DYNAMIC KNOWLEDGE ELICITATION: LEVERAGING STUDENT FEEDBACK FOR IMPROVED LANGUAGE MODEL DISTILLATION

Reuven Muller
Department of Information Technology
Kennesaw State University
Kennesaw, USA
rmuller7@students.kennesaw.edu

Ying Xie
Department of Information Technology
Kennesaw State University
Kennesaw, USA
yxie2@kennesaw.edu

Linh Le
Department of Information Technology
Kennesaw State University
Kennesaw, USA
lle13@kennesaw.edu

Shaoen Wu
Department of Information Technology
Kennesaw State University
Kennesaw, USA
swu10@kennesaw.edu

***Abstract*—Large Language Models (LLMs) have revolutionized natural language processing but remain resource-heavy and impractical for many organizations to deploy locally. Smaller specialist models offer a viable alternative, often developed using Knowledge Distillation (KD). Traditional KD methods, however, rely on static source datasets to elicit knowledge from the teacher model, limiting their ability to dynamically address student model weaknesses during training. This research introduces two adaptive knowledge elicitation methods: "Feedback-Driven Question Generation" and "Targeted Prompt Question Generation". Both methods iteratively expand the training dataset based on student model performance, leveraging a teacher model to target specific deficiencies. Using a Python QA task as a case study, our results show that both our methods enhance the student model's accuracy and response quality, with Method 2, which incorporates specialized prompt configurations, outperforming Method 1. These findings highlight the promise of adaptive KD in bridging the gap between large generalist AI models and smaller domain-specific models. Furthermore, these methods demonstrate an effective data augmentation technique for generating synthetic data specifically tailored to address student model weaknesses without overfitting or resulting in catastrophic forgetting.**

Keywords—Knowledge Distillation, Knowledge Elicitation, Data Augmentation, Synthetic Data Generation

I. INTRODUCTION

Large Language Models (LLMs) have significantly advanced natural language processing (NLP) across various applications [1]. However, their high computational and financial demands make local deployment impractical for many organizations [2]. While third-party providers offer access to these models, some organizations are hesitant to send their private data outside their private networks due to privacy concerns [3].

Very large LLMs, like OpenAI's ChatGPT, excel as generalists across various tasks [4]. However, many applications require only domain-specific capabilities. In such cases, smaller specialist models offer a more resource-efficient alternative. For instance, BioBERT, pre-trained on biomedical text, is better suited for biomedical tasks [5]. Specialist models are both practical for resource-constrained organizations and better aligned with specific domain needs.

Knowledge distillation has emerged as an effective technique for creating these specialist models, as demonstrated by methods like BioBERT and the 'Distilling Step by Step' approach [5, 6]. However, based on our observation existing distillation methods may not fully leverage the teacher model's knowledge and abilities, especially in context extracting the most relevant information that addresses the student model's weaknesses. Consequently, there is room for improved knowledge elicitation techniques that better extract and transfer pertinent knowledge to the student model.

Our research seeks to address this shortage by introducing a novel feedback-driven framework that dynamically augments tailored data to the "transfer set", a term introduced in the works of Hinton et al. (2015) [7], which refers to the dataset used in the knowledge distillation process to elicit knowledge from the teacher model. For clarity in our study, we will refer to the "transfer set" as the "unlabeled dataset" when used to elicit knowledge from the teacher model and the "training dataset" when the teacher has labeled it.

Traditional methods rely on predefined, static transfer sets that cannot adapt to the evolving deficiencies of student models. In contrast, our approach incorporates student feedback to iteratively expand the transfer set, allowing the teacher model to generate data tailored to specific weaknesses in the student model.

We present a case study involving the training of a small specialist model (Gemma 2b) for Python-related question-answering tasks, aimed at assisting students in a classroom setting. Using ChatGPT-4o-mini as the teacher model, our methods enable the development of a resource-efficient, locally deployable specialist model. To achieve this, we propose two methods of adaptive knowledge elicitation: (1) Feedback-Driven Question Generation and (2) Targeted Prompt Question Generation. Both methods iteratively augment the transfer set by leveraging student errors and guiding data generation through the teacher model.

Our findings demonstrate that these iterative approaches improve model accuracy and response quality compared to static methods and enhance knowledge relevance without overfitting.

II. Literature Review

A. Introduction to Knowledge Distillation

Knowledge distillation (KD) is a machine learning technique where a smaller model (the student) learns from a larger model (the teacher) [8, 9]. As we mentioned in the introduction this approach is widely recognized as an effective method for fine-tuning smaller language models to become task-specific specialists [5, 6]. By transferring the teacher model's knowledge to a smaller student model, KD essentially reduces computational and memory requirements, making it a practical solution for deploying language models in resource-constrained environments.

KD typically involves two stages: knowledge elicitation and knowledge transfer [10]. In the elicitation stage, the teacher model processes an "unlabeled dataset" [11], generating output labels or insights that reflect its understanding of the task [12]. In the transfer stage, the student model is trained on these outputs to approximate the teacher model's predictive abilities [12]. In our research our focus is on the stage of knowledge elicitation.

B. Existing Methods for Eliciting Knowledge from an LLM

We categorize knowledge elicitation methods into two main types: (1) Static and (2) Interactive [13], with further sub-categories based on Xu et al. (2024) [10], who provides a survey on knowledge distillation techniques.

(1) Static methods for knowledge elicitation involve generating or curating a training dataset independently of the student model's performance. These approaches leverage a teacher model's capabilities to annotate, expand, or synthesize data. Such methods include:

1. Labeling Methods: Use a teacher model to annotate unlabeled datasets, creating a training set for the student model. For example, Hsieh et al. (2023) [6] proposed Distilling Step-by-Step, which generates output labels with natural language rationales.
2. Expansion Methods: Build on labeling techniques by allowing the teacher model to iteratively generate additional data from a small seed dataset, broadening the training scope. A notable example is the Self-Instruct framework by Wang et al. (2022) [14], which creates new input-output pairs from initial instructions.
3. Data Curation Methods: Use meta-information, like predefined topics or knowledge points, to guide the teacher model in generating diverse, topic-specific instructional datasets. For instance, Ding et al. (2023) [15] introduced UltraChat, a framework where the teacher model creates topic-driven instructional data.

The general process of static knowledge elicitation can be seen in Fig. 1. Note that the methods described above; labeling, expansion, and data curation happen independently from the student model and before the transfer stage.

The methods that we propose in our research relate to these knowledge elicitation techniques of labeling, expanding, and curating data, yet with a key difference in that, we incorporate the student's feedback in expanding the unlabeled dataset which is then used to create the training dataset.

(2) Interactive methods for knowledge elicitation involve introducing a dynamic feedback loop between the teacher and student models. Unlike static methods, these approaches adapt the distillation process based on the student model's performance. Methods employing this approach include:

1. Corrective Feedback Methods: Focus on identifying student model errors and generating corrected training data. For example, Agarwal et al. (2024) [16] introduced On-Policy Distillation, where the teacher model offers token-level feedback to enhance predictions.
2. Learning-to-Teach Frameworks: These methods allow the teacher model to adapt its outputs dynamically based on student feedback. Liu et al. (2024) [13] proposed a framework where the teacher refines its strategy by treating student feedback as performance loss, generating more tailored soft targets.
3. Ranking-Based Methods: These methods use the teacher model to rank student responses, providing reward signals for reinforcement learning. For example, Luo et al. (2024) [17] proposed RLEIF, where ranked feedback trains the student model via Proximal Policy Optimization (PPO) to improve reasoning.

The interactive knowledge elicitation process can be seen in Fig. 2. and the focus is on optimizing labeling using student feedback. While these methods enhance distillation with feedback, they generally rely on a static unlabeled dataset determined before training and emphasize improving the labeling in the transfer stage rather than new examples.

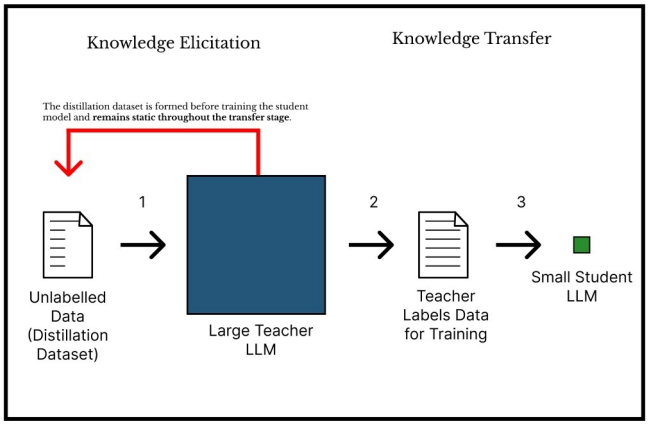


Fig. 1. A diagram illustrating the general process of static knowledge elicitation within knowledge distillation.

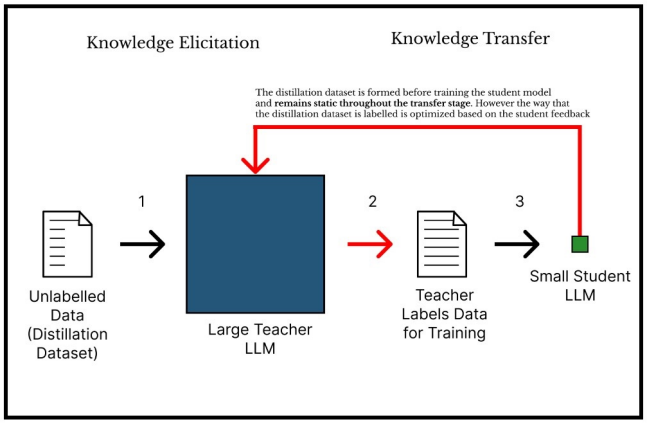


Fig. 2. A diagram illustrating the general process of interactive knowledge elicitation within knowledge distillation.

C. What We Propose

We propose two novel methods, collectively termed "Dynamic knowledge elicitation," for dynamically expanding the unlabeled dataset based on student model performance:

- **Method 1**: Feedback-Driven Question Generation
- **Method 2**: Targeted Prompt Question Generation

Here, "questions" refer to unlabeled data, which the teacher model labels with "answers" to form a complete training dataset. This dataset is iteratively expanded through cycles of testing, augmentation, and retraining, ensuring it addresses the student model's weaknesses as training progresses. The process repeats until no further improvements are observed. The general idea for both methods 1 and 2 is outlined in Fig. 3. as a comparison to what is illustrated in Fig. 1. and Fig. 2.

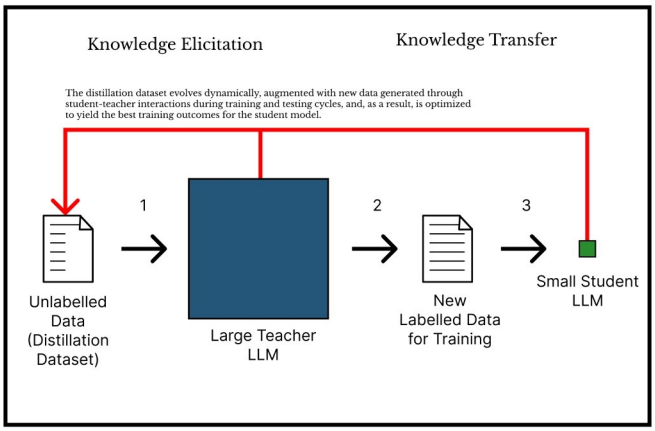


Fig. 3. A diagram illustrating the general Dynamic knowledge-elicitation process (Methods 1 & 2).

D. Other Related Studies

Our approach to adaptive knowledge elicitation also relates to the following studies, such as how datasets are selected or augmented and also reinforcement learning techniques yet with key differences:

1. Label Revision and Data Selection Techniques These methods focus on selecting optimal training data from an existing dataset [18][19]. In contrast, our approach dynamically generates an optimal dataset from the teacher model tailored to the student model's weaknesses.
2. Data Augmentation: While traditional augmentation techniques create generic variations of existing data points to improve model generalization [20], our method generates targeted data to address specific deficiencies in the student model resulting in a more optimally aligned augmented dataset.
3. Data Distillation: Conventional data distillation techniques iteratively refine a dataset to retain key statistical characteristics [21]. Our approach extends this idea to distilling knowledge from language models, by dynamically eliciting data from the teacher that yields the best results when used to train the student model.
4. Reinforcement Learning (RL): While both RL and our method utilize feedback, our approach focuses on adaptively augmenting the unlabeled dataset used for distillation to improve knowledge transfer efficiency, rather than directly optimizing the student policy via some learned reward function.

III. Methodology Of Our Novel Approach to Knowledge Elicitation

A. Feedback-Driven Question Generation (Method 1)

The goal of Method 1 is to elicit a tailored dataset from the teacher model that specifically addresses the student model's weaknesses within a specific domain of knowledge. This interactive method leverages the teacher model's in-context learning ability to identify errors in the student model's responses and iteratively generate new questions and answers addressing these errors, thus expanding the training dataset in such a way that it is tailored to improve the student's performance in areas where it previously made errors.

Implementation Steps for Method 1:

1. Initial Training: Train the student model on a 500-pair base dataset (see Section IV, Table I).
2. Testing and Error Identification: Use the teacher model to evaluate the student on 1,500 feedback QA pairs (batched at 125), identifying errors. The student is never trained on this dataset.
3. Question Generation: For each error, prompt the teacher model to generate 10 targeted follow-up questions.
4. Answer Generation: Pass generated questions to the teacher model for labeling.
5. Dataset Augmentation: Add new QA pairs to the training set, removing duplicates
6. Retraining: Retrain the student from scratch on the augmented dataset to align with the updated data distribution.
7. Iterative Data Generation: Repeat steps 2–6 until no new questions are generated or the error rate falls below 25% (see Fig. 4 and Algorithm 1).

B. Targeted Prompt Question Generation (Method 2)

Building on Method 1, Method 2 introduces specialized prompt configurations, as distinct instantiations of the teacher model, designed to generate targeted follow-up questions based on six evaluation criteria: accuracy, clarity, relevance, completeness, verbosity, and logical consistency. These prompts guide the teacher in addressing specific weaknesses observed in the student model's responses from the perspective of that criterion, resulting in a richer, more focused training dataset. The criteria were selected based on common errors identified during Method 1 testing (see Section IV for prompt examples).

Implementation Steps for Method 2

Steps 1 – 2: remain the same as in method 1.

Step 3: Collaborative Question Generation: For each identified error, prompt the teacher model to generate follow-up questions

- Teacher Model Generation: The teacher model as in Method 1, generates 10 general questions.
- Targeted Question Generation: Unique to Method 2. Six specialized prompt configurations are given to the teacher model, to generate 6 additional questions.

Steps 4 – 7: remain the same as Method 1. See Fig.4.

C. Core Contributions of Each Method

Method 1: This method uses student model errors to drive iterative data generation. For every mistake the student makes, the teacher generates follow-up questions and answers to directly target those gaps. The core contribution is its dynamic feedback loop that tailors and augments training data to address actual student model weaknesses.

Method 2: This method builds on method 1 but uses specialized prompts to generate follow-up questions focused on specific evaluation criteria. Its core contribution lies in enabling controlled data elicitation, where prompt design governs the nature and focus of knowledge extracted from the teacher model, ultimately resulting in an enriched, targeted augmented dataset for training the student model.

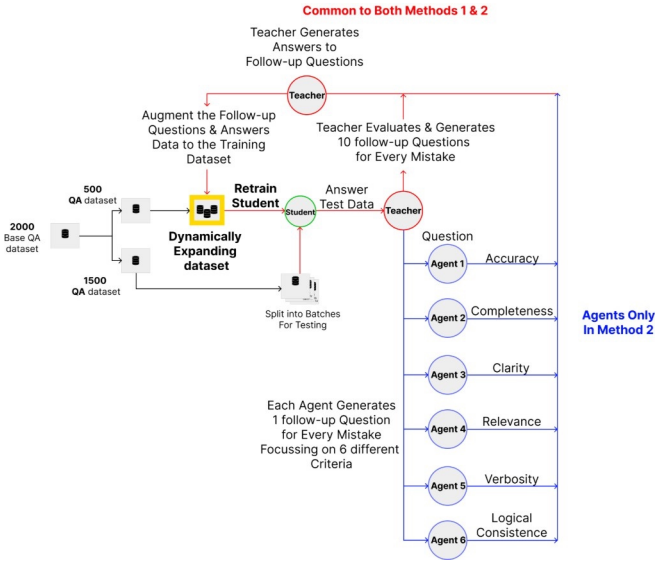


Fig. 4. A flow diagram illustrating the process of method 1 (in red) & method 2 (blue and red) with special prompt configurations (agents).

Algorithm1: The algorithm for implementing the method of targeted prompt question generation.

Require: Teacher model T, Initial dataset D_{base} (500 QA pairs), Feedback dataset D_{feedback} (1500 QA pairs), Six specialized agents $A = \{A_1, A_2, \ldots, A_6\}$ targeting evaluation criteria (accuracy, clarity, relevance, completeness, verbosity, logical consistency), Error threshold ϵ (e.g., 25%)
Ensure: Final trained student model S, Optimized training dataset D_{training}
Initialize:
Initialize student model S.
Initialize training dataset $D_{\text{training}} \leftarrow D_{\text{base}}$.
Train S on D_{training}.
repeat
  Test S **on** D_{feedback}**:**
  if Batch testing **then**
    Evaluate in batches of 125.
  else
    Evaluate on the entire dataset.
  end if
  Record errors E.
  for each error $e \in E$ **do**
    Generate follow-up questions:
    Teacher Model Generation: T generates 10 follow-up questions targeting e.
    for each agent $A_i \in A$ **do**
      A_i evaluates the student model's response against its criterion.
      A_i generates one targeted follow-up question based on its evaluation.
    end for
  end for
  Answer Generation:
  Pass all generated questions (teacher- and agent-generated) to T for labeling.
  Augment Training Dataset:
  Update $D_{\text{training}} \leftarrow D_{\text{training}} \cup$ (new QA pairs – duplicates).
  Retrain:
  Retrain S from scratch on updated D_{training}.
until No new questions are generated **or** error rate $< \epsilon$
return Final trained student model S, Optimized training dataset D_{training}.

IV. Experimental Setup

A. Data Curation and Generation for Methods 1 & 2

We sourced our primary data from the official Python documentation and supplementary textual resources [22]. Utilizing Retrieval-Augmented Generation (RAG), we generated embeddings to traverse and index the text corpus. Relevant sections were identified through an analysis of chapter headings and subheadings. For each identified section, the teacher model produced 15 unique question-answer (QA) pairs and removed any duplicates. This process yielded a total of 2,200 QA pairs, which were categorized into seed, feedback, validation, and test datasets. This data source also served as the scope of the knowledge we want to transfer in the specific domain of Python programming concepts.

Furthermore, as part of the knowledge distillation process, each method resulted in a unique dataset generated throughout the training and testing cycles. In Method 1 the feedback dataset was used to test the student model. The teacher model then generated 10 targeted follow-up questions for each error identified per question. Over 5 iterations of the entire feedback dataset, approximately 20000 unique data items were generated. In Method 2 a similar approach to method 1 but in addition, six specialized prompt configurations were used, each specializing in specific evaluation criteria, producing one additional targeted question per criterion for each error. Over 3 iterations covering the entire feedback dataset, approximately 20000 unique data items were generated in method 2.

A supplementary dataset of Python questions was also incorporated, sourced from Stack Overflow (available on Kaggle titled "Python Questions from Stack Overflow"). A random subset of approximately 20000 questions was selected from a pool exceeding 600,000 items. The teacher model was employed to clean and label this data, which was then used to train another base model for comparison with our proposed methods. See Table I. for a summary of data sources and partitions.

TABLE I. Data Sources and Partitions

Data Source	Datasets Used for Training and Testing		
	Partition	**Number of QA Pairs**	**Purpose**
Python Documentation	Seed Data	500	Initial training data for both methods.
	Feedback Data	1500	To test the student model after each training cycle to identify errors and generate new training data.
	Validation Data	100	To calculate validation loss during training.
	Test Data	100	For evaluating performance after training.
Stack Overflow	Method 0 Training Data	19000	To train a base model for comparison with our proposed methods.
	Test Data	100	For evaluating performance after training.
Generated Through Our New Methods	Training Data	≈20000 (For each method)	To train the models of our new methods.

B. Models and Environment Setup for Both Methods 1 & 2

The student model utilized in this study for both methods was Gemma 2b, an open-source model offered by Google consisting of 2 billion parameters. The teacher model was ChatGPT-4o Mini, by OpenAI and accessed via their API. The Teacher model was configured with a temperature of 0.5, a maximum token limit of 1,024, and a top-p value of 1. For comparative analysis, we used several variations; as a baseline we used an untrained version of Gemma 2b. We also used other fine-tuned variants, Gemma 2b_it, 7b_it, and 9b_it. And as an upper bound for evaluation, we used ChatGPT-4o Mini.

All training and inference for the student model was conducted on a single V100-SXM2-32GB GPU. The training was performed using Causal Language Modeling with Low-Rank Adaptation (LoRA) and mixed precision to optimize performance. The batch size was set to 1 with gradient accumulation over 18 steps. A learning rate of 1e-3 was applied using a linear scheduler returning to zero. We utilized PyTorch and Hugging Face Transformers (SFTTrainer) frameworks to facilitate the training process.

In our study, we employed specific prompting strategies to guide the teacher model in generating effective follow-up questions. These prompts were written to ensure that the generated questions would directly target the weaknesses identified in the student model's responses and adhere to a specific format so that we could extract the individual questions.

Below is an example of a follow-up question generated by the teacher model in response to the same student error. The first follows a generic teacher prompt used in both Methods 1 and 2, while the second uses a specialized accuracy-focused prompt unique to Method 2.

- Original Q&A
  - Q: "*What is the next-to-last scope in the order of searching?*"
  - A: "*The next-to-last scope in the order of searching is the local scope*"
- Generic follow up question:

  "*What are the different scopes available in Python and in what order are they searched during variable resolution?*"
- Accuracy follow up question:

  "*In Python's variable scope resolution, what is the correct order of scope searching, and which scope is the next-to-last in that order?*"

In this example, the generic version of the teacher model generates a more general question that prompts the student to review all scopes and their search order, thereby addressing the gap in the student's understanding. The response from prompt configuration for accuracy, on the other hand, generates a more targeted question that directly challenges the student to correct their misconception about the scope order, focusing specifically on the accuracy of the answer.

By using these tailored prompts in Method 2, we ensured that the generated follow-up questions varied in complexity and focus, effectively addressing different aspects of the student model's weaknesses.

C. Method 0 for Comparison

For an additional comparison, which we refer to as Method 0, we employed a standard KD labeling approach. In this method, the Stack Overflow dataset comprising 19,000 questions was labeled by the teacher model. The base model (Gemma 2b) was then trained using the same environment configuration as in Method 1 and Method 2 on this labeled dataset.

In summary, the methods compared in this study are:

- Method 0: Standard KD labeling approach for baseline comparison.
- Method 1: Feedback-Driven Question Generation.
- Method 2: Targeted Prompt Question Generation.

D. Evaluation with the Teacher Model

Our evaluation framework includes a scoring mechanism that prompts the teacher model to evaluate the student model's responses on a testing dataset, comprising the Python Documentation Test Data 100 and Stack Overflow Test Data 100 (see Table I above). The final score, expressed as a percentage, provides a measurable indication of how well the model can answer questions within the given domain. The inclusion of real-world Python questions from Stack Overflow in the testing dataset offers valuable insight into the model's practical performance in real-world scenarios.

The student model was evaluated using six key criteria, each targeting a specific dimension of performance to ensure a comprehensive assessment. These criteria were selected based on common errors observed during early training in Method 1: accuracy (factual correctness), completeness (coverage of all relevant aspects), clarity (readability and understandability), relevance (alignment with the question), verbosity (lack of redundancy), and logical consistency (coherent flow without contradictions).

For each criterion, a scoring scale from 0 to 5 was used to assess each criterion, where 0 indicates a completely inadequate rating and 5 indicates an excellent rating. To derive a comprehensive evaluation for each response, scores across all criteria were aggregated and averaged to calculate a final evaluation score.

In this framework, the average score across all criteria, and the accuracy score by itself, provided the most reliable measure of how well a question was answered. Consequently, Graph IV in Section V. was established as a reference for evaluation. The graph's line represents accuracy and the overall average response quality based on the specified criteria.

E. Other Metrics For Model Evaluation

We measured training and validation loss to evaluate the student model's learning progress and generalization ability. A gradual decrease in both indicates effective learning; while fluctuating or non-converging loss suggests ineffective learning. Diverging losses signal overfitting, where the model performs well on training data but poorly on unseen data.

Error rate reduction was also monitored throughout the testing stages, providing a clear indication of the student model's ability to effectively learn domain knowledge.

F. Justification for Design Choices

During training, we intentionally avoided relying on external ground truth data, as our primary objective is to maximize the student model's alignment with the teacher

model. In practical applications, this could be extended by integrating methods like RAG to cross-reference outputs with external data sources for enhanced accuracy. Furthermore, in our case, ChatGPT 4o Mini demonstrated sufficient accuracy in the domain knowledge, making ground truth values unnecessary.

During training we also retrained the student model from its initial untrained state after each iteration to ensure it learns across the entire data distribution of the targeted domain, avoiding catastrophic forgetting of previously acquired knowledge. This choice is especially important in our iterative framework, where new training data is generated based on cumulative student errors. By restarting from scratch, we ensure the model internalizes the updated dataset holistically, rather than simply adapting to recent additions. This avoids overfitting to new data and helps maintain consistency with earlier knowledge, ultimately supporting more effective generalization and guidance for subsequent data generation.

V. Results

Here we present an analysis of the results from Methods 1 and 2, comparing them against the baseline Method 0 and other fine-tuned models. Performance is evaluated using the teacher model's scoring framework (Section IV), alongside additional metrics such as training loss, validation loss, and error rate across training steps. We also include real examples of student model responses to illustrate the practical improvements achieved. The findings show that both Methods 1 and 2 substantially improve student performance over the baseline and outperform existing models of comparable size.

A. Teacher Evaluation Scores for Models Trained Using Methods 0, 1, And 2

Table II presents the accuracy and average scores for models trained using Methods 0, 1, and 2. Both Method 1 (18,000 data items) and Method 2 (16,000 data items) outperform Method 0, which used 19,000 data items, demonstrating that our adaptive methods achieve better results with less data. This improvement is attributed to the iterative tailoring of the dataset based on student model weaknesses—a feature absent in Method 0. Furthermore, at around 22,000 training steps, Method 2 surpasses Method 1 by 0.9% after only three feedback cycles, compared to five in Method 1. This suggests that the specialized prompt configurations used in Method 2 result in more effective knowledge elicitation and transfer.

TABLE II. Teacher Evaluation

Method	Teacher Evaluation Scores for Methods 0, 1, and 2 Across Different Training Steps			
	No. of Training Data Items	No. of Testing Cycles	Accuracy Score	Average Score
Method 2	**22168**	**3**	**81.5**	**86.1**
Method 2	16486	2	80.6	85.2
Method 1	**21919**	**5**	**80.2**	**85.2**
Method 1	18281	4	81.9	86.5
Method 0	**19000**	**N/A**	**74.7**	**81.0**

B. Examples Of Actual Student Model Responses

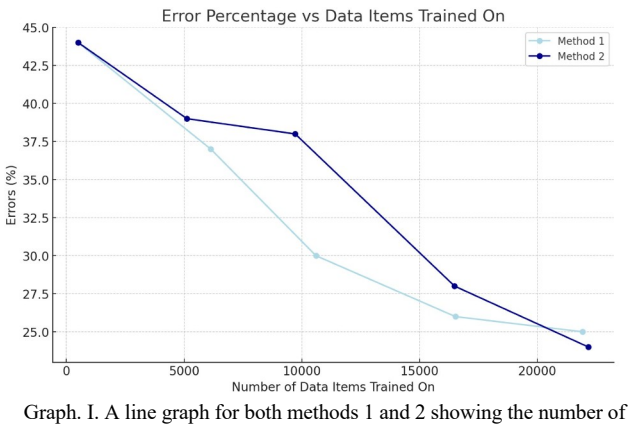
Table III. shows the student model's responses of Method 2 at three training stages 3,877, 16,486, and 22,168 data items. Initially, at 3,877 data items, the response was clear but factually incorrect, scoring 50%. By 16,486 data items, the model produced a correct answer, though it lacked completeness, resulting in an 83% score. At 22,168 data items, the model improved completeness, achieving a final score of 96%, after which training was stopped.

TABLE III. Student Model Responses at Different Stages of Training

Data Items Trained on	An Example of the Evolution of Student Model Responses Over Training Cycles for Method 2	
	Question and Student Model Answer	Avg Score
3877	Q: Can comments appear within a string literal in Python? A: Yes, comments can appear within a string literal in Python. They are denoted by a hash symbol (#) at the beginning of the line. For example: ```python my_string = "This is a string with a comment: # This is a comment" ```	50%
16486	Q: Can comments appear within a string literal in Python? A: No, comments cannot appear within a string literal in Python. Any characters following a `#` in the same line are treated as a comment and will not be included in the string.	83%
22168	Q: Can comments appear within a string literal in Python? A: No, comments cannot appear within a string literal in Python. If you try to include a comment character within a string, it will be treated as part of the string itself. For example: ```python my_string = "This is a string with a # comment" ``` In this case, `#` is part of the string, and the comment will not be recognized until outside the string.	96%

C. Error Reduction Rate

Graph I. illustrates the error rate throughout training for both Method 1 and Method 2, showing a consistent reduction as training progresses. By 22,000 data items, Method 2 achieved a 24% error rate, slightly lower than Method 1's 25%, indicating better performance. The gap in error rates between 500 and 15,000 data items can be attributed to Method 1 being exposed to the feedback dataset more frequently. However, by 20,000 data items, Method 2 had narrowed the gap and ultimately surpassed Method 1, suggesting that Method 2's approach enables more effective learning over time.

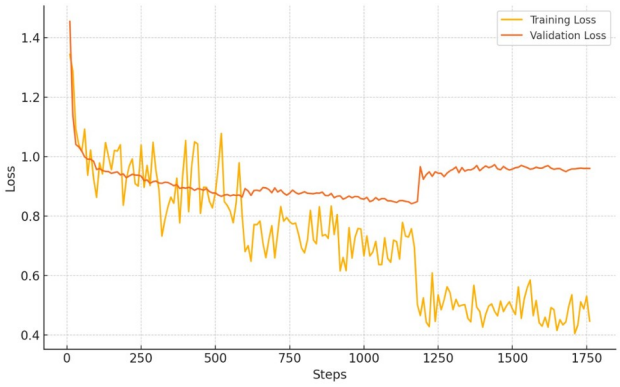


Graph. I. A line graph for both methods 1 and 2 showing the number of errors as a percentage tested on the validation set versus the number of data items trained on.

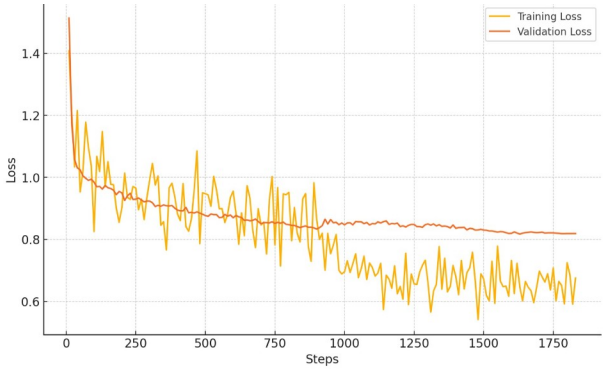
D. Training And Validation Loss

The training and validation loss trends for Methods 1 and 2 were similar; thus, only Method 1's results are shown. In Graph II, the model was trained for three epochs during an earlier feedback generation stage, while Graph III shows results from training for two epochs during a later stage, while using the same number of data items equivalent to two epochs of the earlier stage.

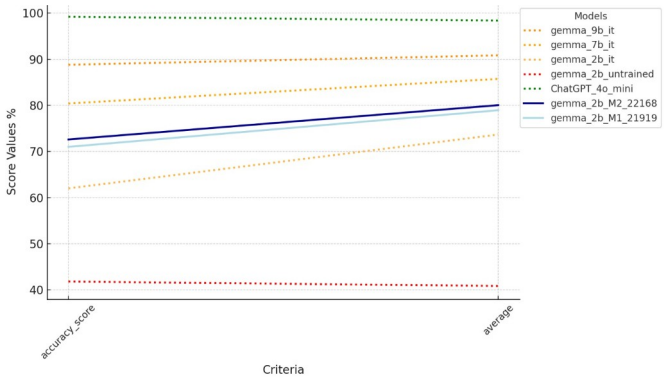
These results indicate that our adaptive knowledge elicitation approach effectively augments the dataset, supporting continued learning while preserving generalization. Training for three epochs led to overfitting, as seen in diverging training and validation loss. In contrast, two-epoch training on the augmented dataset produced converging loss curves, demonstrating the method's ability to improve domain-specific learning without overfitting.



Graph. II. A line graph showing training and validation loss across training steps for 3 Epochs (suggesting over-fitting when just increasing epochs).



Graph. III. A line graph showing training and validation loss across training steps for 2 Epochs plus augmented data. (Showing good generalization by Data Augmentation).



Graph. IV. Line graph showing accuracy and average evaluation scores used to compare overall model performance.

E. Comparing the results of Evaluating Different Models on the Testing Data.

Graph IV compares the performance of various fine-tuned models using the teacher model's evaluation framework. As expected, the teacher model achieves the highest scores (green line), while the untrained model performs lowest (red line). Yellow lines represent instruction-tuned Gemma models of varying sizes, serving as benchmarks for student model performance. Our proposed Methods 1 and 2 are shown in light blue and dark blue, respectively, and both demonstrate significant improvements over the baseline. Notably, Method 2 outperforms the instruction-tuned version of the same-sized model (2B) by 10%, and is only 10% behind the next larger model (7B). These results highlight the effectiveness of targeted knowledge elicitation in achieving competitive performance with smaller, more resource-efficient models. Additional training cycles may yield further improvements.

VI. Addressing Limitations and Future Directions

A. Addressing Limitations

While our approach demonstrates clear gains, it also presents several limitations.

First, evaluation relies solely on teacher-model scores, which despite being consistent may overlook nuances captured through human judgment. Incorporating human evaluations, such as small-scale classroom deployments, will be essential to verify whether observed improvements translate into real learning outcomes.

Second, our comparisons are limited to internal baselines. Without bench-marking against established distillation frameworks (e.g., DistilBERT, TinyBERT, Self-Instruct), it remains unclear how our methods compare in terms of efficiency, accuracy, and resource demands. External validation will help contextualize our contributions within the broader knowledge distillation landscape.

Finally, retraining from scratch in each iteration improves generalization and prevents catastrophic forgetting, but it comes at a computational cost. While feasible for a 2B-parameter model on a 32 GB GPU, this approach may not scale well to larger architectures or more constrained environments. Future work could explore hybrid strategies—such as mixing cold restarts with continued fine-tuning or selectively freezing layers—to reduce cost while preserving effectiveness.

B. Future Research

Building on our results we recommend the following directions for future research:

1. Ensemble of Teachers: Augment the single-teacher setup with diverse model architectures to increase question variety and reduce biases in the models.
2. Ablation Studies on Prompt Configurations: Systematically disable or modify individual prompt strategies in Method 2 to assess their unique impact. This can guide the design of more efficient, targeted prompting techniques for knowledge elicitation.
3. Domain Generalization: Extend the framework to other domains (e.g., medical diagnosis, legal question answering) by adapting feedback datasets and prompt strategies to the specific domain.
4. Scaling Up Student Models: Apply both methods to larger student models (7B–9B parameters) to assess

performance scalability and computational feasibility.

VII. Conclusion

Our research introduces and validates two novel adaptive knowledge elicitation methods— (1) Feedback-Driven Question Generation (Method 1) and (2) Targeted Prompt Question Generation (Method 2)—for creating task-specific specialist models through knowledge distillation. By incorporating student feedback into the data generation process, both methods significantly improved model accuracy, response quality, and generalization to real-world questions compared to traditional static approaches.

Our experiments demonstrate that these methods serve as effective dynamic data augmentation techniques, iteratively tailoring the transfer dataset to address the student model's weaknesses. This approach improves the relevance of the elicited knowledge without overfitting. Additionally, Method 2, which employs specialized prompt configurations, proved particularly effective in later training stages, achieving better results than Method 1 with fewer iterations over the testing dataset. The resulting optimized dataset, designed to target specific deficiencies, also provides a valuable synthetic data resource for training similar models.

In future work, one could investigate an ensemble of teacher architectures leaning more toward an agent-based approach and perform ablation studies to quantify each prompt strategy's contribution. This framework could also be applied to domains beyond Python QA such as medical and legal domains. Lastly, one could explore hybrid retraining schedules to further optimize the trade-off between computational cost and model performance.

VIII. References

[1] M. A. K. Raiaan et al., "A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges," in IEEE Access, vol. 12, pp. 26839-26874, 2024, doi: 10.1109/ACCESS.2024.3365742.

[2] Stanford Institute for Human-Centered Artificial Intelligence (HAI). (2024). AI Index Report 2024 (p. 63). Stanford University. Retrieved from https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_AI-Index-Report-2024.pdf

[3] Stanford Institute for Human-Centered Artificial Intelligence (HAI). (2024). AI Index Report 2024 (p. 174). Stanford University. Retrieved from https://aiindex.stanford.edu/wp-content/uploads/2024/04/HAI_AI-Index-Report-2024.pdf

[4] OpenAI. (2023). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.

[5] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining. Bioinformatics, 36(4), 1234–1240.

[6] Hsieh, C.-Y., Li, C.-L., Yeh, C.-K., Nakhost, H., Fujii, Y., Ratner, A., Krishna, R., Lee, C.-Y., & Pfister, T. (2023). Distilling step-by-step! Outperforming larger language models with less training data and smaller model sizes. In Findings of the Association for Computational Linguistics: ACL 2023 (pp. 8591-8607). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-acl.507

[7] Peris, C., Tan, L., Gueudré, T., Gojayev, T., Wei, P., & Oz, G. (2022). Knowledge distillation transfer sets and their impact on downstream NLU tasks. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track (pp. 128–137). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.emnlp-industry.12

[8] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. NIPS Deep Learning and Representation Learning Workshop. Retrieved from https://arxiv.org/abs/1503.02531

[9] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). FitNets: Hints for thin deep nets. International Conference on Learning Representations (ICLR). Retrieved from https://arxiv.org/abs/1412.6550

[10] Xu, X., Li, M., Tao, C., Shen, T., Cheng, R., Li, J., ... & Zhou, T. (2024). A survey on knowledge distillation of large language models. arXiv preprint arXiv:2402.13116.

[11] Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '06), 535–541.

[12] Hinton, G. (2015). Distilling the Knowledge in a Neural Network. arXiv preprint arXiv:1503.02531.

[13] Liu, Y., Sun, T., Qiu, X., & Huang, X. (2021). Learning to teach with student feedback. arXiv preprint arXiv:2109.04641.

[14] Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2022). Self-instruct: Aligning language models with self-generated instructions. arXiv preprint arXiv:2212.10560.

[15] Ding, N., Chen, Y., Xu, B., Qin, Y., Zheng, Z., Hu, S., ... & Zhou, B. (2023). Enhancing chat language models by scaling high-quality instructional conversations. arXiv preprint arXiv:2305.14233.

[16] Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Garea, S. R., Geist, M., & Bachem, O. (2024). On-policy distillation of language models: Learning from self-generated mistakes. In The Twelfth International Conference on Learning Representations.

[17] Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., ... & Zhang, D. (2023). Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. arXiv preprint arXiv:2308.09583.

[18] Li, L., Lin, Y., Ren, S., Li, P., Zhou, J., & Sun, X. (2021). Dynamic knowledge distillation for pre-trained language models. arXiv preprint arXiv:2109.11295.

[19] Lan, W., Cheung, Y. M., Xu, Q., Liu, B., Hu, Z., Li, M., & Chen, Z. (2024). Improve Knowledge Distillation via Label Revision and Data Selection. arXiv preprint arXiv:2404.03693.

[20] Wada, S., & Morimoto, N. (2024). Investigating relationship between data augmentation intensity and model performance in natural language processing. 2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), 445–446. https://doi.org/10.1109/ICCE-Taiwan62264.2024.10674562

[21] Lei, S., & Tao, D. (2023). A comprehensive survey of dataset distillation. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[22] Halvorsen, H.-P. (2020). Python Programming. Hans-Petter Halvorsen. ISBN: 978-82-691106-4-7. Available at https://www.halvorsen.blog.