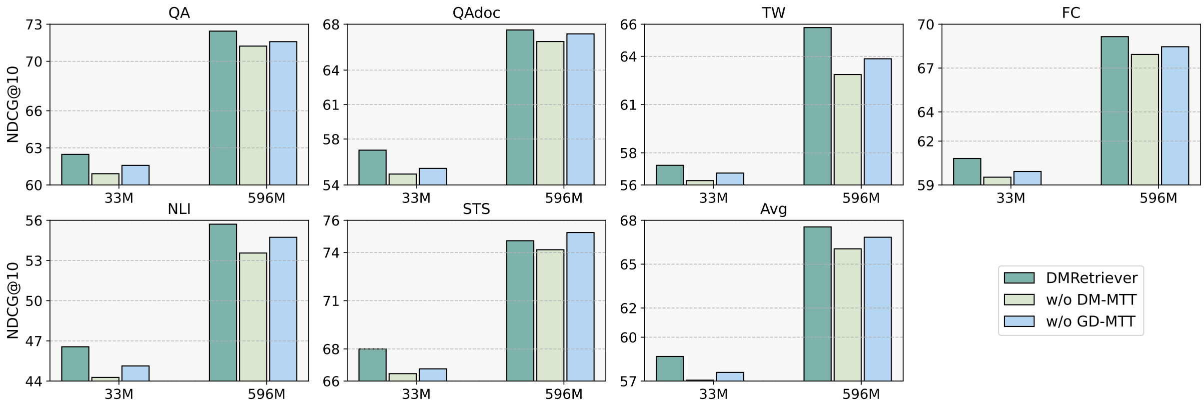

Figure 3: Effects of different fine-tuning datasets. DM-MTT and GD-MTT are datasets used during fine-tuning from disaster management and general domains. Effects of different pre-training datasets are shown in Figure 7.

5 Experiment Results and Analyses

We investigate three research questions: **RQ1:** How does DMRETRIEVER perform across scales compared to existing baselines? **RQ2:** Can our data refinement pipeline generate high-quality data for model learning? **RQ3:** Can our three-stage training framework enable models to effectively absorb intensive domain knowledge?

5.1 DMRETRIEVER Performance (RQ1)

Table 2 details DMRETRIEVER performance against all baselines across all six search intents. Across all scales, our model ranks first on every individual intent and surpasses the previous SOTA by up to 7 points in average performance (335M variant). This performance culminates in our largest model, DMRETRIEVER-7.6B, which sets a new SOTA on the benchmark with an average score of 71.37, surpassing the previous best by 4.3 points.

A key advantage of DMRETRIEVER is its high efficiency. The trade-off between performance and model size is noteworthy: The 33M, 109M, 335M, 596M, and 4B variants retain 82.2%, 84.0%, 88.1%, 94.6%, and 98.7% of the performance of the 7.6B model, while using only 0.4%, 1.4%, 4.4%, 7.8%, and 52.6% of its parameters. This parameter efficiency leads to superior cross-scale performance. For instance, DMRETRIEVER-596M surpasses all XL baselines ($\geq$4B) despite being over 13.3$\times$ smaller, and our smallest model, DMRETRIEVER-33M, outperforms all baselines in medium size range using only 7.6 % parameters.

5.2 DMRETRIEVER-PT Performance (RQ1)

We further evaluate DMRETRIEVER-PT (DMRETRIEVER-Pre-Training), which uses only unlabeled data for pre-training (§3.3.2). As shown in Table 3, it outperforms many supervised models reported in Table 2. Notably, 4B-PT surpasses all supervised baselines, while 596M-PT achieves comparable performance.

This strong unsupervised performance is especially valuable in disaster management, where labeled data is costly and scarce, making our approach a practical and data-efficient way to rapidly develop high-quality IR models for this domain.

5.3 Ablations on Data Refinement Pipeline (RQ2)

Ablations on Training Dataset Component Figures 3 and 7 highlight the importance of disaster management-specific and general-domain corpora in shaping model performance. General-domain data introduces diverse language patterns and semantic relevance, enhancing model robustness and adaptability (Xu et al., 2024a).

Disaster management corpus grounds model in the context of this domain, enhancing its ability to adapt to domain-specific scenarios.

This yields substantial improvements, particularly on search intents that demand intensive domain knowledge (e.g., NLI and Twitter) (Yin et al., 2025), with gains of up to 2.9 points on Twitter in 596M variant. These results underscore the importance of disaster management-specific data for effective domain adaptation and model performance.

Ablations on False Positive Filter Methods We further ablate our data refinement pipeline to assess its role in improving data quality and model performance. We begin with false positive filter, comparing (1) no filtering (NoPos) and (2) a consistency-based filter (CBF; details in Appendix J). Difficulty-

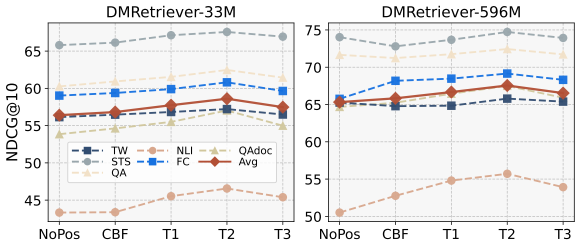

Figure 4: Ablation study of different false positive filtering methods. NoPos: No false positive filtering. CBF: Consistency-based filtering. T1–3: Top-N mutual-agreement filtering strategy, where $N = 1, 2, 3$.

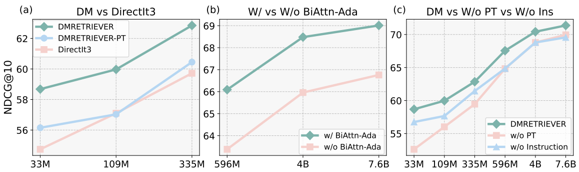

Figure 5: Ablations of progressive instruction fine-tuning. DirectIt3, BiAttn-Ada, DM-PT, and Ins mean fine-tuning directly using iteration 3 data (MTT-0.85), Bidirectional attention adaptation, DMRETRIEVER model, and instruction, respectively. Breakdown for each search intent is in Figures 9, 10, and 11.

aware hard negative mining is analyzed together with three-stage training framework in §5.4.

As shown in Figure 4, CBF method yields only marginal gains over NoPos, improving average performance by 0.4 and 0.5 points for 33M and 596M variants. In contrast, our top-N mutual-agreement filtering consistently outperforms both baselines across most search intents and in average performance, achieving up to 2.2 points higher average performance. This shows CBF struggles to filter low-quality pairs in disaster management, mainly due to the weaknesses of individual IR models in this domain, while our mutual-agreement approach addresses this limitation through model consensus.

An appropriate choice of N further enhances performance: a less strict top-1 agreement ($N = 1$) may retain noisy pairs, while a stricter top-3 agreement ($N = 3$) may exclude true positives, slightly affecting performance.

5.4 Ablations on Three-stage Training Frameworks (RQ3)

Ablations on Progressive Instruction Fine-tuning As shown in Figure 5a, progressive fine-tuning is essential for model performance. Skipping intermediate stages and directly fine-tuning DMRETRIEVER-PT on the final stage dataset (details in Appendix K) causes a sharp performance drop, with up to 1.4 points (33M variant) lower than its pre-trained variant. This degradation indicates a mismatch between model capacity and data difficulty, as the final-stage negatives are too challenging for the model's early-stage learning.

In contrast, our progressive method, which gradually increases the difficulty of hard negatives, enables steady improvement across iterations (Figure 8). This confirms the *necessity of progressive fine-tuning* and highlights *the difficulty-aware hard negative mining* that enables progressive training by providing controlled negatives.

We also find that adding intent-specific instructions to query significantly improves performance (Figure 5c), with gains of up to 2.3 points for 109M variant. This suggests that instruction tuning helps model better align its embeddings with diverse search intents (Asai et al., 2022; Wang et al., 2023).

Ablations on Pre-training Adaptation to bidirectional attention is vital for decoder-only backbones, yielding a significant performance improvement (Figure 5b) with up to 2.7 points improvement (596M variant). This indicates that bidirectional attention could yield more effective embeddings than causal attention, as it allows model to capture knowledge from future tokens (Lee et al., 2024a).

Contrastive pre-training is crucial for effective domain knowledge adaptation, especially for smaller models, yielding robust initialization for retrieval tasks (Figure 5c). The performance gains, however, decrease with increasing model size, as larger models may already possess the capacity to capture domain knowledge (Xu et al., 2024b).

6 Conclusion

We present DMRETRIEVER, a family of six dense retrieval models (33M to 7.6B) designed to advance information access in disaster management. DMRETRIEVER is trained through a novel three-stage training framework with training data produced by an advanced data refinement method.

Extensive experiments demonstrate that DMRETRIEVER sets a new SOTA across all six search intents within each model scale. Furthermore, DMRETRIEVER shows exceptional parameter efficiency, with the 596M version outperforming all XL baselines (≥4B) over 13.3× its size, and the 33M version surpassing all medium-sized competitors using only 7.6% of their parameters.

Limitations

While DMRETRIEVER achieves SOTA performance across all search intents and scales, several aspects merit further investigation. The current work focuses primarily on textual retrieval; extending it to multi-modal retrieval would enhance its applicability in real-world disaster scenarios. In addition, DMRETRIEVER is focused on English-language resources, and future work could explore multilingual and cross-lingual retrieval to broaden its global utility.

Ethics Statement

DMRETRIEVER is designed to support disaster management by enhancing retrieval performance across diverse search intents. All training data are drawn from publicly available sources, with no personally identifiable information included. We acknowledge potential risks of misuse, such as the amplification of rumors during disasters. To mitigate such concerns, DMRETRIEVER is released strictly for research purposes.

A potential issue is test set contamination, where some test samples may overlap with training data (Sainz et al., 2023). We conduct detailed data contamination analysis in Appendix C, where we do not observe any overlap between train and test sets.

Acknowledgments

This work used DeltaAI and Delta GPU at the National Center for Supercomputing Applications through allocation CIV250019, CIS250751, and CIV250021 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by U.S. National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

Reem Abbas and Todd Miller. 2025. Exploring communication inefficiencies in disaster response: Perspectives of emergency managers and health professionals. *International Journal of Disaster Risk Reduction*, 120:105393.

Firoj Alam, Hassan Sajjad, Muhammad Imran, and Ferda Ofli. 2021. Crisisbench: Benchmarking crisis-related social media datasets for humanitarian information processing. In *Proceedings of the International AAAI conference on web and social media*, volume 15, pages 923–932.

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic qa corpora generation with roundtrip consistency. *arXiv preprint arXiv:1906.05416*.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2022. Task-aware retrieval with instructions. *arXiv preprint arXiv:2211.09260*.

Parul Awasthy, Aashka Trivedi, Yulong Li, Mihaela Bornea, David Cox, Abraham Daniels, Martin Franz, Gabe Goodhart, Bhavani Iyer, Vishwajeet Kumar, and 1 others. 2025. Granite embedding models. *arXiv preprint arXiv:2502.20204*.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *European Conference on Information Retrieval*, pages 716–722. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Chao Fan, Cheng Zhang, Alex Yahja, and Ali Mostafavi. 2021. Disaster city digital twin: A vision for integrating artificial and human intelligence for disaster management. *International journal of information management*, 56:102049.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2015. Processing social media messages in mass emergency: A survey. *ACM computing surveys (CSUR)*, 47(4):1–38.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Junseong Kim, Seolhwa Lee, Jihoon Kwon, Sangmo Gu, Yejin Kim, Minkyung Cho, Jy yong Sohn, and Chanyeol Choi. 2024. Linq-embed-mistral:elevating text retrieval with improved gpt data through task-specific control and quality refinement. Linq AI Research Blog.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024a. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024b. Open source strikes bread - new fluffy embeddings model.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Zhewei Liu, Natalie Coleman, Flavia Ioana Patrascu, Kai Yin, Xiangpeng Li, and Ali Mostafavi. 2025. Artificial intelligence for flood risk management: A comprehensive state-of-the-art review and future directions. *International Journal of Disaster Risk Reduction*, 117:105110.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*.

Gabriel de Souza P Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. 2024. Nv-retriever: Improving text embedding models with effective hard-negative mining. *arXiv preprint arXiv:2407.15831*.

Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shalini Priya, Manish Bhanu, Sourav Kumar Dandapat, Kripabandhu Ghosh, and Joydeep Chandra. 2020. Taqe: tweet retrieval-based infrastructure damage assessment during disasters. *IEEE transactions on computational social systems*, 7(2):389–403.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark. *arXiv preprint arXiv:2310.18018*.

Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. 2024. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

UNDRR. 2020. Hazard definition and classification review technical report. Technical report, United Nations Office for Disaster Risk Reduction, Geneva, Switzerland. Supported by BMZ and USAID. Chair: Professor Virginia Murray.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Ran Xu, Hui Liu, Sreyashi Nag, Zhenwei Dai, Yaochen Xie, Xianfeng Tang, Chen Luo, Yang Li, Joyce C Ho, Carl Yang, and 1 others. 2024a. Simrag: Self-improving retrieval-augmented generation for adapting large language models to specialized domains. *arXiv preprint arXiv:2410.17952*.

Ran Xu, Wenqi Shi, Yue Yu, Yuchen Zhuang, Yanqiao Zhu, May D Wang, Joyce C Ho, Chao Zhang, and Carl Yang. 2024b. Bmretriever: Tuning large language models as better biomedical text retrievers. *arXiv preprint arXiv:2404.18443*.

Junhan Yang, Jiahe Wan, Yichen Yao, Wei Chu, Yinghui Xu, and Yuan Qi. 2025. inf-retriever-v1 (revision 5f469d7).

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Kai Yin, Xiangjue Dong, Chengkai Liu, Lipai Huang, Yiming Xiao, Zhewei Liu, Ali Mostafavi, and James Caverlee. 2025. Disastir: A comprehensive information retrieval benchmark for disaster management. *arXiv preprint arXiv:2505.15856*.

Kai Yin, Bo Li, Chengkai Liu, Ali Mostafavi, and Xia Hu. 2024. Crisissense-llm: Instruction fine-tuned large language model for multi-label social media text classification in disaster informatics. *arXiv preprint arXiv:2406.15477*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Algorithm 1: Three-stage training pipeline of DMRETRIEVER

Input:
Backbone $\mathcal{M}$ (encoder- or decoder-only); MLM training data $\mathcal{D}_{MLM}$; Difficulty level schedule $\{\alpha_t\}$ for progressive fine-tuning
Output: Final DMRETRIEVER model

Stage 1: Bidirectional attention adaptation (decoder-only backbone)
if $\mathcal{M}$ *is decoder-only* **then**
  Replace causal mask with full attention mask;
  Train $\mathcal{M}$ on $\mathcal{D}_{MLM}$ using the MLM objective
Stage 2: Unsupervised contrastive pre-training
Construct MTP via Algorithm 2;
Pre-train $\mathcal{M}$ on MTP using InfoNCE loss (Eq. 2) with in-batch negatives
Stage 3: Difficulty-aware Progressive Supervised Instruction Fine-tuning
foreach *iteration* t **do**
  Construct MTT-α_t via Algorithm 3 with difficulty level α_t;
  Fine-tune $\mathcal{M}$ on MTT-α_t using supervised InfoNCE loss (Eq. 4);
  Update $\mathcal{M}$ with the best validation checkpoint
return $\mathcal{M}$

Algorithm 2: Construction of MTP

Input:
$\mathcal{C}_{GD}$: general-domain corpora; $\mathcal{R}_{dm}$: raw disaster-management documents (e.g., PDFs); $\mathcal{I}$: search intents; Γ_i: heuristic rule for intent i; $\mathcal{E}_i^{need}, \mathcal{E}_i^{query}$: in-context examples for information-need generation, and for query–positive passage generation; $\mathrm{Prompt}^{need}, \mathrm{Prompt}^{query}$: prompt templates for information-need generation, and for query–positive passage generation
Output:
$\mathrm{MTP} = \mathrm{GD\text{-}MTP} \cup \mathrm{DM\text{-}MTP}$

1. Build DM-MTP:
foreach $d \in \mathcal{R}_{dm}$ **do**
  $\mathcal{C}_{dm} \leftarrow \mathrm{ProcessToPassages}(d)$;
foreach $p \in \mathcal{C}_{dm}$ **do**
  $s \leftarrow \mathrm{LLM}(p, \mathcal{I}, \mathcal{E}_i^{need}, \mathrm{Prompt}^{need})$;
  $(q, p^+) \leftarrow \mathrm{LLM}(s, p, \mathcal{E}_i^{query}, \mathrm{Prompt}^{query})$;
  $\mathrm{DM\text{-}MTP} \leftarrow \mathrm{DM\text{-}MTP} \cup \{(q, p^+)\}$;
2. Build GD-MTP:
foreach $r \in \mathcal{C}_{GD}$ **do**
  Identify intent i (by dataset design);
  $(q, p^+) \leftarrow \mathrm{ApplyHeuristic}(r, \Gamma_i)$;
  $\mathrm{GD\text{-}MTP} \leftarrow \mathrm{GD\text{-}MTP} \cup \{(q, p^+)\}$;
3. Combine and output:
$\mathrm{MTP} \leftarrow \mathrm{GD\text{-}MTP} \cup \mathrm{DM\text{-}MTP}$

A Details of DM-MTP Construction

We propose a three-stage pipeline to construct DM-MTP: (1) web crawling and processing of disaster-management domain PDF files, (2) information needs generation, and (3) query–positive passage pairs generation.

For web crawling, we construct queries in the format “event type name + location name + .pdf”. We specifically target PDFs because they are more likely to contain structured, information-rich, and credible content, often originating from peer-reviewed publications or official institutions. To ensure coverage, we consider 301 disaster event types as defined in UNDRR (2020) and combine them with the names of the 50 U.S. states, given both the availability of high-quality resources in the U.S. context and the fact that our model is designed for English-language data.

All collected PDF files are then processed into semantically coherent text chunks through a multi-step pipeline proposed by Yin et al. (2025): (1) exact-URL deduplication, where duplicate documents are removed by identifying identical download links; (2) text extraction and preprocessing, where each PDF is converted into plain text and non-textual elements such as tables and figures are removed; (3) locality-sensitive hashing (LSH) deduplication, which eliminates near-duplicate documents with overlapping content; (4) semantic chunking, where cleaned documents are segmented into passages of fewer than 256 tokens to balance retrievability and semantic integrity; and (5) embedding-based near deduplication, where dense embeddings are computed for all chunks, an ANN index is built, and pairs with cosine similarity above 0.9 are discarded. Chunks obtained are regarded as disaster management passages in DM-MTP.

In the second and third stages, given a passage and the definition of a search intent, an LLM generates information-need statement. The information

	QA	QAdoc	Twitter	FactCheck	NLI	STS
Top 1	NV-Embed-v2	Linq-Emb-Mis	SFR-Emb-Mis	Linq-Emb-Mis	NV-Embed-v2	NV-Embed-v2
Top 2	Linq-Emb-Mis	NV-Embed-v2	inf-retriever-v1	SFR-Emb-Mis	gte-Qwen2-1.5B-ins	inf-retriever-v1
Top 3	inf-retriever-v1	SFR-Emb-Mis	Linq-Emb-Mis	NV-Embed-v2	inf-retriever-v1-1.5b	gte-Qwen2-7B-ins

Table 4: Top-3 IR models across different search intents identified in DisastIR. Linq-Emb-Mis and SFR-Emb-Mis are shorten for Linq-Embed-Mistral and SFR-Embedding-Mistral

Task	Instruction Format
QA	Given the question, retrieve most relevant passage that best answers the question
QAdoc	Given the question, retrieve most relevant document that answers the question
Twitter	Given the user query, retrieve the most relevant Twitter text that meets the request
FactCheck	Given the claim, retrieve most relevant document that supports or refutes the claim
NLI	Given the premise, retrieve most relevant hypothesis that is entailed by the premise
STS	Given the sentence, retrieve the sentence with the same meaning

Table 5: Instruction formats used for each search intent.

statement together with the passage is used to construct a user query and its directly relevant positive passage. Prompts for query generation and relevant passage generation based on disaster management-related passages under different search intents are adapted from Yin et al. (2025).

In total, DM-MTP contains 3.3 million query–passage pairs, with an estimated generation cost of about $2,000 using the GPT-4o-mini API.

B Details of GD-MTP Construction

To construct DG-MTP, we generate query–positive passage pairs from publicly available unlabeled corpora using intent-specific strategies as follows: (1) For QA, QAdoc, and TW intents, the user question serves as the query, and its corresponding answer is the positive passage. (2) For FC intent, the claim is used as the query, and a document that supports or refutes it is selected as the positive passage. (3) For NLI intent, the premise is treated as the query and the entailed hypothesis as the positive passage. (4) For STS intent, one sentence from a similar pair is randomly chosen as the query and the other as the positive passage.

C Data Contamination Analysis

Following Wang et al. (2023), we conduct test set data contamination analysis from two aspects: query overlap and passage corpus overlap.

Query overlap. Following Wang et al. (2023), we perform a string-matching analysis between queries in the DisastIR-Test and those in the training sets (MTP and MTT), and find no overlap between training and test queries.

Passage corpus overlap. While sharing the same passage corpus (e.g., DBPedia, NQ, and TriviaQA all using Wikipedia) is a standard evaluation practice in information retrieval and is not regarded as contamination (Wang et al., 2023), we adopt a stricter criterion by performing a string-matching analysis between passages in DisastIR-Test and those in MTP and MTT, finding no overlap between the training and test corpora.

D Determination of α Value during Progressive Fine-tuning

We follow the ablation study results of Moreira et al. (2024) and set $\alpha = 0.95$ for DMRetriever-596M, 4B, and 7.6B.

For DMRetriever-33M, 109M, and 335M, we apply progressive fine-tuning across three iterations, training each model for one epoch per iteration. We consider three candidate ranges of α: $\{0.55, 0.65, 0.75\}$, $\{0.65, 0.75, 0.85\}$, and $\{0.75, 0.85, 0.95\}$. Model performance is assessed on the validation set (DisastIR-DevLite), and the range yielding the best validation performance is selected. Results show that models progressively trained on MTT-0.65, 0.75, and 0.85 could lead to the best validation performance and $\{0.65, 0.75, 0.85\}$ is thus selected.

E Details of DisastIR-DevLite and DisastIR-Test Construction

To facilitate efficient yet effective model development, we construct a lightweight validation set, referred to as DisastIR-DevLite (See Algorithm 4). For each search intent, we randomly sample 80 user queries from DisastIR to form DisastIR-

Algorithm 3: Construction of MTT-α

Input:
$\mathcal{Q}_{GD}$: queries from GD-MTP (with (q, p^+) pairs); $\mathcal{Q}_{DM}$: queries from DM-MTP; N: top-N passages for mutual-agreement; $\mathcal{M}_{top3}$, $\mathcal{M}_{best}$: top-3 and best IR models per intent (from DisastIR); α: difficulty level for hard negative selection

Output: $\text{MTT-}\alpha = \text{GD-MTT-}\alpha \cup \text{DM-MTT-}\alpha$

1. Mutual-agreement false positive filter
foreach $q \in \mathcal{Q}_{DM}$ **do**
  Identify intent i;
  foreach $m \in \mathcal{M}_{top3}(i)$ **do**
    Retrieve top-N passages $R_m(q)$;
  $\mathcal{C}_q \leftarrow \bigcap_{m \in \mathcal{M}_{top3}(i)} R_m(q)$;
  if $\mathcal{C}_q \neq \emptyset$ **then**
    Add $(q, \text{top-1}(\mathcal{C}_q))$ to $\mathcal{D}^+_{DM}$;
$\mathcal{D}^+ \leftarrow \mathcal{D}^+_{DM} \cup \mathcal{Q}_{GD}$;
2. Difficulty-aware hard negative mining
foreach $(q, p^+) \in \mathcal{D}^+$ **do**
  Identify intent i;
  Retrieve top-200 $R_{best}(q)$ using $\mathcal{M}_{best}(i)$;
  $\mathcal{H}_q \leftarrow \{p^- \in R_{best}(q) \setminus \{p^+\} \mid \text{sim}(q, p^-) < \text{sim}(q, p^+) \times \alpha\}$;
  Add $(q, p^+, \text{top-}k(\mathcal{H}_q))$ to $\text{DM-MTT-}\alpha$ if $q \in \mathcal{Q}_{DM}$ else $\text{GD-MTT-}\alpha$;
3. Combine and output:
$\text{MTT-}\alpha \leftarrow \text{GD-MTT-}\alpha \cup \text{DM-MTT-}\alpha$;
return $\text{MTT-}\alpha$

DevLite, while the remaining queries constitute DisastIR-Test. For each query in DisastIR-DevLite, we apply the top three IR models corresponding to its search intent, retrieving the top-10 passages from each model. The union of these top-10 passages forms a passage pool for that query. The final DisastIR-DevLite passage corpus is constructed as the union of the passage pools of all queries in DisastIR-DevLite, whereas DisastIR-Test passage corpus is the entire DisastIR passage corpus. Query–passage relevance labels for both DisastIR-DevLite and DisastIR-Test are directly inherited from the original DisastIR.

Algorithm 4: Construction of DisastIR-DevLite and DisastIR-Test

Input:
DisastIR corpus $\mathcal{C}$ with labeled queries $\mathcal{Q}$ and relevance $\mathcal{Y}$; Search intents $\mathcal{T}$; Top-3 IR models $\mathcal{M}_{top3}$ per intent; $N = 80$ queries per intent for DevLite, $k = 10$ passages per model

Output:
DisastIR-DevLite $(\mathcal{Q}_{\text{dev}}, \mathcal{C}_{\text{dev}}, \mathcal{Y}_{\text{dev}})$;
DisastIR-Test $(\mathcal{Q}_{\text{test}}, \mathcal{C}, \mathcal{Y}_{\text{test}})$

Step 1: Sample queries for DevLite and Test
foreach *intent* $t \in \mathcal{T}$ **do**
  Randomly sample N queries $\mathcal{Q}^{(t)}_{\text{dev}} \subset \mathcal{Q}^{(t)}$;
  Let $\mathcal{Q}^{(t)}_{\text{test}} \leftarrow \mathcal{Q}^{(t)} \setminus \mathcal{Q}^{(t)}_{\text{dev}}$
$\mathcal{Q}_{\text{dev}} \leftarrow \bigcup_t \mathcal{Q}^{(t)}_{\text{dev}}, \quad \mathcal{Q}_{\text{test}} \leftarrow \bigcup_t \mathcal{Q}^{(t)}_{\text{test}}$
Step 2: Build passage pool per DevLite query
foreach $q \in \mathcal{Q}_{dev}$ *with intent* t **do**
  foreach $m \in \mathcal{M}_{top3}(t)$ **do**
    Retrieve $\text{TopK}_m(q)$ from $\mathcal{C}$
  $\text{Pool}(q) \leftarrow \bigcup_{m \in \mathcal{M}_{top3}(t)} \text{TopK}_m(q)$
$\mathcal{C}_{\text{dev}} \leftarrow \bigcup_{q \in \mathcal{Q}_{\text{dev}}} \text{Pool}(q)$
Step 3: Inherit relevance labels from DisastIR
$\mathcal{Y}_{\text{dev}} \leftarrow \mathcal{Y}|_{\mathcal{Q}_{\text{dev}} \times \mathcal{C}_{\text{dev}}}; \quad \mathcal{Y}_{\text{test}} \leftarrow \mathcal{Y}|_{\mathcal{Q}_{\text{test}} \times \mathcal{C}}$
return DisastIR-DevLite and DisastIR-Test

F Efficiency and Effectiveness of DisastIR-DevLite

To evaluate the efficiency and effectiveness of DisastIR-DevLite, we compare it against DisastIR-DevFull, a counterpart that shares the same queries as DisastIR-DevLite but uses the complete DisastIR corpus. For our analysis, we sampled 100 checkpoints from each model, DMRETRIEVER-33M and DMRETRIEVER-596M, during training, and evaluated each checkpoint on both development sets. The evaluation metrics include average embedding time, as well as average and intent-level NDCG@10 score.

The results demonstrate substantial efficiency gains. As shown in Figure 6, DisastIR-DevLite is 34.8× and 30.5× faster than DisastIR-DevFull for DMRETRIEVER-33M and DMRETRIEVER-596M, respectively. Crucially, this speedup does not compromise effectiveness. The checkpoint performance rankings produced by DisastIR-DevLite are nearly identical to those from DisastIR-DevFull, achieving a Kendall's τ correlation of over 0.90 for both models (Table 9).

Dataset	#Queries	#Passages
DisastIR-DevLite	480	9,316
DisastIR-Test	9,120	239,704

Table 6: Statistics of DisastIR-DevLite and DisastIR-Test.

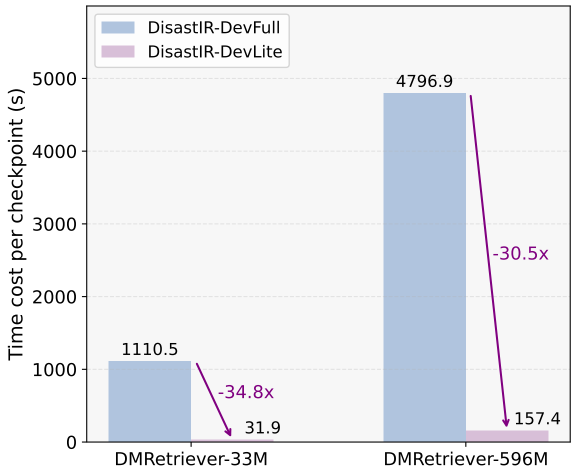

Figure 6: Efficiency comparison between DisastIR-DevLite and DisastIR-DevFull.

G Detailed Reasons for Excluding Knowledge Distilled (KD) Baselines

For baselines (Wang et al., 2022; Awasthy et al., 2025) that use knowledge distillation (KD) during fine-tuning, a cross-encoder is typically employed as the teacher model. This process introduces substantial external knowledge, as the teacher cross-encoder has a more powerful architecture. Unlike bi-encoders (our primary baselines and DMRetriever), cross-encoders process queries and documents jointly, allowing deep token-level interactions that produce highly accurate relevance scores. These fine-grained scores are then used as soft labels for the student model, offering a much richer and more informative training signal than the sparse binary hard labels available to other models.

Consequently, student model does not learn from ground-truth data alone, but is guided by teacher's high-quality interpretation of that data.

For this reason, we exclude all KD-based baselines, which include E5-small, base, large-v2 (Wang et al., 2022), and granite-embedding-125m-english (Awasthy et al., 2025), to ensure fair comparisons.

H Information on Baseline Models and Their Implementation

We adopt baseline models evaluated in DisastIR and additionally include recent models from MTEB that were not evaluated in DisastIR. They are categorized as four scales based on model parameter sizes: small (<=109M), medium (137M - 335M), large (434M- 1.5B), and extra large (XL) (>= 4B).

Detailed specifications of the selected models are provided in Table 12, while their HuggingFace links and licenses are listed in Table 13.

For each baseline model, we adhere to the official implementation guidelines to generate normalized query and passage embeddings. All evaluations are conducted in a zero-shot setting, with input sequences truncated to 512 tokens and a task-specific instruction prepended to each query.

I Implementation Details of DMRetriever Training

Backbones of DMRetriever are shown in Table 1. For bidirectional attention adaptation of decoder-only backbones, we train for 2 epochs with a global batch size of 256. The learning rates are set according to model size (see Table 7). All experiments are conducted on a single H200 GPU.

For unsupervised contrastive pre-training, each model is trained for 1 epoch. We use 3 H200 GPUs for encoder-only backbones and 4 H200 GPUs for decoder-only backbones. To optimize GPU memory consumption, we adopt DeepSpeed ZeRO Stage 1 (Rasley et al., 2020) and gradient checkpointing. Gradient accumulation is set to 1, and the per-GPU batch size is maximized to increase the diversity of in-batch negatives (Xiao et al., 2023), leading to the global batch sizes reported in Table 7.

For progressive supervised fine-tuning, we train encoder-only backbones for 1 epoch per iteration and decoder-only backbones for 1 epoch. Fine-tuning is conducted on a single H200 GPU, with gradient checkpointing disabled to improve training speed and gradient accumulation enabled. The learning rates and global batch sizes for each model variant are listed in Table 7.

Across all experiments, we use brain floating point (bfloat16) quantization and truncate input sequences to 512 tokens. The temperature value is set as 0.01. For each query, we set one positive passage and nine passages during pre-training and fine-tuning. For decoder-only backbones, we apply LoRA with rank $r = 16$ and $\alpha = 32$, following prior work (Lee et al., 2024a; Wang et al., 2023; Xu et al., 2024b), leaving hyperparameter tuning for future exploration.

J Implementation of Consistency-based filtering (CBF) Method

To implement the CBF method, we follow the methodology of Wang et al. (2022). Specifically, for each search intent, we use its best-performing retrieval model identified from the DisastIR benchmark as a reference (Table 4 for best-performing retrieval models). A synthetic pair is then discarded if its LLM-generated passage is not ranked within the top-2 results retrieved by this reference model.

K Implementation of Ablations of Progressive Fine-tuning

In the ablation study of progressive fine-tuning, we fine-tune DMRETRIEVER-33M-PT, 109M-PT, and 335M-PT on MTT-0.85 for 3 epochs. All other hyperparameters, including learning rate and batch size, are kept the same as in progressive fine-tuning which are detailed in Appendix I.

L Performance of DMRETRIEVER-KD

To establish a fair comparison with baselines like the E5 series that employ knowledge distillation (KD) during model fine-tuning and boost performance of our smaller models, we apply KD to our fine-tuned DMRETRIEVER-33M, 109M, AND 335M models. This process yields DMRETRIEVER-KD series (see Appendix M for KD settings). As shown in Table 8, our DMRETRIEVER-KD achieves the best average performance within their respective scales, securing the top rank across the majority of search intents.

The parameter efficiency of these models is remarkable. Notably, DMRETRIEVER-33M-KD surpasses all 560M-series baselines despite being 17× smaller. Furthermore, DMRETRIEVER-335M-KD matches models in the 7B-parameter class, which are over 20× larger. This makes our KD-series models ideal for real-world disaster management applications, where computational resources are limited and low latency is critical.

M Knowledge Distillation Settings

We apply knowledge distillation (KD) to our DMRETRIEVER-33M, 109M, AND 335M models, creating their distilled counterparts, denoted as the DMRETRIEVER-KD series. The distillation is performed using a specially constructed dataset, MTT-0.95-KD.

This dataset is an extension of MTT-0.95, the construction of which is detailed in Section 3.4 and Algorithm 3 (Lines 10–16). To create KD version, we use the best-performing IR model, as identified in DisastIR, for each search intent as a teacher to generate query-passage similarity scores. These scores are then added to the dataset as soft labels.

The training objective for knowledge distillation, $\mathcal{L}_{\text{KD}}$, is formulated as a weighted sum of two components. It combines the contrastive loss, $\mathcal{L}_{\text{CL}}$ (from Eq. 4), on the original hard labels with a Kullback-Leibler (KL) divergence term, $\mathcal{L}_{\text{KL}}$, for distilling the soft labels from the teacher. The balance between these two is controlled by a hyperparameter λ, as shown below:

$$\mathcal{L}_{\text{KD}} = \mathcal{L}_{\text{CL}} + \lambda \mathcal{L}_{\text{KL}}$$

In this work, we set the weighting factor λ to 1 for all distillation experiments.

	33M	109M	335M	596M	4B	7.6B
Bidirectional Adaptation Stage						
Learning Rate	-	-	-	1e-4	5e-5	2e-5
Batch Size	-	-	-	256	256	256
Contrastive Pre-training						
Learning Rate	4e-4	3e-4	1e-4	9e-5	4e-5	2e-5
Batch Size	2400	732	384	384	192	140
Supervised Fine-tuning						
Learning Rate	4e-5	1e-5	9e-6	6e-5	2e-5	1e-5
Batch Size	1024	512	512	512	128	64

Table 7: Training hyperparameters for different model sizes during the three-stage training framework.

Model	Scale	QA	QAdoc	TW	FC	NLI	STS	Avg
Small Size (≤109M)								
e5-base-V2 (2022)	109M	65.52	62.68	57.53	61.86	45.40	74.38	61.23
e5-small-V2 (2022)	33M	65.79	62.64	60.12	61.80	46.99	**74.54**	61.98
DMRetriever-33M-KD (ours)	33M	68.28	64.31	62.92	64.88	49.93	72.70	63.84
DMRetriever-109M-KD (ours)	109M	**69.61**	**66.14**	**65.55**	**66.30**	**52.50**	73.95	**65.68**
Medium Size (109M–434M)								
granite-embedding-125m (2025)	125M	64.58	60.73	46.47	62.45	48.05	71.61	58.98
e5-large-V2 (2022)	335M	59.92	63.07	55.37	61.85	50.69	**74.69**	60.93
DMRetriever-335M-KD (ours)	335M	**71.86**	**67.56**	**63.56**	**67.16**	**52.96**	73.74	**66.14**

Table 8: Performance of knowledge distillation enhanced retrieval models across multiple search intents.

Model	QA	QAdoc	TW	FC	NLI	STS	Overall
DMRetriever-33M	0.8467	0.7504	0.7326	0.8859	0.8610	0.7255	0.9008
DMRetriever-596M	0.8995	0.7937	0.8042	0.9101	0.8730	0.7725	0.9033

Table 9: Effectiveness of DisastIR-DevLite. Kendall's τ value between DisastIR-DevLite and DisastIR-DevFull.

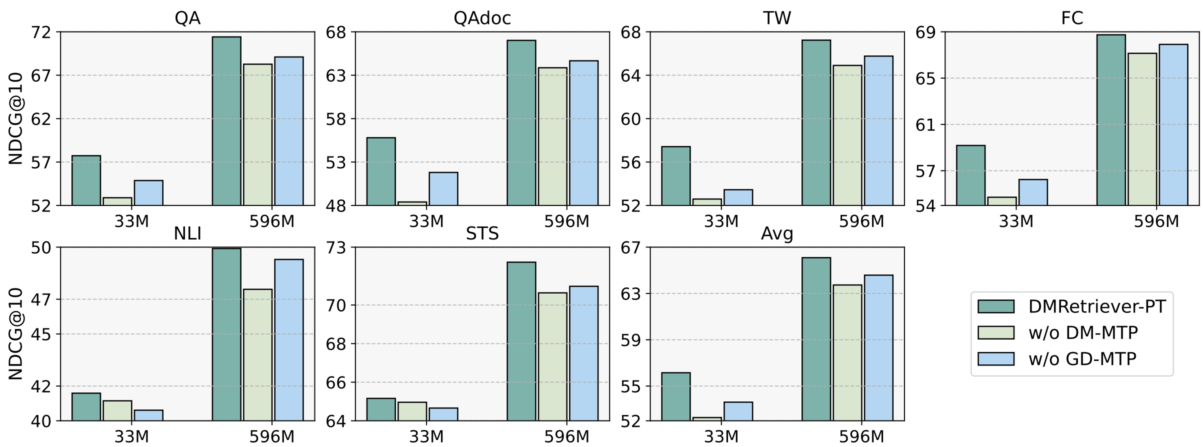

Figure 7: Effects of different pre-training datasets. DM-MTP and GD-MTP are datasets used during pre-training from disaster management and general domains.

Dataset	Size	Task	Link
GD-MTP:			
Squad	87.6K	QA	https://huggingface.co/datasets/sentence-transformers/squad
Paq	4M*	QA	https://huggingface.co/datasets/sentence-transformers/paq
MSMARCO	1M	QA	https://huggingface.co/datasets/microsoft/ms_marco
Gooaq	3M*	QA	https://huggingface.co/datasets/sentence-transformers/gooaq
Eli5	325k	QA	https://huggingface.co/datasets/sentence-transformers/eli5
Nfcorpus	134K	QAdoc	https://huggingface.co/datasets/mteb/nfcorpus
MSMARCO-Document-Ranking	250k*	QAdoc	https://microsoft.github.io/msmarco/Datasets
Hotpotqa	170k	QAdoc	https://huggingface.co/datasets/BeIR/hotpotqa
Signal1m-generated-queries	1M*	Twitter	https://huggingface.co/datasets/BeIR/signal1m-generated-queries
Fever	250k*	FactCheck	https://huggingface.co/datasets/mteb/fever
Quora	15k	STS	https://huggingface.co/datasets/mteb/quora
Quora-duplicates	950k*	STS	https://huggingface.co/datasets/sentence-transformers/quora-duplicates
Altlex	113k	STS	https://huggingface.co/datasets/sentence-transformers/altlex
XNLI	133k*	NLI	https://huggingface.co/datasets/mteb/xnli
All-NLI	321k*	NLI	https://huggingface.co/datasets/sentence-transformers/all-nli
DM-MTP	3.29M		
Total	15M		

Table 10: Composition of the MTP dataset used for unsupervised contrastive pre-training in Stage 2. * indicates that only a subset of the original dataset is used.

Dataset	Size	Task	Link
GD-MTT:			
MSMARCO	141k	QA	https://huggingface.co/datasets/microsoft/ms_marco
MSMARCO-Document-Ranking	57k	QAdoc	https://microsoft.github.io/msmarco/Datasets
Fever	81k	FactCheck	https://huggingface.co/datasets/mteb/fever
Quora-duplicates	4k	STS	https://huggingface.co/datasets/sentence-transformers/quora-duplicates
XNLI	96k	NLI	https://huggingface.co/datasets/mteb/xnli
All-NLI	145k	NLI	https://huggingface.co/datasets/sentence-transformers/all-nli
Signal1m-generated-queries	200k*	Twitter	https://huggingface.co/datasets/BeIR/signal1m-generated-queries
DM-MTT	412k		
Total	1.14M		

Table 11: Composition of **MTT** used for supervised instruction fine-tuning in Stage 3. *: After applying false positive filtering, we randomly sampled 200K examples from the remaining dataset. A portion of subsets in GD-MTP undergo hard negative mining (§3.4.3) to get GD-MTT.

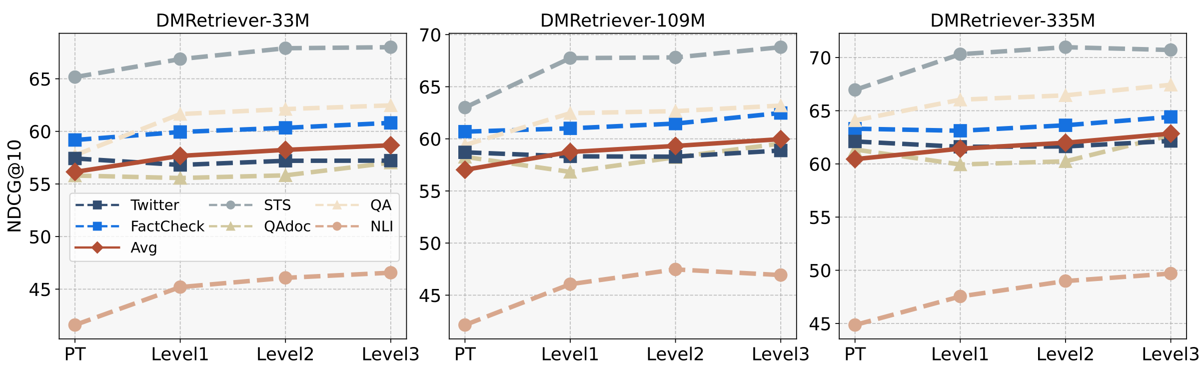

Figure 8: Model performance during different iterations of progressively supervised instruction fine-tuning

Model Name	Param Size	Size Bin	Base Model	Embed. Size	Arch.
Qwen3-Embedding-8B	7.6B	XL	Qwen3-8B	4096	decoder
Qwen3-Embedding-4B	4B	XL	Qwen3-4B	2560	decoder
inf-retriever-v1	7B	XL	gte-Qwen2-7B-instruct	3584	decoder
NV-Embed-v2	7B	XL	Mistral-7B-v0.1	4096	decoder
Qwen3-Embedding-0.6B	0.6B	Large	Qwen3-0.6B	1024	decoder
inf-retriever-v1-1.5b	1.5B	Large	gte-Qwen2-1.5B-instruct	1536	decoder
Linq-Embed-Mistral	7B	XL	E5-mistral-7b-instruct	4096	decoder
NV-Embed-v1	7B	XL	Mistral-7B-v0.1	4096	decoder
SFR-Embedding-Mistral	7B	XL	E5-mistral-7b-instruct	4096	decoder
snowflake-arctic-embed-l	335M	Medium	e5-large-unsupervised	1024	encoder
snowflake-arctic-embed-l-v2.0	568M	Large	gte-multilingual-mlm-base	1024	encoder
snowflake-arctic-embed-m-v2.0	305M	Medium	bge-m3-retromae	768	encoder
gte-Qwen2-7B-instruct	7B	XL	Qwen2-7B	3584	decoder
snowflake-arctic-embed-m-v1.5	109M	Small	BERT-base-uncased	768	encoder
e5-mistral-7b-instruct	7B	XL	Mistral-7b	4096	decoder
snowflake-arctic-embed-m	109M	Small	e5-unsupervised-base	764	encoder
snowflake-arctic-embed-m-long	137M	Medium	nomic-embed-text-v1-uns	768	encoder
bge-large-en-v1.5	335M	Medium	–	1024	encoder
mxbai-embed-large-v1	335M	Medium	–	1024	encoder
snowflake-arctic-embed-s	33M	Small	e5-unsupervised-small	384	encoder
bge-base-en-v1.5	109M	Small	–	768	encoder
bge-small-en-v1.5	33M	Small	–	384	encoder
multilingual-e5-large-instruct	560M	Large	xlm-roberta-large	1024	encoder
thenlper-gte-base	109M	Small	EBRT-base	768	encoder
multilingual-e5-large	560M	Large	xlm-roberta-large	1024	encoder
thenlper-gte-small	33M	Small	MiniLM-L12-H384	384	encoder
gte-Qwen2-1.5B-instruct	1.5B	Large	Qwen2-1.5B	1536	decoder
gte-base-en-v1.5	137M	Medium	EBRT-base	768	encoder
gte-large-en-v1.5	434M	Large	EBRT-large	1024	encoder

Table 12: Information of all baseline models. "–" means no publicly available information is available.

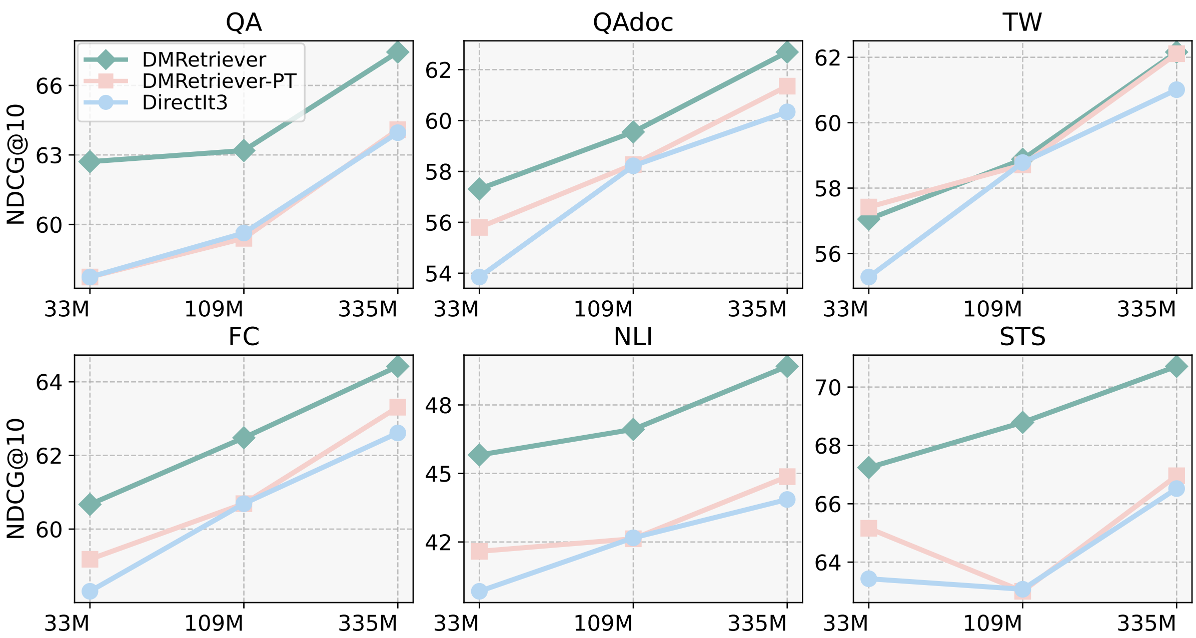

Figure 9: Comparisons of model performance under progressive fine-tuning and direct iteration-3 fine-tuning across 6 search intents.

Model Name	Link	License
Qwen3-Embedding-8B	https://huggingface.co/Qwen/Qwen3-Embedding-8B	apache-2.0
Qwen3-Embedding-4B	https://huggingface.co/Qwen/Qwen3-Embedding-4B	apache-2.0
inf-retriever-v1	https://huggingface.co/infly/inf-retriever-v1	apache-2.0
NV-Embed-v2	https://huggingface.co/nvidia/NV-Embed-v2	cc-by-nc-4.0
Qwen3-Embedding-4B	https://huggingface.co/Qwen/Qwen3-Embedding-0.6B	apache-2.0
inf-retriever-v1-1.5b	https://huggingface.co/infly/inf-retriever-v1-1.5b	apache-2.0
Linq-Embed-Mistral	https://huggingface.co/Linq-AI-Research/Linq-Embed-Mistral	cc-by-nc-4.0
NV-Embed-v1	https://huggingface.co/nvidia/NV-Embed-v1	cc-by-nc-4.0
SFR-Embedding-Mistral	https://huggingface.co/Salesforce/SFR-Embedding-Mistral	cc-by-nc-4.0
snowflake-arctic-embed-l	https://huggingface.co/Snowflake/snowflake-arctic-embed-l	apache-2.0
snowflake-arctic-embed-l-v2.0	https://huggingface.co/Snowflake/snowflake-arctic-embed-l-v2.0	apache-2.0
snowflake-arctic-embed-m-v2.0	https://huggingface.co/Snowflake/snowflake-arctic-embed-m-v2.0	apache-2.0
gte-Qwen2-7B-instruct	https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct	apache-2.0
snowflake-arctic-embed-m-v1.5	https://huggingface.co/Snowflake/snowflake-arctic-embed-m-v1.5	apache-2.0
e5-mistral-7b-instruct	https://huggingface.co/intfloat/e5-mistral-7b-instruct	mit
snowflake-arctic-embed-m	https://huggingface.co/Snowflake/snowflake-arctic-embed-m	apache-2.0
snowflake-arctic-embed-m	https://huggingface.co/Snowflake/snowflake-arctic-embed-m-long	apache-2.0
granite-embedding-125m-english	https://huggingface.co/ibm-granite/granite-embedding-125m-english	mit
bge-large-en-v1.5	https://huggingface.co/BAAI/bge-large-en-v1.5	apache-2.0
snowflake-arctic-embed-s	https://huggingface.co/Snowflake/snowflake-arctic-embed-s	mit
bge-base-en-v1.5	https://huggingface.co/BAAI/bge-base-en-v1.5	mit
bge-small-en-v1.5	https://huggingface.co/BAAI/bge-small-en-v1.5	mit
multilingual-e5-large-instruct	https://huggingface.co/intfloat/multilingual-e5-large-instruct	mit
thenlper-gte-base	https://huggingface.co/thenlper/gte-base	mit
multilingual-e5-large	https://huggingface.co/intfloat/multilingual-e5-large-instructt	apache-2.0
thenlper-gte-small	https://huggingface.co/thenlper/gte-small	mit
gte-Qwen2-1.5B-instruct	https://huggingface.co/Alibaba-NLP/gte-Qwen2-1.5B-instruct	mit
gte-base-en-v1.5	https://huggingface.co/Alibaba-NLP/gte-base-en-v1.5	apache-2.0
gte-large-en-v1.5	https://huggingface.co/Alibaba-NLP/gte-large-en-v1.5	mit

Table 13: HuggingFace model links and licenses for all evaluated models.

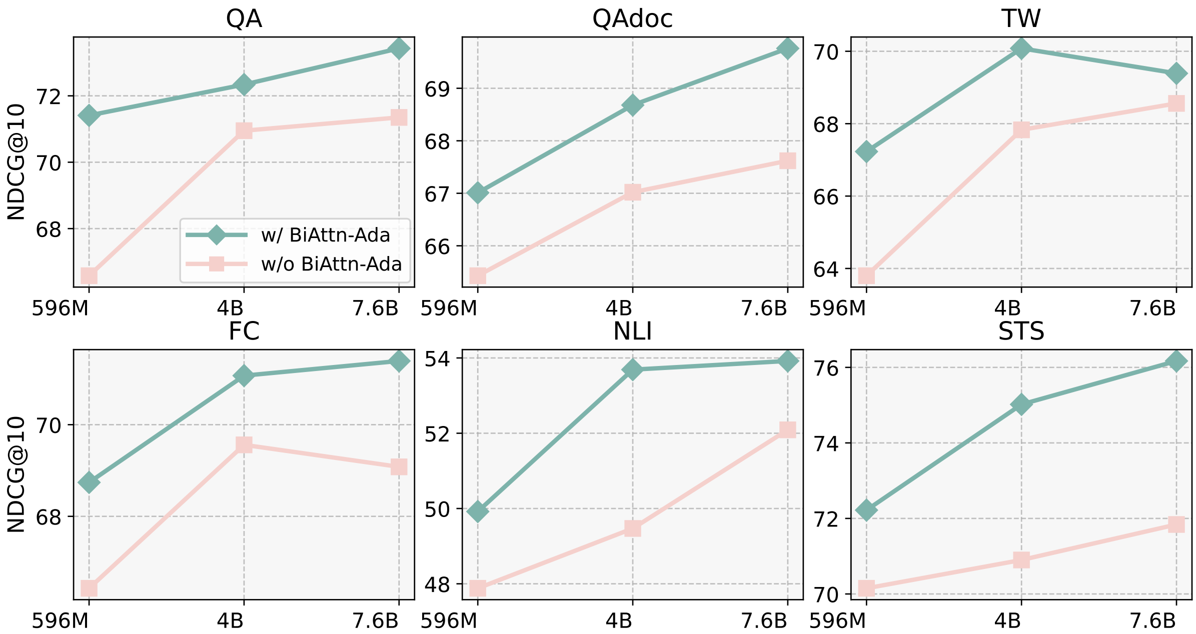

Figure 10: Model performance comparisons between with and without bidirectional attention adaptation across 6 search intents.

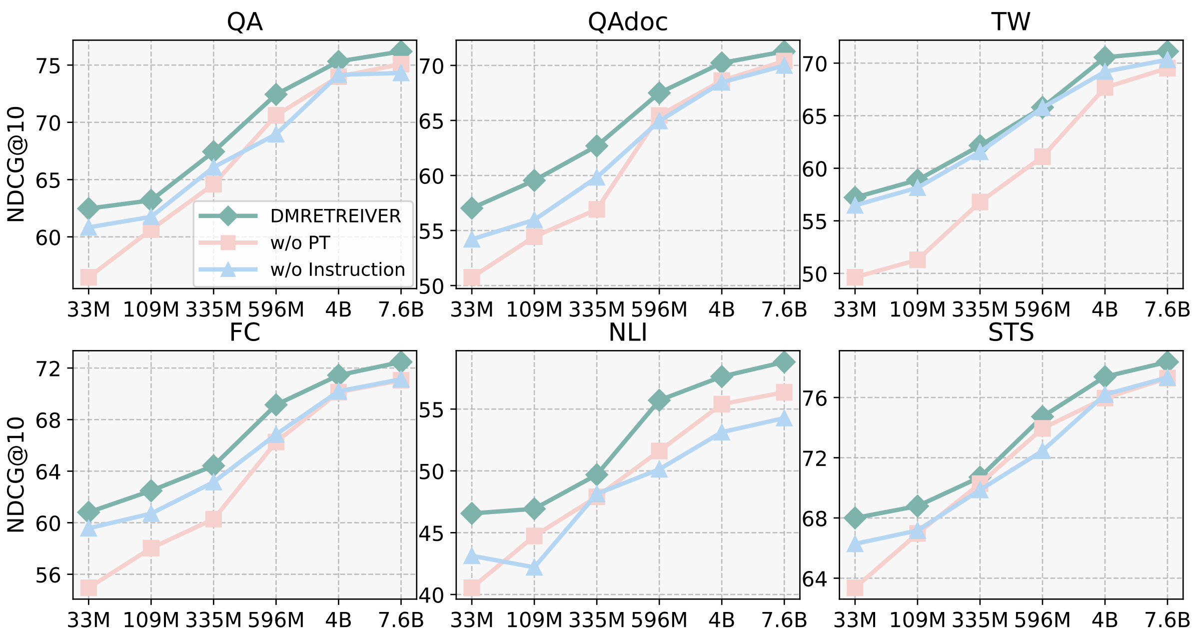

Figure 11: Model performance under ablations of whether conducting pre-training and adding instruction across 6 search intents.