

The Only Business-User-Led Path to Millions of Enterprise AI Agents

Enterprise GenAI suffers from one crucial constraint - even the best AI models can't know the unique terminologies, processes and context of each enterprise because that information is not in its training data. This can be handled in post-training where models are fine-tuned on thousands of examples of enterprise-specific data. instance, enterprises rarely have clean labeled data to enable such fine-tuning and business users don't get immediate benefit from providing such feedback. This becomes especially important at scale. **Enterprises need a completely automated way for end-users to customize AI on their unique data, on their enterprise terminology, for their specialized process, all in their own cloud or edge for security reasons.**

Aible's Interns (AI Agents) solve this problem using the following approach

1. Reasoning Models

Aible has developed fine-tuned models for specific tasks such as SQL generation or document summarization, with an added emphasis on transparency: these models provide clear reasoning steps that users can provide feedback on (please see Reasoning Examples section below). For instance, if the AI selects the wrong variable from the data to answer a user's question, it will clearly indicate which variable was used. To achieve this level of clarity, Aible fine-tuned the Llama 3.3 70 Billion parameter model with just over a thousand examples of Chain of Thought (CoT) reasoning at a cost of **\$4.11** total. Of course, we could have started from a larger model and we will have much more such training data over time, but this was sufficient to effectively demonstrate our method.

2. Coachable Reasoning Models

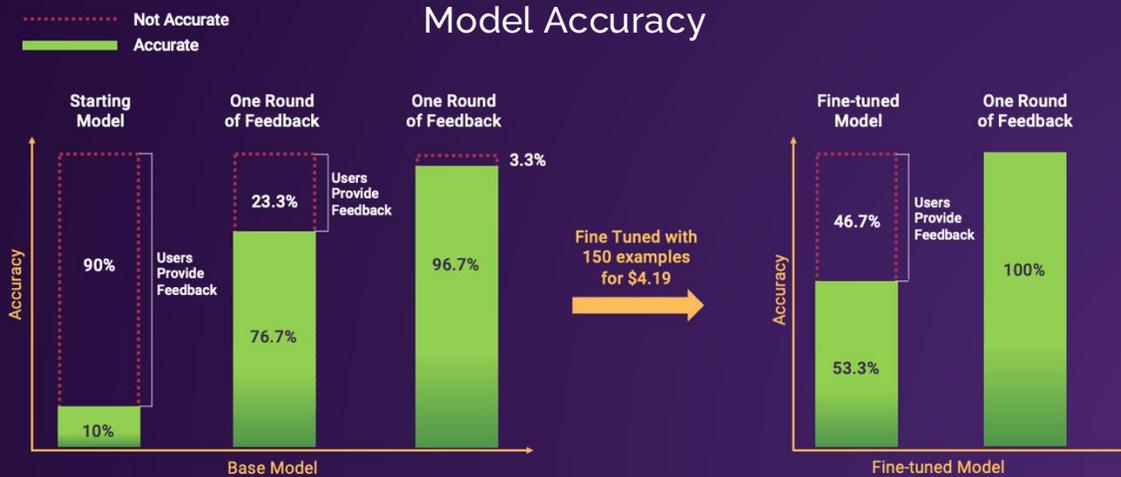
By providing feedback on specific reasoning steps, users can pinpoint exactly where the AI went wrong without having to know how to read SQL. In this case, the user may indicate the correct variable the AI should have used to solve the problem. At first pass, the AI was only **10%** accurate on this use case, which is the same accuracy as the base Llama model for this use case. Using Nemotron 49B would have given us a better starting accuracy. We are working with NVIDIA to use their upcoming NeMo Customizer to fine-tune Nemotron when that becomes available next month.

3. Real-time User Augmented Improvement

With user feedback, the AI immediately retries the question and refines its response. If the resultant answer is correct, the user confirms this and the AI saves the relevant feedback. After just one round of user feedback, the AI achieved an accuracy of **76.7%**. If the answer was still incorrect, the users could provide additional feedback and have the AI try again. The AI was **96.7%** accurate after up to two rounds of feedback. Note that it is extremely difficult to achieve accuracy of over 70% in enterprise-specific use cases even with the largest models.

4. Post-trained Model Improvement

Once a small amount of feedback is received (in the example below, just 150 such user feedback was used) the model is automatically post-trained to specialize it on this specific enterprise use case. The post-training cost was just **\$4.19** but caused the model to improve such that its first-pass accuracy is **53.3%**. This model, after just one additional round of user feedback answered **every question correctly**.

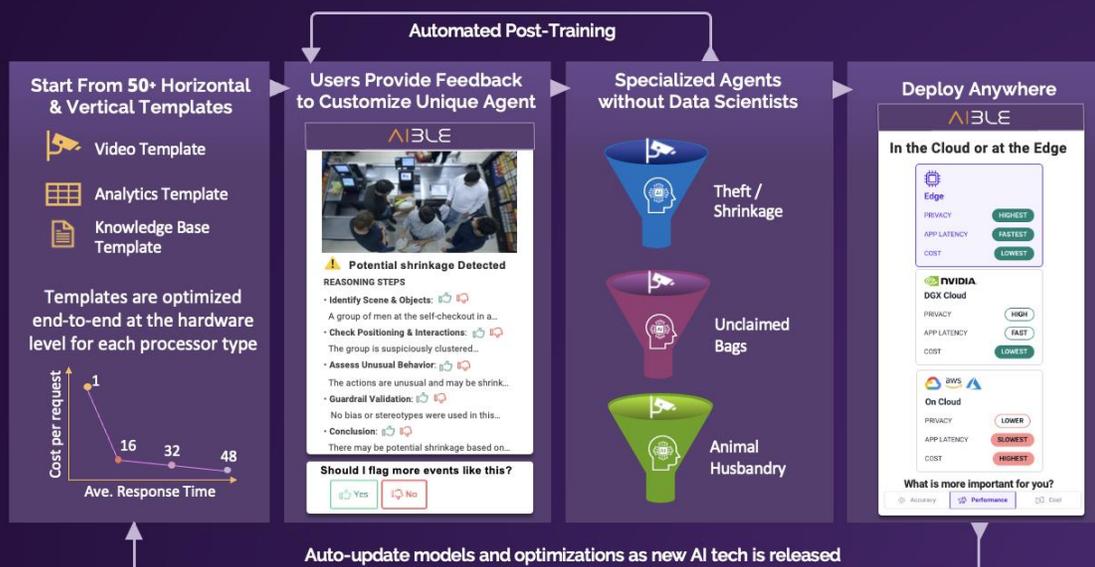


The key here was that this improvement process was completely automated, required no data science skills, and the feedback could be provided by end-users. More importantly, here the business users were incentivized to provide feedback because they got an immediate payback - the improved answer to their questions.

Moreover, in this case we saw an extraordinary improvement in the accuracy of the model with a much smaller amount of user feedback than is typically required. This is because the feedback on specific reasoning steps is much more actionable than feedback on just the output of the AI. Imagine a student that you keep asking questions to and only tell them whether the answer was correct or not. Compare that to a student where you explain exactly what step they got wrong and how they could improve their answer. The second student will learn much faster with a lot less feedback. That is exactly what is happening here.

Enterprises have been disappointed with the level of accuracy they are getting from so-called expert AI agents. We call our starting points 'interns' precisely because they are not yet specialized agents. But with user guidance, they become experts with just 100 or so examples of feedback. We will build hundreds of use-case-specific interns that end users can make their own by simply providing feedback. This is the only way to build millions of enterprise AI agents. As base models improve, such as when the next major version of a model is released, we can simply apply all of the saved enterprise-specific feedback to fine-tune even better specialized models from the new base model completely automatically. This is critical because even the most advanced models, up to and including Artificial General Intelligence, will still need help understanding the unique terminologies and processes of a specific enterprise.

Business AI Agents for Every Use Case @ Scale



Reasoning Examples

The addition of reasoning to a model can significantly improve its accuracy. For example, this was a key component of the accuracy improvements of DeepSeek R1. But CoT reasoning is often verbose and meandering which makes it difficult for business users to understand and provide feedback on the reasoning steps. For example, the following image illustrates the CoT reasoning steps of a leading model when asked “**Which locations have more than 50% of housing units built before 1979**” based on a publicly available table of Social Determinants of Health data which includes this information because houses built before 1979 may have used lead paint. Note: Some of the large language models seem to have been trained specifically on this public dataset. For true enterprise data that the models have never seen, they do much worse.

This model spends a lot of time thinking about the problem, changes its mind a few times and then incorrectly assumes the variable ACS_PCT_HU_BUILT_1979_ZC indicates the percent of houses built in or *after* 1979. Moreover, because its thinking steps are very unstructured, it is difficult for business users to understand or provide feedback on the steps.

Thinking

Okay, let's tackle this query. The user wants to find locations where more than 50% of the housing units were built before 1979. Let's start by understanding the table structure and the relevant columns.

Looking at the table definition, the columns related to housing unit construction years are:

- ACS_PCT_HU_BUILT_1979_ZC : This seems to be the percentage of housing units built in 1979 or later. The name includes "1979", and the prefix "ACS_PCT" indicates it's a percentage. So if this is the percentage built in 1979 or later, then the percentage built before 1979 would be 100% minus this value.

Wait, the column name says "HU_BUILT_1979_ZC". Typically, in data like this, a column named with a specific year might refer to "built before" or "built in or after" that year. Since the percentage is given for 1979, I need to confirm whether this represents units built in 1979 and later, or before 1979. The naming convention here is a bit ambiguous. Let's think: if the column is named "ACS_PCT_HU_BUILT_1979_ZC", it might actually represent the percentage of housing units built in 1979 or later. For example, in some datasets, a column like "PCT_BUILT_1979" could mean built in 1979 or after. So to find the percentage built before 1979, we would subtract this value from 100.

So, the logic would be: $100 - \text{ACS_PCT_HU_BUILT_1979_ZC} > 50$. That translates to $\text{ACS_PCT_HU_BUILT_1979_ZC} < 50$.

Because if 50% or less were built in 1979 or later, then more than 50% were built before.

But wait, let me check if there's another column that directly represents housing units built before a certain year. Looking through the table again, the other related columns are:

- ACS_PCT_10UNITS_ZC : Probably percentage of units in 10+ unit buildings, not relevant.
- ACS_PCT_HU_NO_FUEL_ZC : About fuel, not construction year.
- ACS_MEDIAN_YEAR_BUILT_ZC : This is the median year built, but we need a percentage, not the median.

So the only column that gives a percentage based on a year cutoff is ACS_PCT_HU_BUILT_1979_ZC . Therefore, assuming that this column represents the percentage of housing units built in 1979 or later, the percentage built before 1979 would be 100 minus this value.

So the condition we need is:

```
100. - ACS_PCT_HU_BUILT_1979_ZC > 50
```

Which simplifies to:

```
ACS_PCT_HU_BUILT_1979_ZC < 50
```

Because if less than 50% were built in 1979 or later, then more than 50% were built before.

Now, the query needs to select locations (probably based on ZCTA, STATE, REGION, etc.) where this condition is true. The user probably wants the ZCTA (ZIP Code Tabulation Area) along with the state and region for location identification.

So the SELECT clause would include columns like ZCTA, STATE, REGION, and possibly the percentage for reference.

Putting it all together:

```
SELECT ZCTA, STATE, REGION, ACS_PCT_HU_BUILT_1979_ZC  
FROM data
```

Which simplifies to:

```
ACS_PCT_HU_BUILT_1979_ZC < 50
```

Because if less than 50% were built in 1979 or later, then more than 50% were built before.

Now, the query needs to select locations (probably based on ZCTA, STATE, REGION, etc.) where this condition is true. The user probably wants the ZCTA (ZIP Code Tabulation Area) along with the state and region for location identification.

So the SELECT clause would include columns like ZCTA, STATE, REGION, and possibly the percentage for reference.

Putting it all together:

```
SELECT ZCTA, STATE, REGION, ACS_PCT_HU_BUILT_1979_ZC
FROM data
```

```
WHERE ACS_PCT_HU_BUILT_1979_ZC < 50;
```

This way, the result shows the percentage built before 1979 explicitly.

But wait, what if the column `ACS_PCT_HU_BUILT_1979_ZC` actually represents the percentage built before 1979? That would change everything. How to confirm?

Looking back at the column name: "ACS_PCT_HU_BUILT_1979_ZC". The suffix "_ZC" is probably for the ZCTA. The naming could be ambiguous. For example, in some datasets, "PCT_BUILT_BEFORE_1979" would be clear, but here it's just "BUILT_1979".

Alternatively, maybe the column is the percentage of housing units built in 1979, not before or after. But that's less likely because having a percentage for a single year wouldn't be useful for this kind of query. More likely, it's a cutoff year.

Given the ambiguity, but based on common naming conventions, I think the column represents the percentage built in 1979 or later.

Therefore, subtracting from 100 gives the percentage before.

So the query as written above should be correct.

Also, need to ensure that the column is not null. But the problem statement doesn't mention handling nulls, so we can assume that relevant rows have valid data.

Therefore, the final query selects ZCTA, STATE, REGION, calculates the percentage before 1979, and filters where the 1979-or-later percentage is less than 50.

Generated SQL

```
SELECT
  ZCTA,
  STATE,
  REGION,
  (100 - ACS_PCT_HU_BUILT_1979_ZC) AS pct_before_1979
FROM
  data
WHERE
  ACS_PCT_HU_BUILT_1979_ZC < 50;
```

The Aible Intern model provides much more concise and structured thinking output which enables the user to provide specific feedback on the specific step where the intern made the mistake.



Thinking

- 1. Understand the question:** The question asks for locations (zip codes) where more than 50% of the housing units were built before 1979.  
- 2. Identify the relevant column:** The column `ACS_PCT_HU_BUILT_1979_ZC` seems to represent the percentage of housing units built in 1979 or later. However, the question asks for housing units built before 1979, so we need to consider the opposite of this column.  

What did I do wrong?

`ACS_PCT_HU_BUILT_1979_ZC` is the percentage of housing units built before 1979

Improve Model

Cancel

3. **Formulate the condition:** The condition for the query is that the percentage of housing units built before 1979 is more than 50%. Since the `ACS_PCT_HU_BUILT_1979_ZC` column represents the percentage of housing units built in 1979 or later, we can calculate the percentage of housing units built before 1979 by subtracting this column from 100.



4. **Determine the query:** We need to select the `ZIPCODE` column from the `data` table where the calculated percentage of housing units built before 1979 is more than 50%.



Here is the SQL query to answer the question:

Generated SQL

```
SELECT ZCTA
FROM data
WHERE 100 - ACS_PCT_HU_BUILT_1979_ZC > 50;
```



Retry Question with Feedback

With this clarification the intern gets the answer correct and learns this information for the future.



Thinking

1. **Understanding the question:** The question asks for locations where more than 50% of housing units were built before 1979. This implies that we need to find the zip codes (or locations) that meet this condition.



2. **Identifying the relevant column:** The column `ACS_PCT_HU_BUILT_1979_ZC` represents the percentage of housing units built before 1979. This is the column we will use to filter the locations.



3. **Filtering the locations:** To find locations with more than 50% of housing units built before 1979, we need to filter the rows where `ACS_PCT_HU_BUILT_1979_ZC` is greater than 50.



4. **Selecting the relevant columns:** Since we are interested in the locations, we need to select the `ZIPCODE` column, which represents the zip code of each location.



5. **Writing the SQL query:** Based on the above steps, we can write the SQL query to find the locations that meet the condition.



Here's the SQL query that answers the question:

Generated SQL

```
SELECT ZIPCODE
FROM data
WHERE ACS_PCT_HU_BUILT_1979_ZC > 50;
```



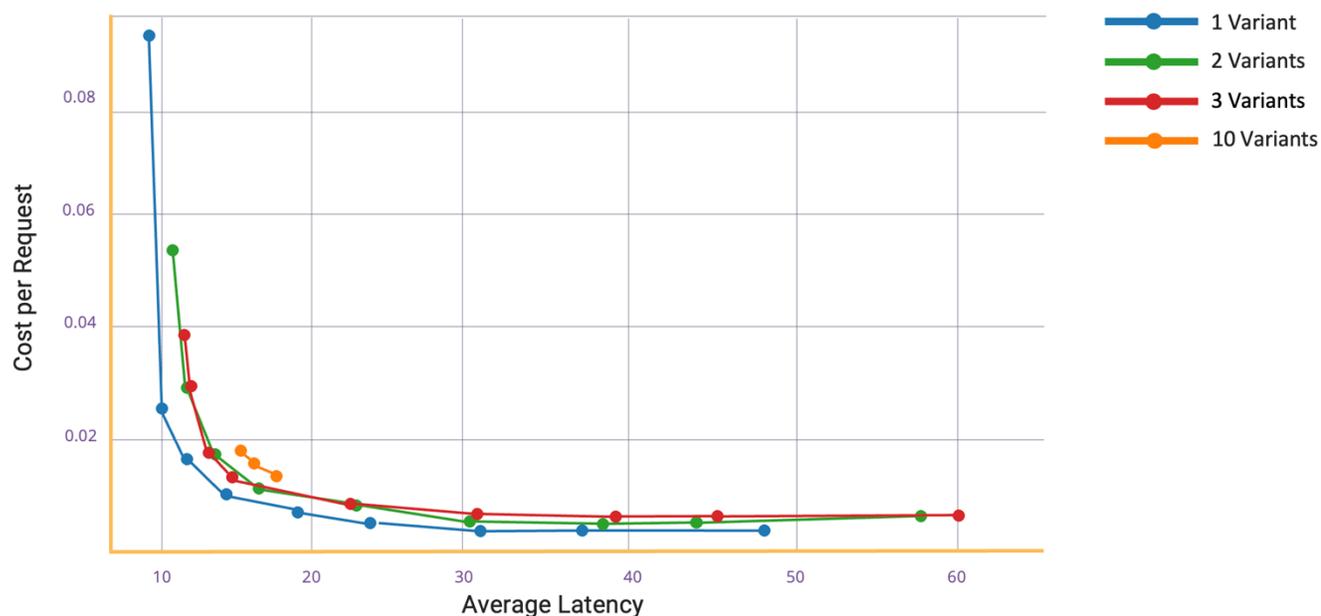
Managing many different fine-tuned models at scale

The Aible Intern approach automatically creates many fine-tuned models specific to enterprises, use-cases, even users. This is the only way to achieve thousands of agents in the enterprise because manual data scientist driven approaches will never get us there. But what is the implication of this on the Cost-Performance tradeoff of models running on GPUs? As was highlighted in NVIDIA's DGX Cloud keynote at GTC, Aible demonstrated 30X better GPU utilization.

We evaluated the cost-per-request vs. average latency curve for a single model, two variants (different post-trained versions) of the same base model, three variants of the same base model, and ten variants of the same base model. The curves were fairly consistent. This shows that even if we utilize multiple specialized variants for different use cases, they can be run efficiently on GPUs at higher concurrencies to achieve close to the cost-performance characteristics achieved with single model variants. Of course, the optimal concurrency settings are different depending on the number of variants being run in parallel. You can see details at:

www.aible.com/aible_intern_model

Cost per Request vs Concurrency for Each Model Configuration



Implications for Agent performance

The Aible Intern approach allows smaller fine-tuned models to perform much better than larger models. This allows us to fit several such smaller fine-tuned models in the GPU part of a single NVIDIA or Apple GPU-CPU combined architecture superchip with shared memory. We then run all of the agent coordination, retrieval augmented generation (RAG), fine-tuning coordination, etc. from the CPU part of the superchip. All of the components can leverage the shared memory to run agents blazingly fast. You can see details at:

www.aible.com/industrialized-agents-at-scale

Serverless Architecture

All of the Aible capabilities including the agent coordination and Retrieval Augmented Generation can be run up to **200X more cost efficiently** on DGX Cloud Serverless Inference compared to dedicated servers. Aible processes data where it is, in Snowflake, BigQuery, etc. and only moves metadata to DGX Cloud for processing, thus addressing the data gravity problem for customers.

Starting Point for Enterprise Use Cases

Aible's 50+ use case templates (soon 100+) give business users an easy way to try, buy, and experience the power of DGX Cloud Serverless Inference and/or Grace Hopper (soon Blackwell) converged architectures. Each template comes with sample data and sample questions that enable users to experience these agents in action immediately. They can then easily use these agents on their own data and customize them by providing feedback on the reasoning steps. This kind of guided experience gets users adopting the underlying NVIDIA technology much faster.

As the number of agents per employee explodes in enterprises, it will be crucial to enable business users to coach such agents themselves. Aible uniquely gives business users a way to start from useful templates to build and coach their own agents without requiring any data science skills, while automatically optimizing them to run efficiently on NVIDIA infrastructure securely at the enterprise cloud and edge.