CHAPTER 6

Why Creating Intelligent Experiences Is Hard

This chapter explores some ways that experiences based on intelligence are different from more traditional experiences. And here is the bottom line: intelligence makes mistakes.

Intelligence is going to make mistakes; to create effective intelligent experiences, you are going to have to get happy with that fact. You are going to have to embrace mistakes. You are going to have to make good decisions about how to work with and around the mistakes.

The mistakes intelligence makes:

- Aren't necessarily intuitive.
- Aren't the same from day to day.
- Aren't easy to find ahead of time.
- Aren't possible to fix with "just a little more work on intelligence."

This chapter is about the mistakes that intelligence makes. Its goal is to introduce you to the challenges you'll need to address to create effective intelligent experiences. Understanding the ways intelligence makes mistakes and the potential trade-offs will help you know what to expect, and, more importantly, how to design intelligent experiences that give users the best possible outcomes—and achieve the system's objectives—despite the mistakes that intelligence makes.

Intelligence Make Mistakes

Intelligent Systems make mistakes. There is no way around it. The mistakes will be inconvenient, and some will be actually quite bad. If left unmitigated, the mistakes can make an Intelligent System seem stupid; they could even render an Intelligent System useless or dangerous.

G. Hulten, *Building Intelligent Systems*, https://doi.org/10.1007/978-1-4842-3432-7_6

Here are some example situations that might result from mistakes in intelligence:

- You ask your phone to order pizza and it tells you the population of Mumbai, India.
- Your self-driving car starts following a road that doesn't exist and you end up in a lake.
- Your smart-doorbell tells you someone is approaching your front door, but upon reviewing the video, you realize no one is there.
- You put a piece of bread in your smart toaster, come back five minutes later, and find that the bread is cold—the toaster hadn't done anything at all.

These types of mistakes, and many others, are just part of the cost of using intelligence, particularly when using intelligence created by machine learning.

And these mistakes are not the fault of the people producing the intelligence. I mean, I guess the mistakes could be their fault—it's always possible for people to be bad at their jobs—but even people who are excellent—world class—at applied machine learning will produce intelligences that make mistakes.

So one of the big roles of experience in Intelligent Systems is to present the intelligence so it is effective when it is right, and so the mistakes it makes are minimized and are easy to recover from.

Consider that an intelligence that is right 95% of the time makes a mistake one out of every twenty interactions. And 95% is a very high accuracy. If your intelligence-producers get to 95% percent they are going to feel like they've done a great job. They are going to want to get rewarded for their hard work, and then they are going to want to move on and work on other projects. They aren't going to want to hear you telling them the intelligence isn't accurate enough.

But in a system that has a million user interactions per day, 95% accuracy results in 50,000 mistakes per day. That's a lot of mistakes. If the mistakes cost users time or money, that could add up to a real problem.

Another useful way to think about mistakes is how many interactions a user has between seeing a mistake. For example, a user with 20 interactions per day with a 97% accurate intelligence would expect to see 4.2 mistakes per week.

Is this a disaster?

It depends on how bad the mistakes are and what options the user has to recover.

But consider the alternative to building systems that make mistakes—that is, to demand a perfect intelligence before building a product around it. Perfection is very expensive, in many cases impossible. And perfection isn't needed to get real value from Intelligent Systems—sitting around waiting for perfection is a great way to miss out on a lot of potential. Consider these examples:

- Speech recognition isn't perfect, it probably never will be.
- Search engines don't always return the right answer.
- Game AI is commonly ridiculed for running in circles and shooting at walls.
- Self-driving cars have accidents.

But all of these (somewhat) inaccurate intelligences are part of products that many of us use every day—that make our lives better. Mitigating mistakes is a fundamental activity in producing intelligent experiences.

Intelligence Makes Crazy Mistakes

At the risk of belaboring the point, intelligences make crazy mistakes. Wild, inexplicable, counter-intuitive mistakes.

Consider—if a human expert is right 99% of the time, you might expect the mistakes they make to be close to correct. You assume this expert pretty much understands what is going on, and if they make a mistake it is probably going to be consistent with a rational view of how the world works.

Machine learning and artificial intelligence aren't like that.

An artificial intelligence might be right 99.9% of the time, and then one out of a thousand times say something that is right out of bat-o-bizarro land. For example:

- A system that recommends music might recommend you 999 rock songs, and then recommend you a teen-pop track.
- A smart toaster might toast 99 pieces of bread perfectly, and decide the 100th piece of bread doesn't need any toasting—no matter what, no heat.
- A system that reads human emotion from images might correctly identify 999 people as happy, then it might see my face and say I am sad no matter how big of an idiot-grin I slap on.

These crazy mistakes are part of working with Intelligent Systems.

And even crazy mistakes can be mitigated, recovered from, and minimized. But to design experiences that work with intelligence you should throw out pre-conceived notions of how a rational thing would act. Instead, you will need to develop intuition with the intelligence you are working with. Use it. See what tends to confuse it and how. Think of ways to support it.

One instinct is to talk to the people making the intelligence and say, "Come on, this is crazy. Fix this one mistake. It is killing us!"

That's fine. They can probably change the intelligence to fix the mistake that's bugging you.

But fixing one mistake usually introduces a new mistake somewhere else. That new mistake is likely just as crazy. And you won't know when or how the new mistake will show up. Fixing obvious mistakes isn't always the right thing to do; in fact, playing whack-a-mole with prominent mistakes can be detrimental. Sometimes it's better to let the intelligence optimize as best it can, then support the intelligence by covering over mistakes with elegant experiences.

Intelligence Makes Different Types of Mistakes

And even though you may be sure I'm belaboring the point by now, Intelligent Systems make different types of mistakes; for example:

- The smart-doorbell might say someone is approaching the door when no one actually is; or it might fail to notify its user when someone is approaching the door.
- The smart toaster might undercook bread; or it might overcook it.
- A speech recognition system might incorrectly interpret what the user said; or it might refuse to try to guess what the user said.

Three main types of mistakes here are:

1. Mistaking one situation for another situation.
2. Estimating the wrong value (for example, the correct toast time).
3. Being too confused (or conservative) to say anything at all.

Perhaps the simplest form of this is confusing one situation for another, as in the smart doorbell example. There are four possible outcomes for the intelligence of a system like this (Figure 6-1):

- **True Positive**—When there *is someone standing at the door* and the intelligence thinks there *is someone standing at the door*, it is called a true positive (this is a correct answer, not a mistake).
- **True Negative**—When there *is not someone standing at the door* and the intelligence thinks there *is no one there*, it is called a true negative (this is another type of correct answer, not a mistake).
- **False Positive**—When there *is not someone standing at the door* but the intelligence thinks there *is someone standing at the door*, it is called a false positive (this is a mistake).
- **False Negative**—When there *is someone standing at the door* but the intelligence thinks there *is no one there*, it is called a false negative (this is another type of mistake) .

		The Intelligence says some is	
		There	Not There
Someone Actually is	There	True Positive	False Negative
	Not There	False Positive	True Negative

Figure 6-1. *Different types of mistakes*

It's often possible for the system's intelligence to be tuned to control the type of mistakes it makes, making more of one type of mistake and fewer of the other. (Usually this involves tuning a threshold on the probability output by the models that drive the intelligence.)

For example, when designing the smart-doorbell experience, is it better to have the system alert the user when there is no one at the door, or to fail to alert the user when someone is at the door?

What if you could get rid of three occurrences of the first type of mistake at the cost of adding one of the second type? Would that be a better trade-off for the user? Would it be more helpful in achieving the systems objectives?

It will depend on the experience. How much does each type of mistake irritate the user? Which type of mistake is easier to hide? Which type of mistake gets in the way of achieving the system's objectives more? And how many times per day/month/year will a user have to experience each one of these mistakes? Will they become fatigued and stop responding to alerts from the system at all? Or will they come to distrust the system and turn it off?

Other examples of trade-offs include:

- Trading off underestimates for overestimates. For example, when the smart toaster predicts how long to toast for, would it be better to over-toast by 5% or to under-toast by 5%?
- Trading off the number of options the intelligence gives. For example, in a recommender system, the system might be pretty sure the user wants to read one of 15 different books. Would it be better to let the user choose from the top 5? Or to display all 15?
- Trading off between the intelligence saying something and saying nothing. For example, in a computer vision system, the intelligence might be 90% sure it knows who is in an image. Would it be better to have the intelligence label a face in an image with a name that is going to be wrong 1 out of 10 times? Or would it better to have the system do nothing? Or ask for the user to help?

Intelligence Changes

Traditional user experiences are deterministic. They do the same thing in response to the same command. For example, when you right click on a file, a menu comes up, every time (unless there is a bug). Deterministic is good, for many reasons.

Intelligent Systems improve themselves over time. An action that did one thing yesterday might do a different thing today. And that might be a very good thing. Consider:

- Yesterday you searched for your favorite band and got some OK articles. Today you search for your favorite band again and you get newer, better information.
- Yesterday you cooked some popcorn in your microwave and a few of the pieces burned. Today you cook popcorn and the microwave doesn't cook it quite so long, so the popcorn comes out perfectly.
- Yesterday your navigation app directed you drive through a construction zone and your commute took an hour longer than usual. Today the app directs you on an alternate route and you arrive at work happier and more productive.

These are examples of positive change, and these types of change are the goal for Intelligent Systems—from the user's perspective the system works better today than it did yesterday. If something is a bit off right now, that's no problem—the user can trust that the system will improve and that problems will decrease over time.

That's the goal. But change can be disruptive too. Even change that is positive in the long run can leave users feeling confused and out of control in the short term. Imagine the following:

- Yesterday you were taking notes with a smart-pen, and the ink was a nice shade of blue. But overnight the pen connected to its service and the service—optimizing for something you may or may not ever know—decided to tweak the shade of blue. Today the notes you take are in a different color.
- Yesterday you went to the local convenience store and bought a big, cool cup of your favorite soda, drank and enjoyed it. Overnight the soda machine connected to its service and tweaked the recipe. Today you bought another cup of the soda and you don't like is quite as much.

Are these changes good? Maybe. Maybe they improve some very important properties of the businesses involved. Maybe you'll realize you actually do like the changed behavior—eventually—even though it bothered you at first.

Or maybe the changes are simply disastrous.

When dealing with changing intelligence, some things to consider are:

1. Can the **rate of change be controlled**? This might be done by putting some constraints on the intelligence producers. For example, maybe the intelligence changes its predictions on no more than 5% of interactions per day. Taken too far, this could result in static intelligence (and eliminate much of the potential of intelligence improving over time), but some simple constraints can help a lot.
2. Can the experience **help the user navigate through change?** Perhaps it does this by letting them know a change has happened. Or maybe by keeping the old behavior but offering the user the changed behavior as an option. For example, the smart-pen could let the user know it has found a new shade of ink that makes notes 5% more useful, and offer to make the change if the user wants or keep the ink the old way if the user prefers that.
3. Can the experience **limit the impact of the changing portions** of the system, while still allowing them to have enough prominence to achieve impact? For example, by using a more passive experience with the parts of the intelligence that are changing the most?

The Human Factor

Intelligent experiences succeed by meshing with their users in positive ways, making users happier, more efficient, and helping them act in more productive ways (or ways that better align with positive business outcomes).

But dealing with Intelligent Systems can be stressful for some users, by challenging expectations.

One way to think about it is this.

Humans deal with tools, like saws, books, cars, objects. These things behave in predictable ways. We've evolved over a long time to understand them, to count on them, to know what to expect out of them. Sometimes they break, but that's rare. Mostly they are what they are; we learn to use them, and then stop thinking so much about them.

Tools become, in some ways, parts of ourselves, allowing us powers we wouldn't have without them.

They can make us feel very good, safe, and comfortable.

Intelligent Systems aren't like this, exactly.

Intelligent Systems make mistakes. They change their "minds." They take very subtle factors into consideration in deciding to act. Sometimes they won't do the same thing twice in a row, even though a user can't tell that anything is different. Sometimes they even have their own motivations that aren't quite aligned with their user's motivations.

Interacting with Intelligent Systems can seem more like a human relationship than like using a tool. Here are some ways this can affect users:

- **Confusion**: When the Intelligent System acts in strange ways or makes mistakes, users will be confused. They might want (or need) to invest some thought and energy to understanding what is going on.
- **Distrust**: When the Intelligent System influences user actions will the user like it or not? For example, a system might magically make the user's life better, or it might nag them to do things, particularly things the user feels are putting others' interests above theirs (such as by showing them ads).
- **Lack of Confidence**: Does the user trust the system enough to let it do its thing or does the user come to believe the system is ineffective, always trying to be helpful, but always doing it wrong?
- **Fatigue**: When the system demands user attention, is it using it well, or is asking too much of the user? Users are good at ignoring things they don't like.
- **Creep Factor**: Will the interactions make the user feel uncomfortable? Maybe the system knows them too well. Maybe it makes them do things they don't want to do, or post information they feel is private to public forums. If a smart TV sees a couple getting familiar on the couch, it could lower the lights and play some Barry White music—but should it?

Be conscious of how users feel about the intelligent experience. Manage their feeling by making good choices about what to present and how forceful to be.

Summary

Intelligence makes mistakes. No matter what, it will make mistakes. Complaining won't help. The mistakes aren't there because someone else is bad at their job. And chasing mistakes, playing whack-a-mole, is often (usually) detrimental.

The mistakes an intelligence makes can be quite counter intuitive. Even intelligence that is usually right can make really strange mistakes (ones that no human would ever make).

There are different types of mistakes. One common way to express them is as: False positives (saying something happened when it didn't) and false negatives (saying something didn't happen when it did). Intelligence can often make trade-offs between these types of mistakes (for example, by adding a few more false positives to get rid of a few false negatives).

Change can be disruptive to users, even when the change is positive in the long run. Intelligent Systems change, regularly. Helping users deal with the change is part of creating an effective Intelligent System.

Intelligent Systems can have negative effects on users, leading them to dislike and distrust the system.

Creating intelligent experiences is hard.

For Thought…

After reading this chapter you should:

- Understand that intelligence makes mistakes—it is one of the fundamental challenges that intelligent experiences need to address.
- Realize why these challenges are not intelligence bugs (which should be fixed) but are intrinsic to working with intelligence (and must be embraced).
- Have a sense of how users will experience the negative parts of an Intelligent System and some empathy to help create experiences to help users.

You should be able to answer questions like:

- What is the worst mistake you've seen an Intelligent System make?
- What could an intelligent experience have done to make the mistake less bad?
- For what product change did you have the wildest reaction, disliking it at first, but coming to like it more over time? Why?