Analyzing seismic workflows with large language models: a case study on error detection

Sujitha Martin[*†], Manoj Alwani[†], Yuan Tian[†], Jared Kramer, and Arun Ramanathan, Amazon Web Services; Ben Lasscock and Alejandro Valenciano, TGS*

SUMMARY

Raw seismic data consists of complex signals captured by specialized instruments that geophysicists must process through carefully sequenced filters to extract meaningful geological information. When designing these processing workflows, experts frequently encounter technical challenges ranging from incorrect configurations to problematic information flow patterns. Traditionally, these errors remain undetected until workflow execution, resulting in significant resource waste and project delays. Our research explored how large language models (LLMs) can process graph-based workflow representations to predict execution failures before implementation. We developed two complementary approaches: an embedding-based supervised contrastive learning method for classifying workflows, and a fine-tuned LLM approach for generating error explanations. Our findings indicate that the embedding-based approach achieves 59% precision across a diverse set of error types, while the fine-tuned LLM, though comparable to pretrained models in failure identification (52.87% vs. 49.32% precision), significantly outperforms in explanation capabilities—achieving 32.61% explanation accuracy compared to 0% for the base model. These results demonstrate the potential for AI-assisted workflow validation to reduce computational costs and accelerate geophysical data processing in production environments.

INTRODUCTION

Large Language Models (LLMs) have significantly advanced natural language understanding and generation, becoming pivotal across multiple scientific and computational disciplines. Their effectiveness in embedding complex textual and structured data into semantically meaningful representations and their generative capability for coherent explanatory text have expanded their application scope considerably. Despite their broad success, adapting LLMs to the domain of seismic data processing poses distinct challenges due to the highly specialized nature and complexity of the geophysical workflows involved.

In seismic data processing, workflows typically consist of directed computational graphs with distinct nodes representing individual geophysical modules interconnected by precise data flows. These workflows exhibit complexity due to their structural and parametric variability and involve a substantial amount of domain-specific terminology and nuanced technical specifications rarely encountered in general-purpose model training datasets. Additionally, the error landscape in seismic processing workflows exhibits a pronounced long-tail distribution, characterized by a few commonly recurring issues accompanied by numerous infrequent yet critical error scenarios. Consequently, accurately predicting execution failures before runtime poses substantial challenges to conventional modeling approaches.

Our study introduces two complementary approaches tailored for seismic workflow validation to address these specialized requirements. Firstly, we employ supervised contrastive learning, leveraging embedding techniques to capture nuanced distinctions between successful and unsuccessful seismic processing workflows. This method effectively addresses the severe class imbalance and subtle complexity inherent in seismic workflow datasets. Secondly, we explore fine-tuning methodologies tailored to adapt LLMs to the geophysical domain, enhancing their capacity to generate precise, actionable explanations for potential workflow execution failures. We also highlight critical data curation practices, focusing on boundary-case scenarios that substantially improve predictive performance and interpretative accuracy. While the classification and explanation accuracies achieved indicate meaningful progress, they remain insufficiently robust for immediate deployment into production environments.

APPROACHES

Approach 1: Supervised contrastive learning

Each seismic workflow incorporates multiple information sources: workflow graph structure, associated metadata (execution parameters, environmental variables), and interpretive instructions. These graphs can be large and can have lot of unnecessary information. Picking signals from these graphs to predict workflow is going to successful or failure is similar to find needle in haystack. In addition, real world datasets exhibit natural class imbalance, with successful workflows substantially outnumbering failed ones. To address these limitations, we propose supervised contrastive learning which consist of following critical components:

1. Better representation using fixed embedding model
Embedding models are trained [Devlin et al. (2019); AWS (2024)] to compress input (image/text etc.) and extract only important information in form of dense vector (embedding) representation. In our case, We used Amazon Titan embedding model which takes entire workflow graph as input and generate vector that capture essential semantic and structural information. These vectors serve as the foundation for downstream processing without being further fine-tuned during training.

2. Contrastive learning to handle class imbalance
Contrastive learning is a machine learning approach to learn information from data by contrasting positive (successful workflows) and negative (failed workflows) pairs [Chopra et al. (2005)].

[†]Equal contribution

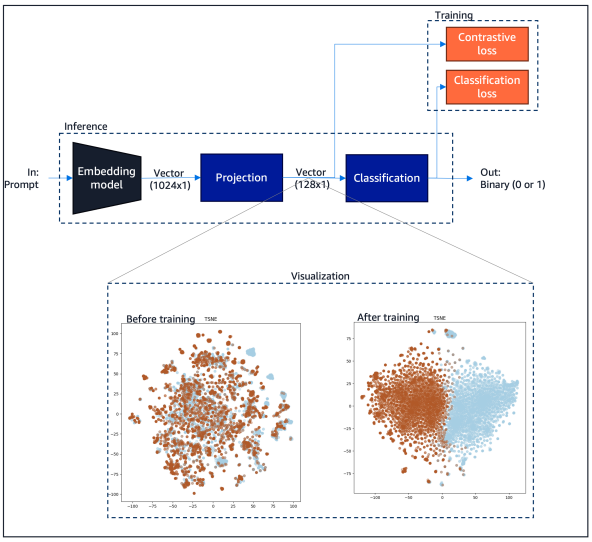

Figure 1: This figure depicts the embeddings based supervised contrastive learning approach

To implement contrastive learning mechanism for our use case, we first randomly choose embeddings of successful workflows and one unsuccessful workflow. We pass these embeddings to same multi-layer perceptron (MLP) depicted as projection layer in Figure 1. This MLP generates embedding in lower dimension which goes to contrastive loss. Contrastive loss minimize the distance of successful workflows embedding and maximize distance between successful and unsuccessful workflows embedding. This creates distinct clusters in the embedding space that correspond to success and failure cases as shown in Figure 1.

Contrastive learning mechanism creates natural augmentation of the dataset by comparing every positive workflow with all the positive and negative workflows. For example, in our dataset if we have N successful workflow and N unsuccessful workflows, then for each successful workflow we can choose from other N-1 successful workflows as positive pair and from any N unsuccessful workflow as negative pair, creating total N-1 X N combinations for each successful workflow. This augments the dataset and addresses the issue of class imbalance.

3. Supervision using classification task
Finally, we trained a small binary classifier on top of the embeddings coming from projection (MLP) layer as shown in Figure 1. This classifier learns that given workflow is going to be successful or not. We train both MLP and classification layer together which supervise contrastive learning mechanism to extract important information from the input embeddings to MLP and create two clusters. The combination of supervised classification and contrastive learning is called supervised contrastive learning [Khosla et al. (2020)].

During inference, new workflows are processed through the fixed embedding model to generate embeddings, which are then passed through the trained MLPs and classificaton to predict execution outcomes.

Approach 2: Fine tuning LLM

Our fine-tuning approach for large language models to analyze seismic workflows relies on three critical components: targeted data curation, effective representation, and strategic prompting. Rather than random sampling, our data curation process specifically targets boundary cases—pairs of workflows where a failed execution was subsequently corrected to create a successful one. These paired examples highlight the subtle differences that determine workflow success or failure, providing the model with highly informative training signals. For representation, we adopted an existing technique to transform computational workflow graphs into natural language descriptions [Fatemi et al. (2024)]. This conversion translates the native graph structure (nodes as processing modules, edges as data flows) into descriptive text that LLMs are optimized to process, making the complex technical information more accessible to the model's pre-trained capabilities. Our prompting strategies are designed to elicit both binary success/failure predictions and explanatory reasoning about potential workflow issues, enabling the model to identify critical differences between successful and unsuccessful seismic processing workflows. An example portion of a real graph representation is shown below in Figure2.

For each workflow, the model's task was to predict if there would be a field existence error. A failed prediction would identify both the specific module (node) where the error occurs and the missing field, as in this example expected output in a structured JSON format:

```
{
    "status": "failed",
    "Module": "Node53 Radfilt",
    "explanation": "Field ABSILOFF was not available on input."
}
```

The graph representation strategy significantly reduced the token count compared to raw JSON while preserving all information necessary for error prediction. Each example in the training data included approximately 6,200 tokens for the prompt, allowing the models to process the complete graph structure

```
##Workflow Graph##
The workflow represents a DIRECTED graph among nodes Node0
      through Node57 where each node
represents a processing module. In this graph:

##NODES##
Node0 is 'kernel_loader' module with the input parameter
     values:
- TASK_1=R_LINE
- TASK_2=REC_ID
- TASK_3=SLINEBLK
...

Node57 is 'TimeSeriesMuting' module with the input
     parameter values:
- dynamic_field=trace_data
- mute_header=TOPMUTE
...

##EDGES##
- Node0 directs to Node1
- Node1 directs to Node2
...
- Node56 directs to Node57
- Node57 directs to Node51
```

Figure 2: Representation of a seismic processing workflow graph showing modules and their connections.

while optimizing for cost and latency.

RESULTS

Dataset and metrics

Our experimental evaluation utilized two distinct sampling strategies to accommodate the different requirements of our approaches. The first comprehensive sampling strategy included workflows representing the full spectrum of error types, ranging from common failures to long-tail cases that appear infrequently in production. This dataset maintained the natural distribution of error frequencies while only constraining samples based on the context length limitations of the embedding model. This approach allowed us to assess model performance across the entire error landscape, providing insights into generalizability across diverse failure modes. The second, more focused sampling strategy concentrated exclusively on the most frequent error type in our dataset. By narrowing to this specific failure mode, we could perform deeper analysis on how well models detect subtler variations within a single error category and determine whether specialized models outperform general ones for common cases.

For evaluation metrics, we measured performance using precision, recall, and F1-scores for the binary classification task of predicting workflow success or failure. For explanation generation, we primarily relied on exact string matching to quantitatively assess the model's ability to produce the correct error identification. We also conducted sanity checks to verify that models weren't simply memorizing training examples and regurgitating them during inference. These checks included examining explanations for workflows with similar but not identical error conditions and verifying that model predictions varied appropriately with meaningful input changes. This combination of automated metrics and verification tests helped confirm that the models were genuinely learning to identify workflow issues rather than just memorizing the training data.

Classification accuracy

Table 1: Dataset for classification task and imbalance between positive and negative class.

	Failures(%)	**Successful**(%)
Total Data	29725 (5.0%)	579954(95.0%)
Train Data	24188 (4.9%)	469137(50.9%)
Valid Data	2613(4.7%)	53332(95.3%)
Test Data	2924 (4.8%)	57485(95.2%)

To evaluate our models performance on classification task (success vs failures), we collected all the data which can fit in models context length. As shown in Table 1, We found more than half million of such cases with extreme data skew of only 5% belongs to successful workflow category. This data skew mimics the real world scenarios and we kept this distribution for our training and testing. We keep 80% of the data for training, 10% for validation and 10% for testing for our experiments.

Table 2: Impact on Model performance with different dataset sizes in Training.

Experiment	**Successful**	**Failures**	**Precision/Recall/F1(%)**
Balanced	5000	5000	13/63/21
Imbalanced	24188	469137	59/59/59

To understand impact of data size and imbalance, we first trained a model with balanced data of 5000 successful and failures cases. Our model was able to achieve only 21% F1 score in this case. For next experiment, we trained our models on full imbalanced training data set with almost 0.5 million samples. The performance improves significantly in this case and reach to 59% F1 score along with 59% precision and recall. This shows that large amount of data also brings diversity and model was able to learn more diverse cases of failures and success which improves the performance.

Table 3: Impact on model performance with different embeddings

Workflow	**Precision/Recall/F1(%)**
With Meta data,Instructions, prompt	59/59/59
Remove meta data	8/43/13
Remove Instructions	9/17/11

Contrastive training performance relies on the fixed embedding generated by embedding foundation model as shown in Figure 1. To investigate the effect of generated embedding on model performance, we first generated embedding providing all the information to embedding model. This information incorporates workflow graph structure, associated metadata (execution parameters, environmental variables), and instructions in the prompt. We found that providing more information model generates good embedding and reaches to F1 score of

59% while removing any details like meta data or instructions drops the performance significantly.

Explanation accuracy

We compared the fine-tuned model against the base model on a test set of 264 samples (132 failed and 132 complete workflows), focusing on their ability to provide accurate explanations for workflow failures. As shown in Table 4, the fine-tuned model showed significant improvement in providing correct explanations for workflow failures.

The results show that while pretrained and finetuned models have similar precision in identifying failures (49.32% vs. 52.87%), the fine-tuned model achieved a 32.61% explanation accuracy, correctly identifying both the failing module and the specific missing field. In contrast, the base model was unable to provide any correct explanations for the failures it identified.

Table 4: Explanation accuracy comparison

Metric	Fine-tuned	Base
Predicted failures	87	219
True failures (of predicted)	46	108
Correct explanations	15	0
Failure precision	52.87%	49.32%
Explanation accuracy	32.61%	0.00%

Error Analysis of Model Explanations

To further evaluate our fine-tuned model's explanation capabilities, we conducted a detailed analysis of the 31 cases where the model provided incorrect explanations. Our primary concern was whether the model hallucinated non-existent modules or nodes when generating explanations.

The analysis revealed that in all 31 cases of incorrect explanations, the model referenced modules that were actually present in the workflow graphs. This indicates that while the model sometimes incorrectly identified the source of errors, it consistently operated within the constraints of the actual workflow structure rather than inventing non-existent components.

We also examined whether the model was simply memorizing common error patterns. The distribution of correctly predicted modules, as seen in Table 5 across the 15 cases where the model provided accurate explanations shows diverse predictions rather than consistent repetition.

Table 5: Distribution of correctly predicted modules

Predicted Module	Count
Node18 InFlowMath	2
Node8 KillTrace	2
Node1 tesla_ensemble_redefine	2
Node26 InFlowMath	1
Node5 KillTrace	1
Node19 task_redefine	1
Node15 InFlowMath	1
Node22 ASCII_Write	1
Node1 ScalarStatics	1
Node3 InFlowMath	1
Node52 KillTrace	1
Node16 Split	1

The variety of modules in correct predictions, with most appearing only once, suggests that the model is not simply memorizing common patterns but is able to reason across different workflow structures to identify the true sources of errors.

CONCLUDING REMARKS

This study demonstrates the feasibility and potential value of integrating Large Language Models (LLMs) into seismic workflow validation processes. While the achieved classification and explanation accuracies indicate meaningful progress, they remain insufficiently robust for immediate deployment into production environments. Nevertheless, the results illustrate how advanced AI techniques can substantially enhance the preemptive identification and interpretation of workflow errors in geophysical data processing. The methodologies developed provide a foundational step towards more sophisticated, reliable AI-driven systems. Future research should focus on expanding training data with real and synthetic data generation to address sparsity issues, refinement of prompting strategies for more actionable insights, and continued domain-specific pre-training to achieve genuinely practical geophysical applications.