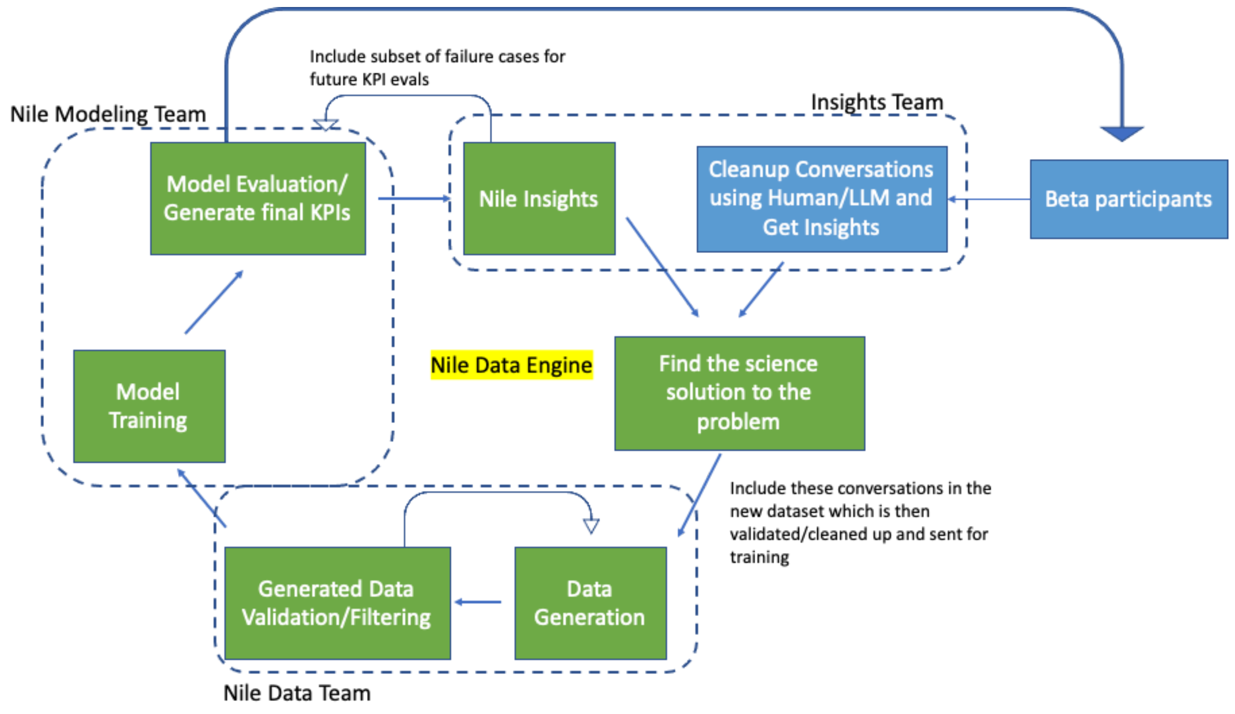
High Quality Data Generation/Validation for instruction-tuning Nile Models

Problem Statement

In this doc, we discuss how can we generate and validate high quality demonstration data at scale for instruction-tuning of Nile models. This is a living document which will change based on new information.

Nile Data Engine

In order to generate high quality data, we need to align the data team closely with Nile model training/alignment teams. This will help us create a data engine (shown below) which can ensure that high quality and relevant data is generated which will help improve Nile model KPIs. Current process includes generating large amounts of data which are then sent to Nile foundation model training without a fast feedback loop. The feedback loop has to be fast enough for us to generate relevant data. The latency from data to model to evaluation should be minimized. Eventually it can lead to active learning, where the trained model can decide which data is useful to further improve the model.


Include subset of failure cases for future KPI evals
Nile Modeling Team
Insights Team
Model Evaluation/ Generate final KPIs
Nile Insights
Cleanup Conversations using Human/LLM and Get Insights
Beta participants
Model Training
Nile Data Engine
Find the science solution to the problem
Include these conversations in the new dataset which is then validated/cleaned up and sent for training
Generated Data Validation/Filtering
Data Generation
Nile Data Team

The data engine consists of following key steps:

1. **Data Generation**: In the first step, we will generate demonstration which can be either LLM generated or Human generated. Initially, we should create demonstrations using humans on diverse range of topics which can act as seed for

future LLM generated data. We can ensure that the generated demonstration data include a few adversarial scenarios (check section 3.1 in this paper). Therefore rather than filtering the demonstrations if they do not clear the data validation stage (next section), we should divide demonstrations into various buckets based on those attributes and provide a few samples for every bucket.

2. **Data Validation:** It is important to maintain the quality and diversity of initial data, so human based validation will be the best option. However, since it is not scalable we can use LLM to validate and filter out bad demonstrations. In the end, we should use Nile model evaluation KPIs as the guidance towards whether or not use LLM for data validation and filtering. The data validation output can be fed back for improved data generation where we can use LLM for auto-correcting the demonstration (LLM-based AutoCorrections).
3. **Model Training/Validation:** Use the generated data to instruction tune Nile model and generate KPIs on the test data. This will be handled by the Nile foundation team.
4. **Nile Insights** (Analyze failure cases)**:** This will be an important step to guide new data generation. The query on which Nile model's output is worse than other LLMs (e.g. ChatGPT) will be analyze by the Insights team and the corresponding insights will passed to data generation team. There is a risk of test data leaking into the training data in this case. However, it can be mitigated by expanding test dataset (while still including old samples) with every new trained model. The golden test split initially can cover basic use cases and later expand to include more demonstrations and use cases. Additionally, failure cases should be included in future KPI evaluations for regression testing ensuring that future models are not performing worse on past data.
    a. Additionally Nile insights can provide the attributes which are important for model performance which can guide our data generation sampling strategy.
5. **Find the science solution to the problem:** Given the insights from failure cases, we have to figure out the science solution to generate better data for improving the model. One of the solution is to generate demonstrations similar to the failure cases.
    a. **Generate similar demonstrations to failure cases:** Given the failure case demonstrations, we can use LLM or humans to generate more demonstrations similar to the ones where current model fails. We can feed the generated data back into training pool after it is validated. This will ensure that model keeps improving over the failure cases.
6. **Include demonstrations from beta participants:** Since we cannot create all possible demonstrations, it will be useful to include the demonstrations from beta participants after they are sanitized.

Additional advantages of building the Nile data engine:

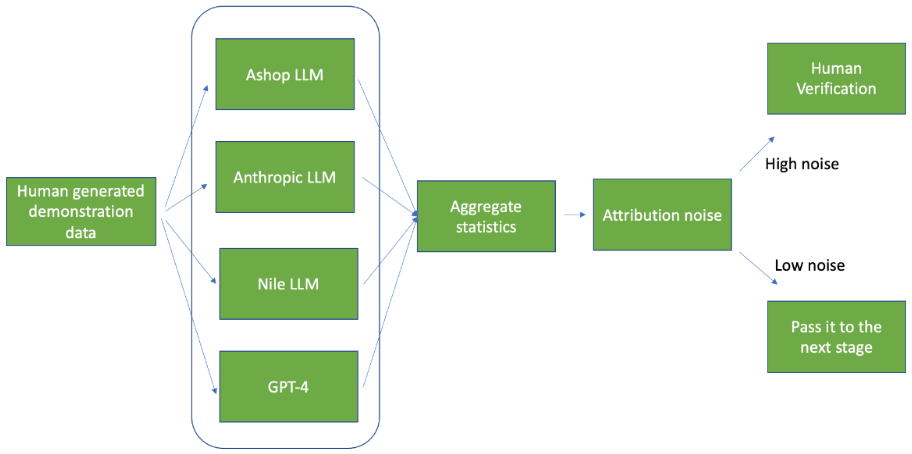
1. Since Nile model and the science team are our customer, feeding back the KPIs from science team will help us generate demonstration which are useful to improve the final KPIs (**Optimizing human capacity**)
2. We should eventually use Nile LLM to decide which demonstration data to keep which to throw out depending on how consistent the generated demonstration is with Nile LLMs output. We should avoid heuristics as much as possible when sanitizing data and let Nile LLM decide. (**Gatekeeping**).

Recommended Strategy for Model based Validation

Use current LLM models as an evaluator and get noise output (P0): As a first step, we need to evaluate current publicly available LLM model. We have discussed the evaluation of Anthropic model here. Similarly, we can other internal/external candidate models. These models should also be evaluated on public benchmarks like UniEval for a fair comparison against other evaluators. However, these models won't perform well on all attributes or demonstrations as shown in recent papers. Also, discussed in risks. There are multiple techniques we can use to improve its performance:

1. **Self-consistency sampling**: Run the same or multiple LLM multiple times and take the majority vote or weighted mean.
2. **Fine grained scoring:** Predict float score (1-5) instead of yes/no for each attribute and compute the weighted mean when using self-consistency.

3. **Get prediction confidence**: When we run same or different LLMs multiple times, we can also get the variance across predictions. The variance gives us an idea of how confident the model is about its predictions. The noise can be used to decide whether to send the prediction back to human for verification or to send it to Nile. Figure below shows the overall flow. For example: If for a demonstration data, AShop LLM predicts a score 4 where as Anthropic model predicts a score of 2, we know that the LLM evaluators are not confident about its evaluation. We can pass the input multiple through the same LLM to get prediction noise as well.
4. **Noise threshold:** Noise threshold of when to send the data to Nile vs Human verification should be decide using annotated data with human annotations for attributions. Nile science team should be involved in the discussion as well since we have to answer the question how much attribution noise should be ok to train Nile LLM models.



Demonstration for each attribute question (P1):

1. Each attribute question can be further explained using few shot demonstration which explains what we mean for each question. Current queries like “Is this answer plausible” is ambiguous unless we provide a few examples of what we mean by this question.
2. The solution would be to provide a few example of each attribute before asking yes/no question.
3. However, this will increase the overall prompt length and it requires further experimentation if we can feed a longer prompt to Anthropic LLM or not. Maybe we need to breakup the prompt into smaller prompts.

Auto-prompt engineering for model validation (P2):

1. Current LLM models requires the effort of language engineers to define the prompt for validation tasks. This is not scalable when new requirements have to be supported.
2. Recent research has shown that an LLM can search over a pool of instruction candidates which is proposed by another LLM to maximize a score function (link).
3. We can use LLM to choose the best set of attribute questions to improve the overall validation performance on our dataset. This can also be designed as a combinatorial search problem where we can send multiple combinations through LLM and check which prompt has the best performance on our dataset.

Training a truthfulness classifier using LLM activations (P3):

1. Recent research showed that the internal state of LLM can be used to reveal the truthfulness of a generated statement (link).

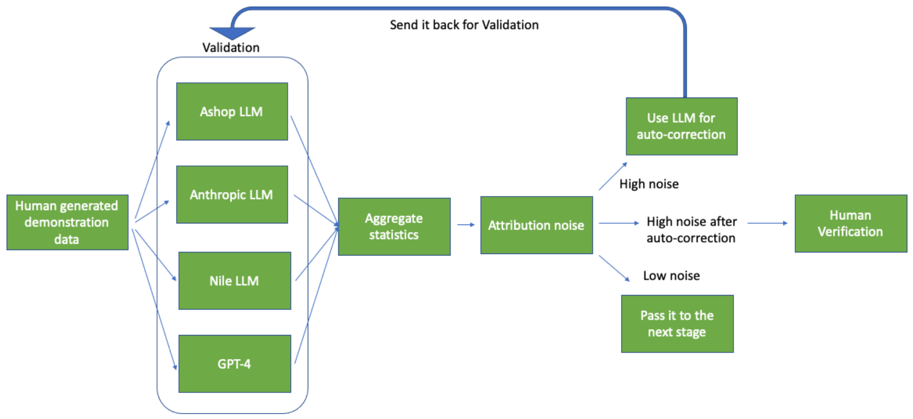
2. We can extend this approach to improve truthfulness of the validator by training a classifier on the internal state of the validator. We can human annotated data where each attribute is labeled by humans to train this classifier.
3. The truthfulness classifier is light-weight and therefore can be easily trained/evaluated with another LLM.
4. The truthfulness classifier can be connected to the above flowchart. If the truthfulness prediction is low, we can pass the demonstration for human verification.

Instruction-tuning of a validator LLM (P4):

1. In all the above approaches, we'll be limited by the capabilities of LLMs. This can result in a lot of data being sent back to humans for verification. Therefore, we'll need to finetune a LLM for the task for validation itself.
2. Given human annotated validation demonstration data, containing Q: {query, context, answer, attribute1, attribute2...} and corresponding human labeled answers, we can use it finetune an LLM.
3. This can help improve the accuracy of LLM for validation tasks.

LLM-based AutoCorrections

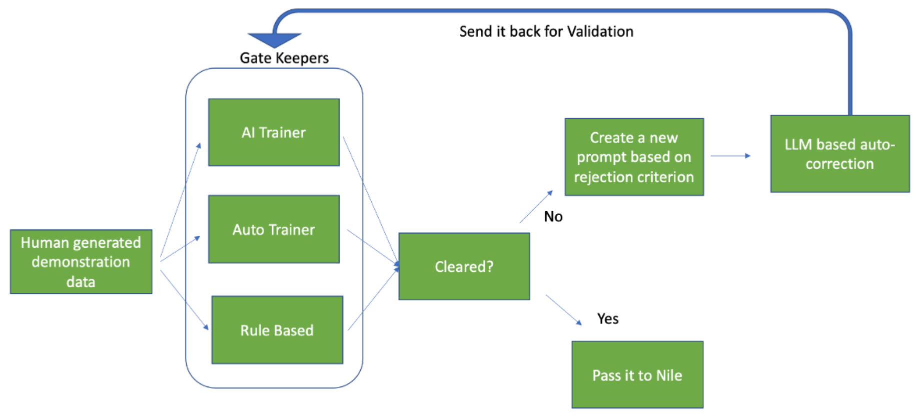
Model based validation can result in a lot of demonstrations being rejected. We can avoid that by using another LLM to correct the answers generated by humans. These answers will be sent to back for validation using the same model based validation workflow. If the attribution noise is higher even after correction, it can be sent back to humans for verification. The flowchart is shown below:


Send it back for Validation
Validation
Ashop LLM
Anthropic LLM
Nile LLM
GPT-4
Human generated demonstration data
Aggregate statistics
Attribution noise
Use LLM for auto-correction
High noise
High noise after auto-correction
Human Verification
Low noise
Pass it to the next stage

Risk: However, since the current LLM validation models prefer LLM generated text, it might create a loop where LLM autocorrected statements are preferred by the validator over human generated original text.

Improved Gatekeeping

Currently we use a strike based mechanism to discard bad demonstration data (link). This can result in a large percentage of data being rejected by AI trainer or rule based or model based validators. On the other hand, involving human for re-checking all the rejected conversations is not scalable. We can use LLM based auto-correction to regenerate a sample output given the original demonstration and rejection reason (rule based/model based/AI trainer reasons). The regenerated data along with original demonstration data can be sent back to an LLM or Human annotator to rank which one they prefer. The flowchart is shown below:


Send it back for Validation
Gate Keepers
AI Trainer
Auto Trainer
Rule Based
Human generated demonstration data
Cleared?
No
Yes
Create a new prompt based on rejection criterion
LLM based auto-correction
Pass it to Nile

Human Validation Sampling Strategy

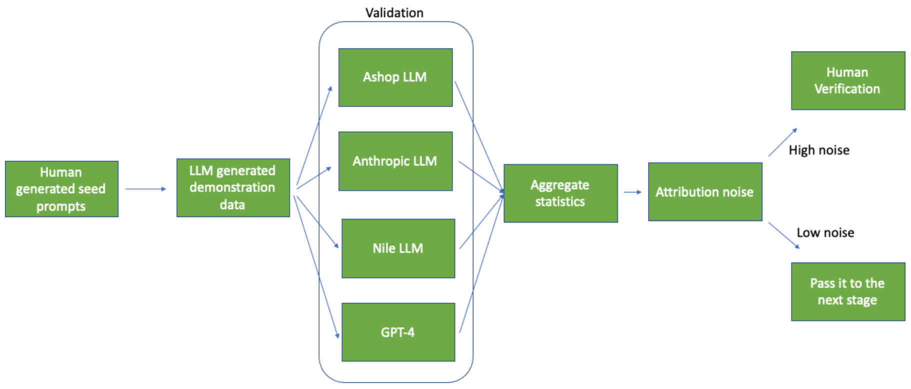
Problem Statement: How humans can annotate 40000 attributes for validation? How do we decide which attributes to get annotated or which demonstration to get annotated?

1. Initial attribute level classification based on Anthropic LLM validation: Based on this doc: Anthropic Model based Data Validation, we can decide with attribute is LLM more accuracy and confident on as compared which of those we need human verification.
2. Later on Insights team can provide the most critical attributes for model improvements which can then be human verified. Rest of the non-critical attributes can be verified by LLMs.

Risks

1. **Bias in demonstration data:** Demonstration data generated either using Humans or LLMs will have bias. This bias can creep into the the trained Nile model itself. We can minimize bias through these:
    a. When using LLM to generate demonstration data:
        i. We should include guardrails when generating data using LLMs by additional prompts to ensure that LLM generated text is not biased.
        ii. Make LLM robust to adversarial scenarios using red teaming approaches: https://huggingface.co/blog/red-teaming
        iii. Principal driven self-alignment where we provide in-context demonstration of certain principles that LLM should follow: https://arxiv.org/abs/2305.03047
    b. When using humans to generate demonstration data:
        i. Even when humans are generating the data, inherent biases can creep in depending on which geographic location is the annotator located in.
        ii. Self-consistency metric can help filter out biased demonstration where we can compare human generated demonstration with another LLM (ChatGPT, Anthropic) generated demonstration.
2. **Noisy validation of demonstration data:** Recent work has shown that using GPT-4 LLM for validation has a Spearman correlation of 0.51 when compared to Human annotators for attributes like Coherence, Consistency, Relevance etc. on SummEval benchmark and 0.57 on Topical-Chat benchmark. This raises few questions:

   a. How does noisy samples from incorrect model based validation affect Nile model training? Nile data engine can help answer if we can speed up end to end workflow.
   b. We can use some of the methods discussed for reducing bias here as well for reducing the validation noise.
3. **Using LLM for generation and validation:** LLMs are biased towards LLM generated data and this can create a self-reinforcing loop and can result in a model which is not aligned with Humans. We have to keep validating a random subset of data to ensure that LLM validation is not getting biased. Another solution can be to use multiple LLMs to validate the demonstration and use self-consistency to discard the validations where LLMs disagree with each other. The demonstration data where LLMs disagree on the attribution can then be passed to another human for further check. The flow chart is shown below:



4. **Long tail of possible queries and demonstrations:** We might not be able to cover all possible demonstration which can be used to train Nile models. The long tail and adversarial queries asked by production users can impact the overall reviews of the experience. We should incorporate negative samples in the demonstration data using red teaming.

Data Sources

There are two main data sources: Human generated and synthetic LLM generated data. Each data source can be validated either through human annotations of attribution or model based methods (e.g. using another LLM to evaluate). In the appendix, we have created the list of all possible data source and validation combinations along with corresponding pros and cons and possible follow up questions.

Appendix:

APPROACH 1: HUMAN GENERATED AND HUMAN VALIDATED (PRIORITY: 0)

- **Data Source**: Human Generated
- **Validation**: Human Validation
- **Pros**: High quality data and validation.
- **Cons**: Not scalable to the amount of data which we need. We want to minimize human in the loop.

APPROACH 2: HUMAN GENERATED AND LLM EVALUATED. (PRIORITY: 1)

- **Data Source**: Human Generated
- **Validation**: LLM Validated

- **Pros**: High quality data. Current analysis shows that for good data LLM evaluation matches human evaluation for a subset of demonstrations. Additionally, since the validation is scalable, it will increase the overall throughput of the generated data.
- **Cons**: It is not clear how good is the LLM evaluation for noisier data.
- **Further question**:
    - How does the noise in LLM evaluated data affect the final Nile model KPIs? We need a risk based analysis here comparing the scale of data needed to train Nile models vs the potential noise that can be introduced because of using LLM for evaluation.
    - Can we improve the quality of LLM evaluation using self-consistency?
    - Should we use LLM evaluations only for a subset of attributes?

APPROACH 3: LLM GENERATED AND LLM EVALUATED. (PRIORITY: 2)

- **Data Source**: LLM Generated
- **Validation**: LLM Validated
- **Pros**: Most scalable solution to use LLM for both generation and validation.
- **Cons**:
    - LLM can generate noisy and non diverse dataset. Recent paper has shown that quality of data rather than quantity is more important for instruction tuning of the model. So, even though LLM can generate and validate a large amount of data, is it useful for final Nile model KPIs?
    - Also, LLM validators have shown a bias towards LLM generated text which can result in a self-reinforcing loop.
- **Further question**: Similar to above, how does the noise in LLM generated data affect the final Nile model KPIs? Maybe we can use a combination of clean human generated data and LLM generated data. We will discuss that below.

APPROACH 4: LLM GENERATED AND HUMAN EVALUATED. (PRIORITY: 2)

- **Data Source**: LLM Generated
- **Validation**: Human Validated
- **Pros**: Scalable solution for generating demonstration using LLM and evaluating it using humans. Human in the loop evaluation will ensure that model does not generate garbage demonstration. We can use techniques like RLHF to ensure that the model generates quality output.
- **Cons**:
    - Less scalable than using LLM for both generation and evaluation.
    - Prompt engineering using good quality diverse prompts will be needed to ensure that LLM model is aligned to our data needs. This might need more human data. Check this paper. We will discuss it more below.

APPROACH 5: (LLM + HUMAN) GENERATED AND HUMAN EVALUATED. (PRIORITY: 3)

- **Data Source**: LLM Generated with human in the loop for prompts or seed demonstrations.
- **Validation**: Human Validated
- **Pros**: Scalable solution for generating demonstration using LLM and evaluating it using humans. Human in the loop prompt generation ensures that we get diverse data. Human in the loop evaluation will ensure that model does not generate garbage demonstration.

- **Cons**:
    - Similar to approach 4, this approach is less scalable than using LLM for both generation and evaluation.
    - Prompt engineering would need more human effort.
    - Human evaluation won't scale.

APPROACH 6: (LLM + HUMAN) GENERATED AND (LLM + HUMAN) EVALUATED. (PRIORITY: 3)

- **Data Source**: LLM Generated with human in the loop for prompts or seed demonstrations.
- **Validation**: Human Validation for randomly sampled subset of data to ensure that we are not feeding garbage to Nile models. For most of the data, we will use LLM to validate.
- **Pros**: Scalable solution for generating and evaluation demonstration using LLM. Human in the loop can ensure that the data is not garbage. We still have to figure out how to perform LLM + Human based evaluation.
- **Cons**:
    - Similar to approach 4, this approach is less scalable than using LLM for both generation and evaluation for the whole dataset.
    - Prompt engineering would need more human effort.
    - It is not clear which percentage of data should be evaluated using humans vs LLMs. We will need risk based analysis here as well to closely align with Nile model KPIs.