CHAPTER 15

Intelligent Telemetry

A telemetry system is responsible for collecting observations about how users are interacting with your Intelligent System and sending some or all of these observations back to you.

For example, a telemetry system might collect information every time any of the following happens:

- The user visits a particular part of the application.
- The user clicks a particular button.
- There is an unacceptably long loading time.
- A client connects to a server.
- The server sees a malformed query.
- The server is low on RAM.

Telemetry is used to verify that a system is working the way it is supposed to. In an Intelligent System, telemetry also contains information about the interactions users had with the intelligence and the outcomes they achieved. These are the signals intelligence creators need to improve intelligence.

Why Telemetry Is Needed

There are three main tasks for telemetry in an Intelligent System:

1. Making sure the system is working the way it is supposed to.
2. Making sure users are getting the intended outcomes.
3. Gathering data to create new and better intelligence.

We will explore each of these in more detail.

G. Hulten, *Building Intelligent Systems*, https://doi.org/10.1007/978-1-4842-3432-7_15

Make Sure Things Are Working

Telemetry should contain data sufficient to determine that the intelligence implementation is (or isn't) working. This should include:

- That the intelligence is flowing where it is supposed to, when it is supposed to.
- That the runtime is properly loading and interpreting the intelligence.
- That contexts are being properly collected at the runtime.
- That contexts are being properly converted into features.
- That the runtime is executing models reasonably, without errors.
- That the predictions of various intelligence sources are being combined as intended.
- That the outputs from intelligence are being interpreted correctly and showing the right user experiences.
- And so on.

These are the types of things you would want to include in the telemetry of just about any service or application (and most services will want telemetry on many more things, like: latency, uptime, costs, and utilization).

One important use for this type of telemetry in an Intelligent System is to ensure that the intelligence runtime environment is behaving the same way the intelligence creation environment is; that is, that the intelligence is being used in same conditions that it was created in. These types of bugs are very hard to find: the whole system is running, the user is interacting, nothing is outputting an error, nothing is crashing, but user outcomes aren't as good as they could be because sometimes the intelligence is simply making extra mistakes that it doesn't need to.

Understand Outcomes

Telemetry should contain enough information to determine if users are getting positive or negative outcomes and if the Intelligent System is achieving its goals. It should answer questions like these:

- Which experiences do users receive and how often do they receive them?
- What actions do users take in each experience?
- What experiences tend to drive users to look for help or to undo or revert their actions?
- What is the average time between users encountering a specific experience and leaving the application?
- Do users who interact more with the intelligent part of the system tend to be more or less engaged (or profitable) over time?

This type of telemetry should be sufficient to ensure that the system's experiences are effectively connecting users to intelligence, that users are being directed where they are having better interactions, and that they are behaving in ways that indicate they are getting good outcomes.

For example, imagine a system to help doctors identify broken bones in x-rays. To help understand outcomes, the telemetry should capture things like:

- How long doctors look at X-rays where the system thinks there is a broken bone.
- How long doctors look at X-rays where the system thinks there is not a broken bone.
- How many times doctors order treatment when the system thinks there was a broken bone.
- How many times doctors order treatment when the system thinks there is not a broken bone.
- How many times patients are re-admitted for treatment on a broken bone that the system flagged, but the doctor decided not to treat.

This type of telemetry should help you understand if patients are getting better outcomes because of the intelligence or not. It should also help you diagnose why.

For example, if the intelligence is correctly identifying very subtle breaks, but doctors aren't performing treatment—why is it happening?

Maybe patients are getting bad outcomes because the intelligence makes lots of mistakes on subtle breaks so doctors don't trust that part of the system and are ignoring it.

Or maybe patients are getting bad outcomes because doctors simply aren't noticing the user experience that is trying to point out the breaks to them—it needs to be made more forceful.

Telemetry on user outcomes should help you identify and debug problems in how intelligence and experience are interacting with users, and take the right corrective actions.

Gather Data to Grow Intelligence

The telemetry should also include all the implicit and explicit user feedback needed to grow the intelligence.

This includes:

- The actions users took in response to the intelligent experiences they saw.
- The ratings the users left on content they interacted with.
- The reports users provided.
- The escalations the users made.
- The classifications users provided.
- And all the implicit indications that the user got a good or a bad outcome.

Review Chapter 9, "Getting Data from Experience," for more details.

In order for this type of telemetry to be useful, it must contain all of the following:

1. The context the user was in at the time of the interaction.
2. The prediction the intelligence produced and what experience it led to.
3. The action the user took.
4. The outcome the user got (either implicitly or through explicit feedback).

And it must be possible to connect these four components of a single interaction to each other in the intelligence-creation environment. That is, it must be possible to find a particular interaction, the context the user was in, the prediction the intelligence made, the action the user took, and the outcome the user got from the interaction—all at the same time.

This type of telemetry is the key to making an Intelligent System that closes the loop between usage and intelligence, allowing the users to benefit from interacting with intelligence, and allowing the intelligence to get better as users interact with it.

Properties of an Effective Telemetry System

Of course telemetry is good. You want more of it—heck, intelligence creators are going to want all of it. They are going to want every last detail of every last interaction (no matter the cost). And if you don't give it to them they will probably complain. They'll ask you why you are trying to ruin them. They'll wonder why you are trying to destroy their beautiful Intelligent System.

(Not that I've done that. I'm just saying. It could happen…)

This section discusses ways to balance the cost and value of telemetry. These include:

- Sampling
- Summarizing
- Flexible targeting

Sampling

Sampling is the process of randomly throwing away some data. That is, if there are 10,000 user interactions with the system in a day, a telemetry system sampling at 1% would collect telemetry on 100 of the interactions (and collect nothing on the remaining 9,900 interactions).

This sampled data is cheaper to collect, easier to store, faster to process, and can often answer key questions about the system "well enough."

The simplest form of sampling is uniform. That is, whatever type of event happens, sample 1% of them. But sampling is often targeted differently for different types of events. For example, one policy might be to sample telemetry related to verifying the

implementation at 0.01% and sample telemetry related to growing intelligence at 10%, except for telemetry from France (where the intelligence is struggling) which should be sampled at 100%.

Common ways to define populations for sampling include:

- Separate by geography—for example, sample 20% of the events in Europe and 1% in Asia.
- Separate by user identity—for example, sample 0.01% of the events for most users, but sample 100% of the events for these 10 users who are having problems.
- Separate by outcome (what the user did after seeing the intelligent experience)—for example, sample at 2% for users who had normal outcomes and at 100% for users who had a problem.
- Separate by context (what the user was seeing before the intelligence fired)—for example, sample 10% of the events that occur at night and 0.1% that occur during the day.
- Separate by user properties (gender, age, ethnicity, and so on)—for example, sample at 80% for new users and at 1% for long-time users.

One particularly useful idiom is to sample 100% of data from 0.1% of users, so you can track problems that occur across the course of a session; and also sample 1% of events from 100% of users, so you can track problems that occur in aggregate.

And note: the telemetry system must record the sampling policy used to gather each piece of telemetry it collects, so the intelligence creator can correct for the sampling method and produce accurate results when they use the telemetry.

Summarizing

Another approach to reducing telemetry data is summarization. That is, taking raw telemetry, performing some aggregations and filtering of it, and storing a much smaller summary of the data.

For example, imagine you want to know how many users undo an automated action per day. You could keep raw telemetry around and calculate the count whenever you need it. Or you could calculate the count every day, save it, and delete the raw telemetry. The size of raw telemetry grows with usage (and can get very big); the size of a count is constant no matter how many users you have.

Summaries can be very effective for understanding how the system is working and if it is achieving its goals. And as you come to understand your system, and identify critical summaries, you can transmit and retain less and less raw telemetry over time.

Summarization can be done on the client or on the service.

To perform *client-side summarization*, the client simply collects telemetry for some period of time, aggregating as the telemetry arrives, and periodically reports the summarized results to the telemetry server (where they may be combined with aggregations from other clients).

Client-side summarization is useful when bandwidth is a big driver in cost.

Server-side summarization is useful when bandwidth for telemetry is either free (it is a byproduct of how the client accesses server-side intelligence) or when storage is a serious cost driver. In this case, a summarization job can be run periodically to extract all the relevant measurements. Once complete, the raw data is purged to save space.

Flexible Targeting

There are many types of telemetry in a system; some types are more valuable than others, and the value of a particular type of telemetry can change over time.

For example, early in the system development, before all the code is working properly, telemetry about implementation details—features, model deployments, and so on—can be extremely useful. But once the system is verified and deployed to users, and runs successfully for months, this type of telemetry becomes less useful.

And once the system starts attracting a lot of users, telemetry that helps grow the intelligence becomes more important: it captures all the value of users' interactions, their judgments, their preferences. While the intelligence is growing, this type of telemetry is the core value of the system, the key to achieving success.

And once the intelligence plateaus (if it does), additional training data might become less valuable and telemetry on user outcomes or rare problems might become relatively more important.

But when something goes wrong, all bets are off and telemetry from all across the system could be critical to finding and correcting the problem.

An intelligence telemetry system should support different collection policies for different types of telemetry, and these policies should be changeable relatively quickly and easily—so orchestrators and intelligence creators can do their work.

Common Challenges

This section discusses some common data pitfalls and the role that a telemetry system can have in dealing with them. These include:

- Bias
- Rare events
- Indirect value of telemetry
- Privacy

Bias

Bias occurs when one type of event appears in the telemetry more than it should. For example, imagine getting 10,000 examples of users interacting with content on your Intelligent System. Ten thousand is a good number. That should be enough to understand what is going on, to create some good intelligence.

But what if you find out that all ten thousand interactions were with the same piece of content. Like 10,000 different users interacting with a single news story?

Well, that's not as good.

Or what if you find out all 10,000 samples came from just one user interacting with the service. For some reason, the telemetry system decided to sample just that one user over and over.

That's not good either.

Here are some reasons bias might occur:

1. **The world changes, but the sampling policy doesn't**. You set up a sampling policy that selects 1,000 users to sample at 100% (so you can see end-to-end interactions). Then your user base grows by a factor of 10 and significantly changes, including less technical users, users in other countries, and so on. The initial 1,000 users may have been right at the time you selected them, but they are no longer representative of your user population—this causes bias.
2. **Users behaving in ways you don't expect**. For example, a user who is having a bad experience might simply turn off their computer. If your telemetry system expects them to shut down the app before sending telemetry, you might miss out on their data.

One approach to dealing with bias is to always collect a raw sample across all usage—for example, 0.01% of all interactions—and cross-check your other samples to make sure they aren't wildly different from this baseline sample.

Rare Events

In some settings many events are rare events; a few contexts are prominent, but most contexts your users encounter are infrequent.

For example, in a library, the most popular 100 books might account for 50% of the checkout volume, but after those 100 books, usage becomes highly dispersed, with thousands of books that each get one or two checkouts per month. When a book comes up to the checkout stand, there is a 50% chance it is one of the top books and a 50% chance it is one of the other twenty thousand books.

Imagine a system to help recommend library books in this library.

In order to recommend a book, the system needs to see which other books it is commonly checked out with. But the chances that any pair of rarely used books are checked out at the same time is very, very low.

For example, if the telemetry is created by sampling 10% of the checkout events, after a month you might have:

- 10,000 samples of the top 100 books being checked out.
- 5,000 books with just one or two checkout events.
- 15,000 books with no checkout events at all.

Not great. That kind of data might help you build recommendations for the popular books (which is simple, and you could have probably done it by hand much cheaper), but it won't give you any information on the unpopular books.

One option is stratified sampling. Maybe the top 100 books are sampled at 1%, while the remaining (unpopular) books are sampled at 100%. The amount of data might be similar to using the uniform 10% sampling rate for all books, but the value of the data might be much, much greater for producing the type of intelligence this system needs to produce.

Indirect Value

It's often difficult to quantify the value of telemetry, but quite easy to quantify the cost. Rational people will ask questions like:

- Can we turn the sampling of this particular type of data down from 10% to 1%?
- Can we stop getting any telemetry on this use-case that we don't have any plans to ever add intelligence to?
- Do we really need to store 90 days of historical data, or would 30 do?

These are all reasonable questions, and they are difficult to answer. Here are some approaches that can help:

1. Keep a very small raw sample of all activity, sufficient to do studies to estimate the value of other telemetry.
2. Make it easy to turn on and off telemetry, so that intelligence orchestrators can turn telemetry up quickly to track specific problems, and intelligence creators can try new approaches for short periods of time to see if they are worth the cost.
3. Set a reasonable budget for innovation, and be willing to pay for the potential without micromanaging each specific detail and cost.

Privacy

Privacy, ah privacy.

When using customer data, it is important to "do the right thing" and treat your user's data the way you would want someone else to treat your data. Keep in mind some best practices:

- Make sure customers understand the value they are getting from your product and feel they are getting a good deal.
- Try as much as possible to scrub any personal information (like names, credit card, social security, addresses, user generated content, and so on) out of telemetry you store.

- Consider an aggregation policy for data you intend to store long term that combines information from many users to make it harder to identify any single user's data (for example, storing aggregate traffic to each piece of content instead of storing which content each user browsed).
- Have a reasonable retention policy and don't keep data for too long.
- Make sure telemetry is handled with utmost care from hacks and accidental leaks.
- Make sure customers know what you are storing and how to opt out.
- Use data for the purpose it was collected, and don't try to re-purpose it for wildly different activities.
- Make sure you know the laws that apply in the countries where your service is operating.

But remember: you need data—you can't create intelligence without data…

Summary

Good telemetry is critical to building an effective Intelligent System. No way around it: without telemetry you won't have an Intelligent System.

Telemetry is used for three main purposes:

1. To make sure the system is working the way it is intended to work—that there aren't any bugs or problems.
2. To make sure that users are getting the outcomes you want them to get—that everything is coming together for success.
3. To grow the intelligence over time—that the contexts and outcomes are being recorded in a way you can use to create intelligence.

The value of these types of telemetry will change over time, and a system should be able to adapt. A good telemetry system will:

- Support sampling that allows tracking specific focused problems, and also view aggregate effects.
- Allow summarization of events.
- Be flexible in what it measures (and what it doesn't).

Some common pitfalls for telemetry include:

- Bias that makes the telemetry not representative of what users are experiencing.
- Rare events that are important but are very hard to capture in sampled telemetry.
- Proving the value of the telemetry in the face of reasonable questions about the associated costs.
- Protecting user privacy while still allowing intelligence to grow.

For Thought…

After reading this chapter, you should:

- Understand how telemetry enables Intelligent Systems by making sure they are working, achieving their objectives, and allowing them to improve as they are used.
- Be able to design a telemetry system that meets the needs of your application.

You should be able to answer questions like these:
Consider an Intelligent System that you think one of your friends would like to use.

- If it's too expensive to collect all the possible telemetry, how would you limit the telemetry you do collect?
- What facilities would you need to drill in when there are specific problems?