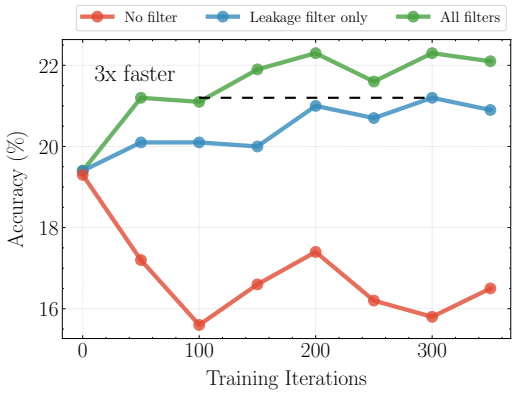

Stage	Number (%Total)
Source Articles	248,321
Question Generation	744,963 (100%)
Validation	295,274 (40%)
Best Question Selection	157,260 (21%)
Fixing Leakage	150,500 (20%)
Answer Type Filtering	62,279 (8%)
Resolving after 2024-01-01	52,183 (7%)
Final Set	52,183 (7%)

Table 1: (Left) **Benefits of our filtering stage.** Without leakage removal (red), the model worsens at forecasting, possibly learning shortcuts. With only the leakage removal step (blue), we find that achieving the same performance requires 3× more compute and data. Applying all filtering steps (green) leads to higher accuracy. (Right) Number of questions after each filtering stage.

Generating samples. One practical issue we face is that many top news sources, such as The Reuters and Associated Press (AP), have disallowed scraping even for CommonCrawl, due to the rise of commercial use in language model training (Grynbaum & Mac, 2023; Longpre et al., 2025). Still, we are able to collect articles from popular outlets spanning diverse geographies and topics.

Particularly, for our training set, we start with $\sim 248,000$ deduplicated English-language articles between June 2023 to April 2025 from *Forbes*, *CNN*, *Hindustan Times*, *Deutsche Welle*, and *Irish Times*. The distribution is described in Figure 17. From each article, the sample creator produces three forecasting samples, yielding $\sim 745,000$ samples.

Filtering samples. Table 1 (right) contains a breakdown of questions remaining after each filtering stage. 60% of question-answer candidates are marked invalid — most commonly because the article does not unambiguously resolve the question to the given answer. At this stage, zero questions remain from 40% articles, and 21% articles yield exactly one valid question, which we keep as is. For the 39% with multiple valid questions, the sample selector picks the best one. Finally, we remove samples with numeric answers.

Editing to fix leakage. Despite explicit prompts to avoid it, over 40% of selected questions directly contain the answer string. The sample selector's rewriting and rejection removes ∼90% of such cases. We apply a string matching filter to remove the remaining questions with such direct leakage and finally keep only those questions which resolve after January 1, 2024.

Validation Set. We use the same recipe to create a validation set of 207 questions using 500 randomly sampled articles from TheGuardian in the month of July 2025. To ensure high-quality, we used `o4-mini-high`, a much more capable model than DeepSeek-v3, to generate the seed questions. It is $\sim 10\times$ costlier, but followed the generation guidelines better, leading to $\sim$ 40% retention rate (207 final questions out of 500 starting articles).

Result 1: Filtering Improves Performance. Table 1 (left) shows the effect of our filtering steps. We train Qwen3-8B using RL with identical hyperparameters on three data variants: (red) 30,000 original generated questions, without any filtering; (blue) 30,000 samples obtained after the question editing step to remove leakage; and (green) 10,000 samples sourced from Forbes and included in `OpenForesight`. First, we observe the drastic impact of leakage in training. Training without leakage removal (red) worsens the model, perhaps due to shortcut learning. After the leakage removal steps, training improves the model (blue line). Yet, using all filtering stages (green line) leads to both higher accuracy and Brier score, in one-third the data and training steps.

In Section B.1, we also ablate the effect of training on binary-only, free-form-only, and combined binary and free-form data for `Qwen3-8B`. We find that free-form data is crucial for improving open-ended forecasting, however, training solely on freeform data does not

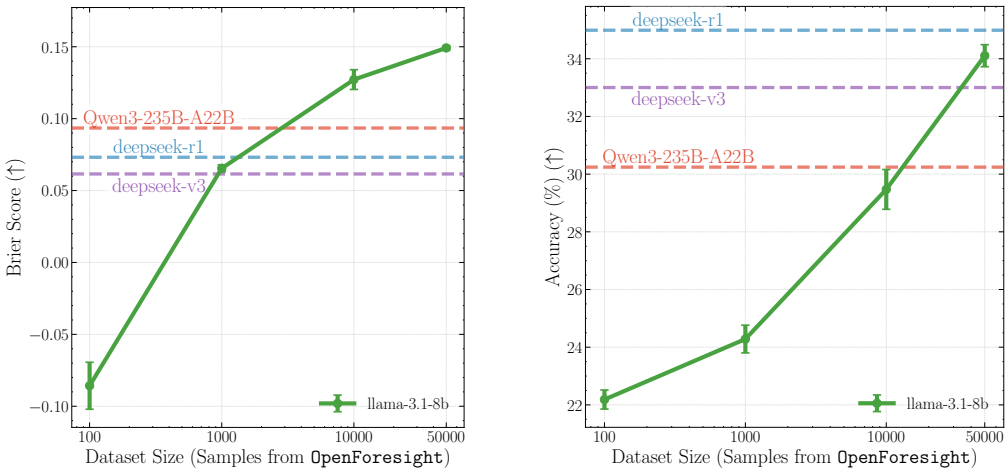

Figure 3: **Benefits of scaling training data.** We take the best scores from training Llama-3.1-8B-Instruct on different sized subsets of `OpenForesight`. We see continued improvements in both accuracy and brier as the data size increases, eventually making Llama-3.1-8B-Instruct match or surpass much larger, more recent models.

improve performance on binary Metaculus questions. Training with both kind of questions achieves the best trade-off, so we use this for our final training runs presented in Section 6.

Final training dataset. Across stages, we remove $\sim 90\%$ of questions, yielding a training set of 52K samples, each drawn from a unique article. We evaluated Qwen3-32B on this corpus while providing it the source article alongside the generated questions. The model achieves 95% accuracy confirming high validity. We release this training dataset, as `OpenForesight`.

Result 2: Continued Improvements from Scaling Training Data. Figure 3 shows the effect of scaling training dataset size. For each dataset size, we report test set (described later) results from the best validation checkpoint over prolonged training runs. We use Llama-3.1-8B-Instruct as it starts without any RL post-training. We see continued, and large gains in both accuracy and Brier score as we continue to scale up the training data, demonstrating the importance of our contribution in releasing `OpenForesight`.

5 Prediction System

We now present intermediate results that guided the design decisions for our prediction system. We did not measure performance on the held-out test set throughout development, making decisions solely based on our validation set. *We did not any find any notable difference between training in temporal order (sorted by resolution date), compared to training in a randomly shuffled order*. Below we present results which show the benefits of our reward design for RL training and retrieval system.

Reward Design. For training with RL, we compare three reward functions:

1. **Only Accuracy** *(Baseline):* $R = \mathbb{1}_{y \equiv y^*}$. Vanilla success rewards are commonly used in literature on LLM RL with verifiable rewards (Guo et al., 2025).
2. **Only Brier score** *(Damani et al. (2025)):* $R = S'(q, y, y^*) = -q^2 + \mathbb{1}_{y \equiv y^*} \cdot 2q$. This incentivizes both correct predictions and calibrated confidence estimates.
3. **Accuracy + Brier score** *(Ours):* $R = \mathbb{1}_{y \equiv y^*} + S'(q, y, y^*)$. We hypothesise that optimizing the Brier score alone might hurt exploration as when the model assigns a low confidence to its prediction, the correctness has a small impact on the Brier score. To fix this, we propose adding the accuracy term as well. In this case, even on hard questions, which merit low confidence, models get a large boost for correct predictions.

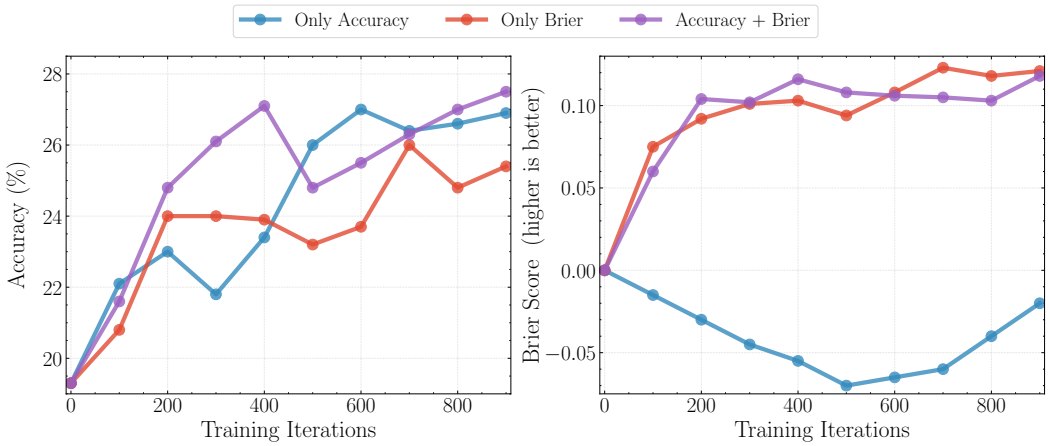

Figure 4: **Accuracy + Brier score reward performs the best.** Accuracy alone leads to poor calibration. While brier score incentivizes both correct predictions and calibration, on hard questions with low confidence, it provides little signal on correctness. Adding the accuracy term boosts exploration in this situation.

Result 3: Accuracy + Brier score as the reward improves performance. Figure 4 shows the validation set results of training with all three reward functions on the full `OpenForesight` dataset. We observe that optimizing accuracy alone (blue line) leads to negative brier scores, worse than a constant (0) baseline. In contrast, optimizing the Brier score alone (red line) also improves the accuracy, but to a lesser extent. Our proposed reward, accuracy + Brier (purple line), leads to the best performance on both metrics. It improves accuracy beyond the Brier alone while maintaining equal brier score on the validation set. Analyzing output distributions, we find that the brier-only trained model predicts "Unknown" with near-0 confidence in $\sim$ 40% of samples, due to low reward for correct yet low-confidence guesses, which hurts exploration. In contrast, our proposed reward yields "Unknown" in only $\sim$ 4% of samples, making low-confidence guesses on hard cases—improving both accuracy and Brier score.

Retrieval. Like prior work (Zou et al., 2022; Halawi et al., 2024), we provide the same relevant recent information across forecasting models. This provides them new evidence, or competing viewpoints to weigh, that was available before the resolution date, but potentially after the model's training cutoff. To prevent leakage issues (Paleka et al., 2025a), we use our offline CCNews corpus, and only use articles up to *one month* before the question's resolution date. Our overall pool consists of 1 million articles across 60 different sources. We de-duplicate the articles and split each into fixed-size chunks (512 tokens) and embed each chunk with the Qwen3-embedding 8B model. During evaluation, we retrieve the top-k most relevant chunks and append them to the model prompt in order as context.

Retrieval leads to large improvements in accuracy. As shown in Section 5, providing retrieved information improves accuracy by $9-18\%$ across model families and sizes. In Appendix Figure 10, we vary the number of retrieved chunks and find that improvements plateau after five chunks. Thus, we fix $k = 5$ chunks for all evaluations henceforth.

Training the final forecasting system. Based on the above design decisions guided by validation set performance, we now describe our final training methodology: We use the Qwen3-8B embedding model to retrieve the 5 most relevant chunks from news articles until a month before each question's resolution date. During training, we add a random number of such article chunks (between 0 to 5) in the prompt to make our forecaster generalizable to variable number of articles. We train the Qwen3-8B thinking model with GRPO on `OpenForesight` which has $\sim$ 50,000 samples and also include 2000 binary resolved questions from Metaculus (from 2024), both with retrieval. For the reward, we use our Accuracy + Brier score for free-form questions and only brier score for binary questions. We provide configuration details for training and evaluation in Section C.1.