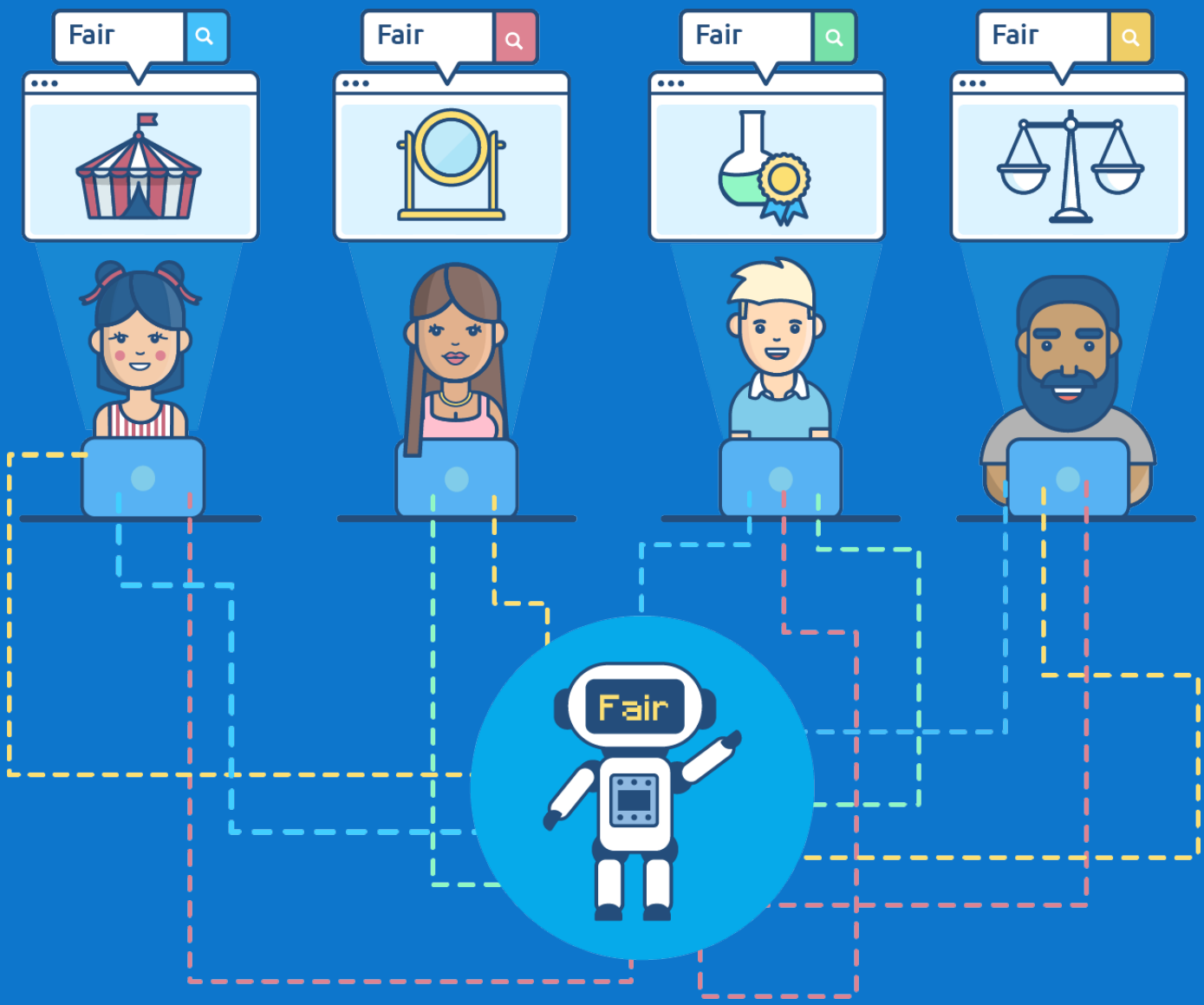
test.ai

# Testing AI and Bias

Jason Arbon



Jason Arbon
Testing AI and Bias

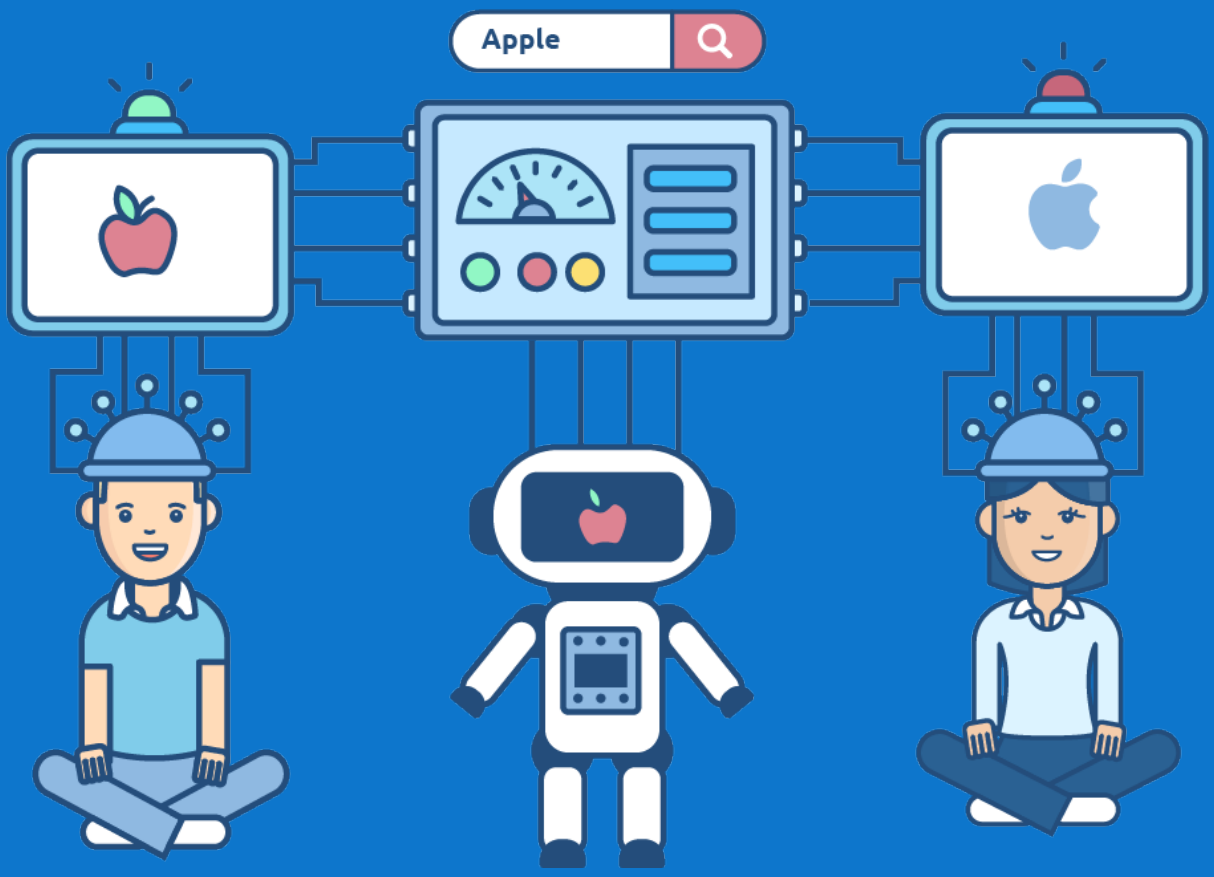

Published by: test.ai
Edited by: Dave Kong
Art Direction by: Kanika Rehani
Book Design and Illustrations by: Emily Eder
Distributed by: test.ai

Bias: “prejudice in favor of or against one thing, person, or group compared with another, usually in a way considered to be unfair.”

— Google

# Testing AI and Bias

Jason Arbon




# INTRODUCTION

As our world increasingly depends on AI-based software, our engineering focus needs to broaden from avoiding bugs to understanding bias. These AI systems are already controlling our cars, diagnosing diseases, influencing the legal system and the information we are exposed to. Biases in these critical systems can not only produce incorrect results, but can also make unethical decisions. We can't blame the AI because it is us who are training and enabling these biases. In this new world of software engineering, we have engineers who don't truly understand bias on one side, and on another we have testers who don't truly understand machine learning. In short, we need to understand these issues and add new tools and techniques to ensure a safe and ethical world.

In this discussion, we will investigate bias in the software, specifically AI systems. We'll walk through where biases can appear and how to minimize unwanted bias across the following major stages of building these AI-powered apps:

**Stage 1: Training Data**

**Stage 2: Training Processes**

**Stage 3: Productization**

We'll use real world examples from web search engines to illustrate bias. We picked these examples because most people are familiar with what search engines do and most people agree that they shouldn't be biased. I also happen to have intimate experience testing the world's most well-known search engines and will share some of those stories.

## SPOILER ALERT: IT'S NOT POSSIBLE TO ELIMINATE BIAS

If you remember just one thing, remember that it's not possible to eliminate bias. There will always be bias in an AI system. This is because there is always bias in the training data that AI learns from. Most people want to eliminate all bias. However, that is not possible and is also unnecessary because not all bias is bad. The term bias is typically used to suggest some AI system is negatively biased against a particular group of people. As we'll see, AI can also be biased in constructive ways toward different groups of people or situations.

You may wonder if there is truly such a thing as an unbiased training set. An unbiased training set would be random noise where AI would learn nothing. In other words, you can't train an AI system with random noise. The moment a human or machine selects data to train with, bias has entered the picture. Even if the human is ethical, intends well, and is super smart, there are still many sources of bias that are introduced at every point along the way.

## AI CRASH COURSE

**Not Familiar with AI?**

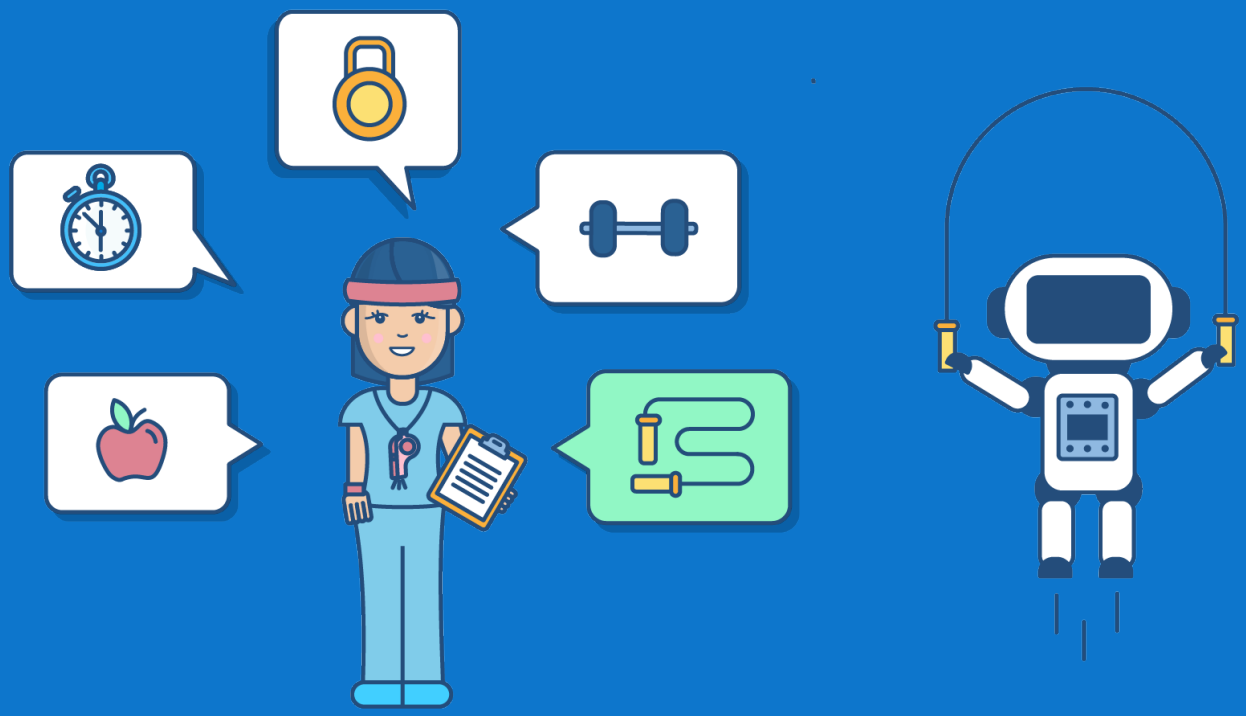
If you're not familiar with AI, think of AI as a student at school. Much like a teacher selects the books and tests used to teach students, AI engineers select the training data and models to teach AI systems. But unlike actual students, AI will learn and take a test hundreds (or even millions) of times until it gets it 'right'. With enough data, the AI will almost always get it right on average.

So, if our training data is correct and ethical, will AI be correct and ethical too? Nope. If our training data is 'bad', will AI also be 'bad'? Yes. Even the most well-intentioned teachers and engineers have their own biases, and it leaves an impression on the student's and AI's brains.

In practice, there are three major types of AI:

- **Supervised learning** - Supervised learning closely matches the teacher/student model, as the teacher is supervising the training material and testing process. The vast majority of commercial deployments are supervised learning.
- **Reinforcement learning** - Reinforcement learning is more like the reward and punishment used to train a new puppy. You can easily extend the lessons here to the rewards of reinforcement learning. By definition, the bias in training data (or rewards) will result in bias in the AI system.
- **Unsupervised learning** - Unsupervised learning is a branch of AI where the machine organizes, or clusters, information based on its similarity without using training data. This may sound like a less biased way to build AI, but the selection of the data to train on, and even the interpretation of the data still invite many opportunities for bias to creep in.

All three major branches of AI and machine learning suffer from many of the biases that we will discuss in this document.

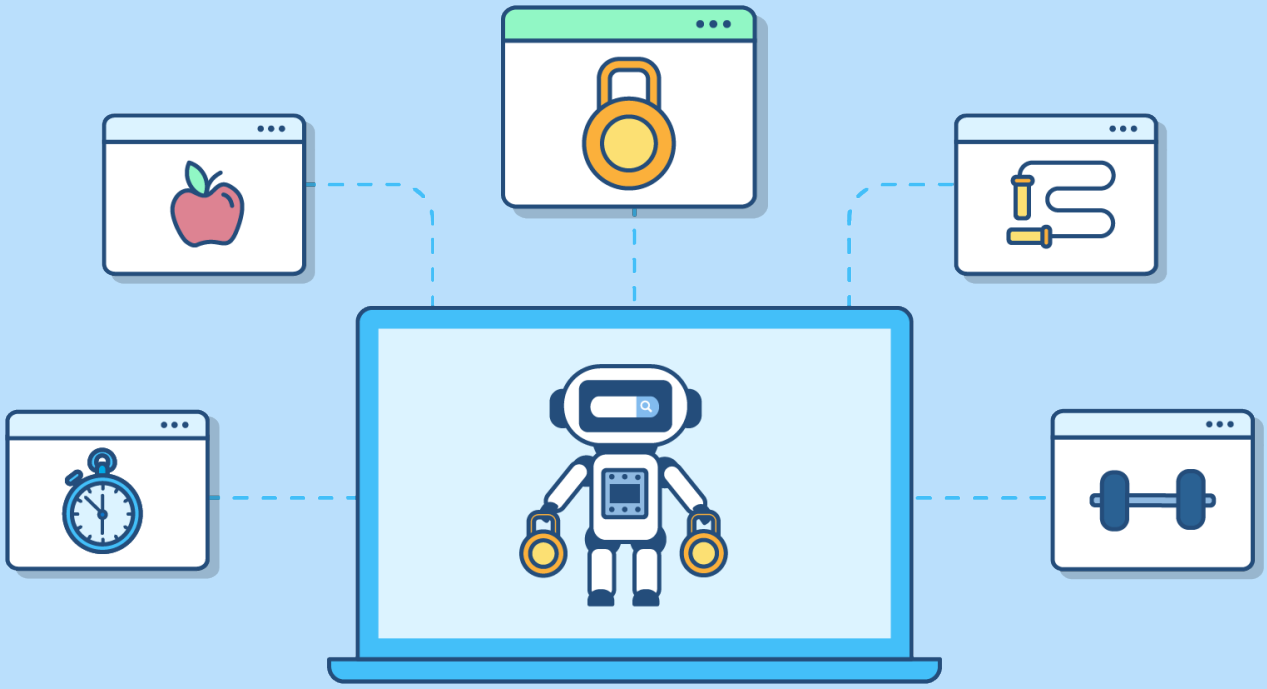


# Stage 1 : TRAINING DATA

The first stage in building your AI-powered app is acquiring the right training data. This is actually a group of steps that involve:

- Sourcing data
- Sampling data
- Labeling data
- Defining training and testing sets

Sourcing data refers to acquiring the dataset to train your app on. Sampling data means picking which specific examples from the dataset to train on. Labeling is the process of applying tags to sample data for use in machine learning models. Defining training sets refers to identifying which labeled data samples will be used for training, testing and validation. Even at these early stages, bias is already introduced in each of these steps.



# BIAS IN SOURCING

One of the biggest challenges in training an AI system is sourcing data. Imagine that you are tasked with building an AI-powered search engine. There are two types of data needed to train an AI-based search engine: search queries and websites. Our AI needs to learn how to match search queries to the best websites. A logical way to get website data is to build a web crawler that indexes as many pages as it can find on the web. It's here that we see our first bias challenge. Web crawlers find websites by starting at one site and then following all the links on the page, and then every link on those linked pages, and so on. We might be able to get millions of web pages indexed, but ask yourself:

**Where did the web crawler start?**

The web crawlers probably started on popular, top websites. Given that the web crawlers only find millions of the possible billions of websites, there is now a bias in the dataset towards top web sites. Many of those unpopular, unlinked websites that weren't found are left

out of the dataset, which means the AI can't learn from them. The resulting AI-based search engine will only be learning from the most well-formed and well-written websites. The search engine will not be learning how to deal with less popular and possibly poorly written, mis-spelled pages with broken or malformed HTML code in them.

**But what if we had all the web pages?**

Can we address this bias by applying more data? Many engineers are tempted to build a better web crawler to find all those tiny websites. That could relieve some bias, but there are a few problems with this approach. By definition, you can't actually know if you have crawled the entire web, because you don't know what you don't know. If you did, you'd know about it. There are also technical reasons why you can't actually crawl all websites. For example, many web pages are behind a login, or a paywall, or on private corporate or government web servers, not linked at all, or will even tell the web crawler not to view them. Therefore, our dataset is only going to be comprised of publically available, accessible web sites.

Even with a truly complete dataset, you cannot eliminate all bias. There are socio-economic and demographic factors that are inherently part of all data. If we magically did have all of the world's web pages, you'll still find that most websites are written in English. That the richest countries produce more web pages per-capita. That democratic countries produce more web pages. Even if every demographic bias was somehow adjusted for, the AI learns from the average of all the data. As most humans aren't average, bias creeps in again because the outliers don't get much of a vote in how the search engine AI will be trained.

While it may seem that we're attempting to solve an impossible problem, our job is to acknowledge the bias and understand it. We need to ask ourselves: Is this an OK form of bias? and do we understand how bias will manifest itself in our app?

You might say that this bias is OK because the best results will probably come from top web pages. Or you might think this is concerning because the search engine will have a bias for commercial sites and a bias against sites from underfunded, non-technologists. Do we deliberately seek out these less-connected web sites? How do we know when we have enough of them? Determining if these biases are acceptable, or even preferable, is actually up to you.

## MSN IN THE 90s

In my previous life, I had the opportunity and privilege to contribute to Microsoft's MSN search engine. This was one of the early search engines born in the 1990s. At MSN, we had a dominant position as the default search engine in most of machines around the world. With this market position, you would think we'd have a lot of great training data. But since you have read this far, I'm sure you already know that is likely a wrong assumption and you are already starting to see problems.

Where are all those search queries coming from? If we think about it, we might think the bias is OK because our operating system is the dominant desktop operating system so our query samples are coming from all sorts of different demographics and locations. Wrong. The people who were using MSN Search back then were people that either weren't aware of Google, didn't know how to change their search engine default, or as we found out later, thought MSN search results were actually Google search results. (At that time, almost every engineer at Microsoft was using Google as their default search engine.) Given that our queries were coming from these people, there was a new bias we are feeding to the AI. The bias was toward lower-income, lower-education, located in the midwest of the country. We were training AI to be great at search queries for this demographic, but not other demographics such as engineers, doctors, lawyers, etc.

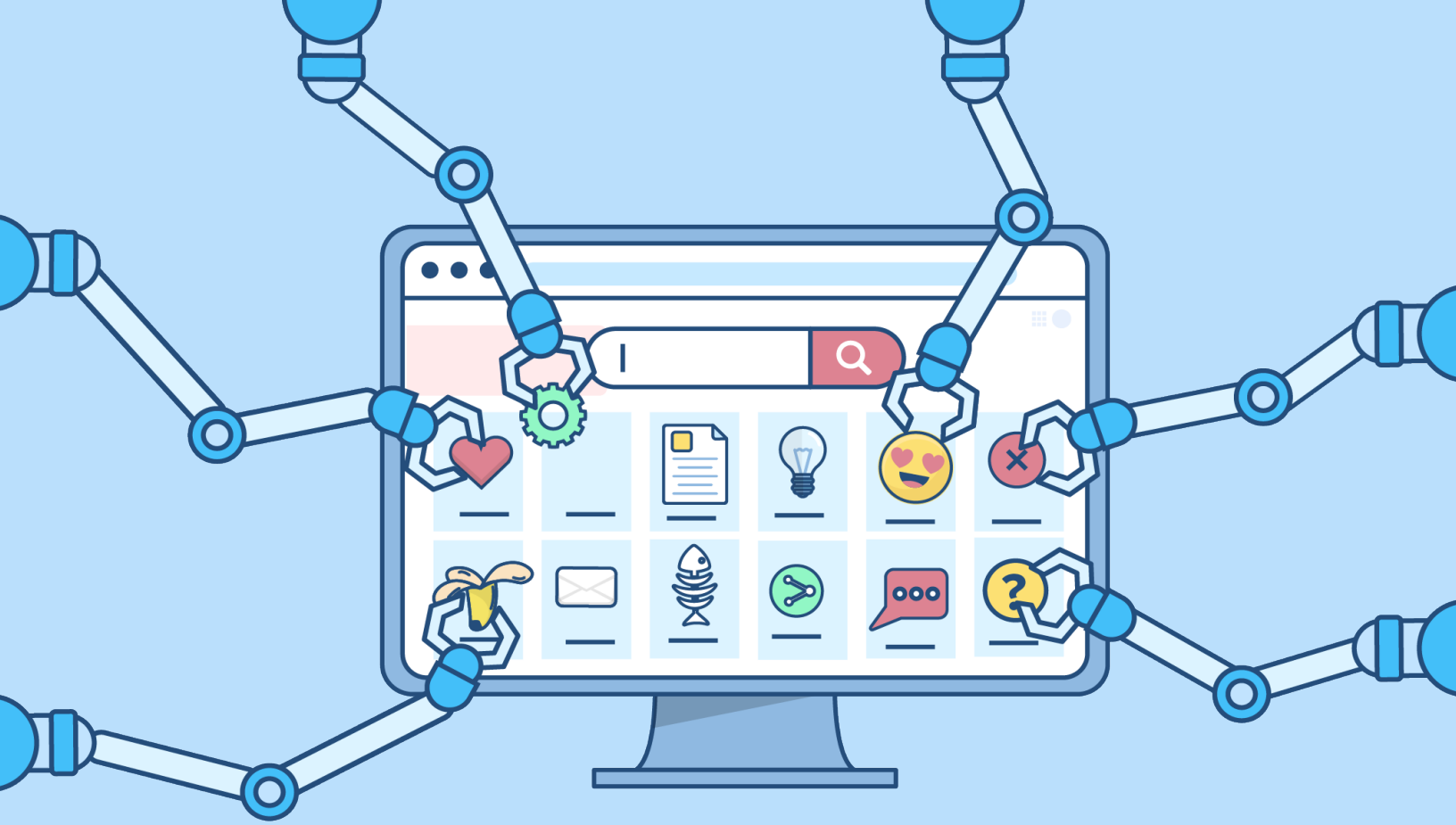
We might look at this and say one of two things. One, this bias will deliver subpar results to knowledge workers and this is a bad bias. Or two, this bias is OK because we are making a search engine positively biased toward the very people that are using our search engine, and therefore are more likely to be able to monetize queries. There is a bias, but determining if it is good or bad is sometimes also a business decision.

## MYTH:

### I SHOULD SOURCE PRODUCTION DATA

Machine learning engineers often want to train on the data gathered in production, but this can introduce additional unintended bias and should only be done with great care. In the world of search engines, an example of this is click-through data. Most search engineers and average search engine users believe that you can simply rate search results higher if they are clicked more often by the search engine users. There are a few problems with this. One, this data is biased by your user demographics. More importantly though, humans semi-trust the search result ordering, so they are far more likely to click an irrelevant search result at the top versus the best result if it occurs at position six.

There are ways to try and get around this positional bias problem, but they are fraught with bias themselves and can be very expensive to run. A naive way to do this is to have many different flights so that your users get the results in a randomized order to remove the positional bias. This means we have deliberately added random bias to the results for many of our production users. Many queries (some 15%) are unique and many more have only been executed a few times. Training with this clickthrough data biases the engine for doing better on frequent queries and likely worse on infrequent queries. It is tempting for engineers to use their production data, but it needs to be done carefully to avoid inadvertent bias loops in the training and productization process.



# BIAS IN SAMPLING

Sampling is the process of picking examples in your dataset to train on. There are many different ways to sample data, and of course, many different ways bias can creep in.

Say we needed 50,000 queries to train the AI (any more doesn't improve the system, takes more time to train, and reaches a point of diminishing returns). As a search engine, you have millions of searches happening each day. What if we picked a single point in time and took the last 50,000 search queries? What bias have we introduced? If 50,000 search queries happen every hour, and we took our sample of queries between 4pm and 5pm Pacific Time from a random machine in production, we have now introduced a bias based on time of day. People generally search differently (e.g. technical searches) when they are at work versus on the way home from work (e.g. restaurant searches). Since we also picked a single machine, albeit at random, individual machines are often serving different regions of the country, so we also just introduced bias based on location.

To avoid building a search engine that is biased toward time of day, or location, we could take a random sample of queries from a random set of machines in production. From our earlier discussion, we know that this will still be biased based on search frequencies and higher population density regions, but maybe that is OK because our user base also follows these biases. But, we still have a day-of-week bias. If we take the sample on Monday, a weekday, the Monday-based trained-AI probably won't fare as well on the weekend. So we sample from a whole week of data. But was that week a holiday? Was that week in the winter or summer? Was there a major world news event, such as an election?

## SAMPLE RANDOMLY

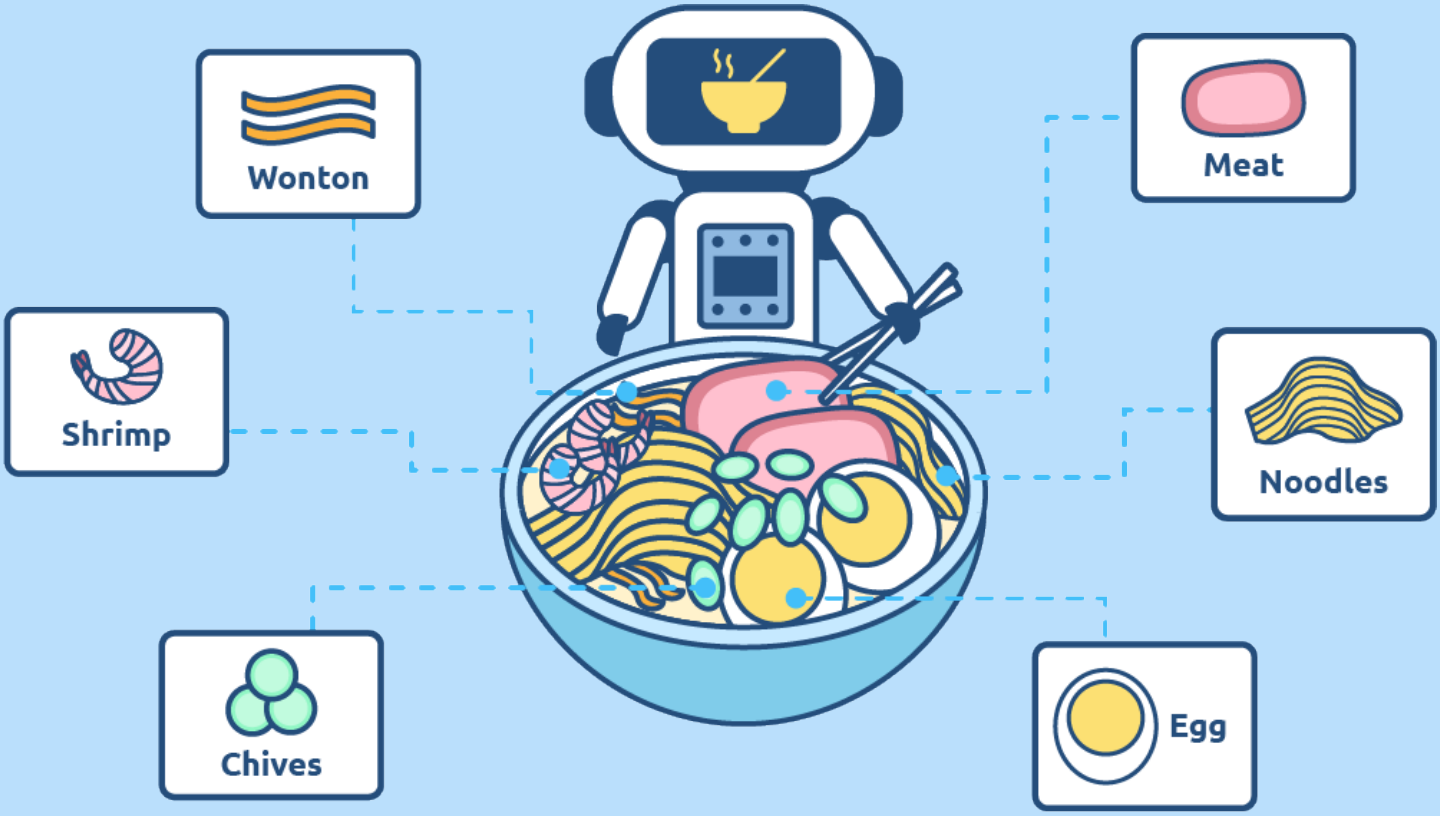
One solution to this problem is to sample randomly. Utilize data from the most recent hour, and mixing in fewer samples from previous times such as last week, even fewer from last month and even last year. That way our sample is at least sprinkled with time-varying bias, with a recency bias, and will probably work well in general, on the average, and not be what is called over-fit to a particular time. That said, the resulting search engine won't be the best weekday search engine, but on average it will be as good as it can be.

## SAMPLE SIZES

How do you know if you have enough data? How do you know how much more you need? This question is often very domain-specific. The amount of data required varies widely between different machine learning algorithms. Some are data hungry like deep neural networks while others like reinforcement learning may only need a few examples. Some problem domains have very noisy data, or even data that contradicts itself (aka 'disagreement', 'confusion' or 'entropy') in the training data. Generally, the more variety in possible training values and the more confusion there is in the dataset, the more data is needed to train a great AI, and avoid a poorly biased system. For example, if you were teaching a student to identify pictures of cats vs. dogs, you might need a lot of examples as there are a wide variety of dogs and cats. But if you were training an AI to discriminate between a picture of a night sky (mostly black with white stars) and a sky during the day (mostly blue/white), you may not need as many training examples.

For a very simple example of bias due to sample sizing, consider a political poll that only asked 50 people for their opinion. It is very likely several people from large demographics will be included in the poll. But what about the opinion of people that represent 1% of the population? The odds of their opinion being represented in the summary poll can be less than 50%. And, what if we did manage to catch that 1% in the poll? They would be overrepresented in the polling data, being 1/50th of the people polled, which is 2%. Their voice would have twice the weight it should have. To understand the right sample size, the data engineer needs to understand the texture of the data, the underlying differences and frequency of those differences in the data and ensure the sample size is large enough to avoid bias in the sampling.

Remember, the AI algorithms have both the training and test data, and can learn and take the test thousands of times during training. The AI training process is continually testing the AI's output against the expected output (test or validation dataset), so most AI training processes can tell you precisely how accurate the model is given the training/test and validation data. In addition to giving a % correct, these training systems often also provide a confidence interval and other measures of quality such as a confusion score, records of false-positives and false-negatives, and precision and recall. Often, teams will add more data (or change hyperparameters) to the training process until they see diminishing returns on these scores. For your given application, you need to balance the time it takes to train, the amount of data, and these training results to fit your engineering goals.




# BIAS IN LABELING

Labeling is the process of getting humans, or raters, to look at some output and describe what they see. After applying labels to your dataset, you can apply labeled data to machine learning models for training your AI system. It may sound pretty simple in principle, but it is actually far from it in practice.

In labeling, there are two major areas where bias is found:

1. In the labeling process
2. In the raters

**Bias in the Labeling Process**

In our search engine example, when labeling, the data you would show raters are search queries and possible search results. We'd then ask the raters to rate how good that result is on a scale of 1-10. Let's look at the guidelines Google gives to these people today: https://static.googleusercontent.com/media/guidelines.raterhub.com/en//searchqualityevaluatorguidelines.pdf.

The guidelines themselves introduce some intentional and possibly unintentional bias. Here are just three biases introduced by the rating instructions:

- **Ads-based Bias** - The guidelines encourage raters to discount web pages that have distracting ads. This biases raters to favor websites by entities like governments or large corporations who don't need to advertise to pay for their costs. The person who has the best content or answer may not be rated highly because they have to have a giant banner ad to pay for their web hosting.

- **Business-based Bias** - You could also even read some accidental business bias into these guidelines. Google, which is a web advertising business, wouldn't serve annoying ads based on their definition, so sites that have Google-powered ads will likely receive a few more page views and ad clicks and that means more money for Google. Whether intentional or not, this type of bias can lead to business and perception issues.

- **Authority-based Bias** - The guidelines instruct raters to prefer websites that are authoritative and from experts. If there are two web pages written about a political topic, one written by a professor, and the other by someone who has first hand knowledge on the topic who is not a professor, Google is essentially asking raters to favor the professor. The rater may interpret anything as expertise. Perceived expertise might be the fact that the name of the author of the page is well-known, or in their community, or agrees with their position on a topic. Is this bias a good thing in all cases? What about in some cases?

Companies rarely have the money and time to even think about how to test for bias in the labeling process, but it can be done. A great tester and bias investigator will want to see search ratings from people that have or have not received different versions of these instructions and measure how the results of the AI-based search engine performs. The search quality engineer should inspect which search results are affected by different clauses in the instructions, document the variation, and judge if this is a good or bad bias. You can define what a good or bad bias is in terms of the impact on the business, customers, users, and even social impact. Applied more broadly, exceptional testers will understand the biases inherent in their process and make a judgement on the bias.

**Entropy**

As you enlist the help of multiple raters, you'll quickly realize that people rarely agree. If you ask the same labeling question to multiple people, you will get multiple answers. One person's best search result may be another's worst. This is called entropy in the training dataset. The AI wants to learn the best answer, but how can it do that if the training dataset conflicts with itself? This is akin to a teacher giving textbooks that have conflicting versions of history and facts. Rather than risk getting just one person's opinion, data scientists will often generate overlap, which is simply asking more than one person the same question to see how much they agree or disagree.

**"Bush"**

Let's look at an example I investigated in the past: the search term "bush". If a rater enjoys gardening in the mornings, the rater may justifiably think it obvious that a website describing shrubbery is the best result. If you are over a certain age, have a political bent, or have heard of recent news about one of the two former U.S. presidents, then you might think their wikipedia pages, or a news article, is the best search result. If you are a fan of alternative music, the website or a music video of a band called "Bush" is the best search result. With this single and simple query, we could get wildly different answers if we asked a few random people what the best website results should be. This happens to be a very common problem when

labeling data for training AI systems. Knowing that our training data isn't consistent, there are several bias and correctness issues we need to tackle.

To mitigate bias in the labeling process, we need to:

1. Quantify Entropy
    a. Identify Areas with the Most Variance
    b. Identify Areas with the Least Variance

2. Increase Overlap Where Needed

The first step is to quantify the amount of entropy in the system. We want to understand the areas with the most variance, the least variance, and the distribution. If we don't take this step and decide to pass this data to the AI training system, the AI will be unlikely to get 100% of the tests correct because it will generally average the results. With this step, let's take a look at:

- **Areas with the Most Variance** - In our search engine example, we want to test the queries with the most variance in ratings. If these differences are understandable, have we captured a representative sample of these ratings? On an ambiguous query like "bush", with an overlap of 3, and all three have given us different answers, we need to increase the overlap. Increasing overlap is simply asking more people to rate the results. Maybe there are other interpretations of the best results for "bush" and we haven't found them yet, and that bias is missing in our training set. Maybe there are really only three major opinions on the best results for "bush" and we were lucky to catch all three of them by asking three people at random. But, if we train now, the AI will learn the average of these opinions and equally weight the three opinions. That may sound like a reasonable bias, but with only three ratings, we don't know the relative ratio of biases. If seven out of ten raters think the best results are websites with plants on them, our training data doesn't represent that. We need to get far more overlap in ratings to capture the entropy for this query. You can add more and more overlap until the relative ratios converge to a value. The best mathematical result from this is training data that is biased to popularly held beliefs, and suppresses the outliers. But, is this the correct bias?

- **Areas with the Least Variance** - Counterintuitively, you also need to test the queries with the least variance in ratings. On simple, navigational queries such as “CNN”, almost every human rater believes https://www.cnn.com is the best answer. You would think we don’t need a whole lot of overlap, but it’s quite the opposite. Our search engine is going to show at least ten web results for this query. Figuring out what the second and third best results for a query like this requires more overlap to discover alternative and minority results such as those for ‘Convolutional Neural Networks’, a search result only nerds reading this would appreciate. This is another form of bias and whether we address these issues or not is our decision.

Our task is to apply the necessary overlap needed to mitigate entropy. Using benchmarks such as relative ratios are good indicators of how close or far you are from an ideal dataset. But because this is often very costly, knowing when enough overlap is enough becomes a balancing act of risk in the bias and the cost of the human labeling data program which can run into the many tens of millions of dollars each year. Since we live in a world with scarce resources, we need to balance the amount of overlap needed versus its financial cost.

# MYTH:

## I SHOULD CLEAN TRAINING DATA

All too often, well-intentioned data scientists try to clean up the training data thinking good data in, good data out. Cleaning up training data is not only another point where bias is introduced, but it is almost always it is a bad idea. Here are some examples:

- **Misspellings** - If you look through search queries and find some that are obviously misspelled such as 'buush', or 'CNNN'. Most engineers' instinct is to remove these from the training set, or fix the spelling because it might confuse the AI--much like a teacher is likely to hide errata from a textbook so as not to confuse the students. But, in the real world, search queries are often misspelled, and more interestingly, certain types of misspellings are shared across different query strings. Leaving these queries out of the training set introduces a bias toward searchers who spell correctly, making it more difficult for people with lazy typing, learning disabilities, or poor educational backgrounds to find information on the web.

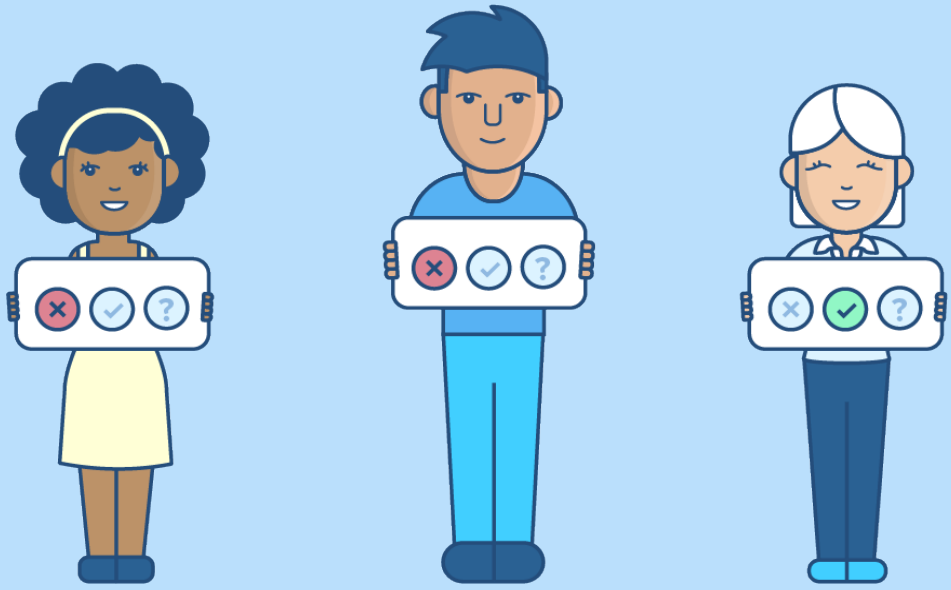
- **Weird / Rare Data** - With any form of sampling, strange samples will inevitably show up in the training data. A real-world example is the search for a GUID. It doesn't much matter what that is, but the point is that it can look like garbage data and be tempting to remove from the training set. For example, the query "{72631e54-78a4-11d0-bcf7-00aa00b7b32a}" is a deliberately unique string. If we throw out this example query, we don't provide the AI the data it needs to learn how to rank similar long strings of similar letters and numbers. To some folks, this particular GUID string may be super valuable (computer engineers working on battery systems in this case). If the data looks odd, there is probably a good reason for it; leave it in the training set.

- **Scientist's Bias** - It is always tempting for a Data Scientist/Engineer to fix up a bad

result. If the training data is 'obviously' wrong, they want to fix it. But, their idea of 'obviously wrong' now introduces more bias, i.e. the bias of this individual scientist who probably has a different view of the world than most humans that will use the AI in production.

- E**xcluding Raters** - Tendencies to clean or scrub the training data are most dangerous when they are automated or systematized and invisible to the data science team. The task of gathering labels/ratings is often passed to an external vendor and not performed by the data scientist. Vendors will often try to clean up the data to make it look 'good'. This data cleansing can take many forms, from removal of outlier data to the systematic firing of raters who constantly disagree with their peers. Think about your training sets, even publically available sets--do you know what data was excluded? Perhaps a strange looking cat was excluded from the dataset, or the label of what is obviously a dog (but is really a cat) had its label fixed by someone who knew better. Or perhaps, semi-nefariously, someone wanted an improved AI by the metrics and got rid of some confusion from their training or test set. These hidden sources of bias are dangerous in that they are often never noticed. The mitigation here is obvious: control as much of your data sourcing as possible.

- **Social Pressure** - Imagine a traditional, functional software engineering company pivoting to data science and building search engines. You can imagine that the small, fragile, search engine team would get pressure from other parts of the organization to 'fix' bugs in the search results. For example, what if you searched for "best search engine" and your competitor showed up at the top of the search results? The AI might have learned that your competitor is a better search engine, but what does a fledgling data science team do when a senior VP complains? Of course, they hard code the correct answer and bypass the AI subsystem. This rarely works out because it reduces user trust in your system for other search queries. It is best not to introduce hard coded bias. The mitigation here is twofold. First, analyze the system code looking for places where hardcoded results might lie (especially in the rendering/UI system). Second, employ psychology. For example, make it easy for people to give feedback and thank them kindly for it. But, since this is just another form of bias, ignore this anecdotal data and stick to statistics and sampling.



# BIAS IN THE RATERS (DEMOGRAPHICS)

Perhaps the most dominant and overwhelming bias introduced into supervised learning systems is the bias inherent in the people doing all that labeling work. The judgements of the people creating all that labeled training data directly influences the resulting AI. The AI-based search engine is effectively trying to mimic the collective wisdom and reasoning of these human raters.

Raters are far from demographically diverse or representative of the users of the search engine. These raters are unsurprisingly all paid the same amount of money for their time. They are paid between $4 and $15 dollars an hour for their work. This means the labeled data is unlikely to match the thinking of a physician, engineer, retired or young person. Moreover, what demographics desire this type of work? Often they are single parents of multiple children working from home, living in the midwest. It is a great job because they can tend to their children and work their own hours while making a decent income. But, this means that a very specific demographic bias is training our AI-search engine.

It is often cost-prohibitive and sometimes impossible to get labeled data from the right

mix of demographics. It is difficult to imagine even Microsoft or Google being able to afford and find people from highly paid professions such as lawyers, engineers, or scientists willing to do such mundane work. Further, consider that people who cannot see cannot perform these tasks. The practicalities of labeling has significant bias of its own and needs to be understood.

Ironically, this labeler demographic bias may not always be a bad bias. If your search engine was primarily used by and designed to serve this demographic, it might be OK. In practice though, the business wants to attract and monetize more affluent demographics and the engineers want the search engine they work on to be the one they use on a daily basis.

It is also worth noting that this demographic bias in the human labeling may not be as concerning for different applications. In the training of an AI to discriminate cats versus dogs for example, a data scientist could argue that there would be little difference in the bias from the demographics not represented. Some applications might also not be interesting or very-low risk in practice, and not warrant concern.

There are several ways to mitigate the bias stemming from this demographic homogeneity. The most direct way is to systematically ensure that you include the demographic variety that is appropriate by hiring at great expense those underrepresented demographics to perform some labeling. If the labeling activity is infrequent, this might be feasible. At minimum, this dataset can be used to test the bias in the system. With data from different demographics, it is possible to check for the amount of disagreement within the larger training set. This can guide directed labeling efforts on the types of queries where there is significant demographic differences. This data can also be used to identify which demographics and queries the search engine does poorly on to understand the basic characteristics of the labeling bias.

# BIAS IN DEFINING TRAINING SETS

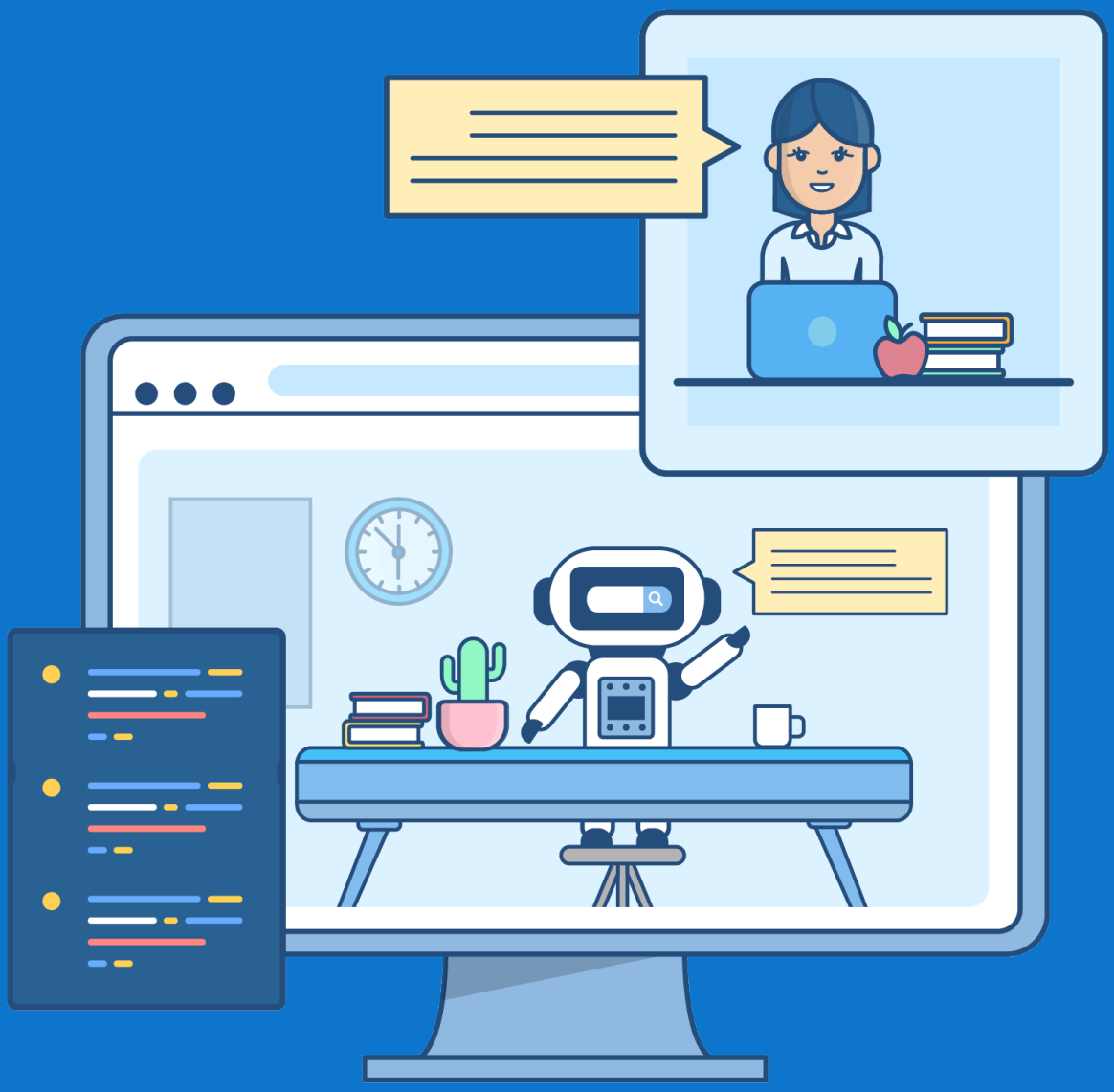
The concept of the test set is taking all the samples that you have to train the AI and splitting them into two different parts: the training set and the testing set. Often a third part called the validation set will be used outside of the training process for additional verification.

What happens during AI-training is that the training data is used much like a teacher's book and other materials to teach concepts. The test data is just what it sounds like, which is data that may have been shared in the teaching process but also tests how well the student does with new data or new concepts given their learnings from the training materials. If you have a lousy exam built from poor test data, you'll give A's to the wrong students. The integrity and bias of the test dataset is critical to the training of the AI system.

**Apply Random Sampling**

Typically the split between test and training sets is done by random sampling from the data. As with a teacher's tests, this means some test data may be shared with the training set. The importance of this testing is that the student (AI) has learned the concepts and not just memorized the answers. Random selection is interesting because it seems to minimize this bias, or at least is an attempt to do so. Generally speaking, it is best to make sure the test set

is as equally a random sample as the training set--it is OK for some test questions to also be in the training data. If queries for “Obama” are very popular, you do want them to also appear in the testing set, not just the training set. But it is important to test that the training and test data are not equivalent to ensure that the test data has a degree of independence from the training set that reflects the variation of queries expected in your production system.



# Stage 2 : TRAINING PROCESS

Now that you've acquired the right data, training is the second stage in building an AI-powered app. AI training systems are composed of many parts, including: feature detection, initial weights, hyperparameters, model updates and quantified precision and recall of the trained system. For our discussion, we'll focus on bias from features.

Generally speaking, machines can't actually see pictures of dogs and cats, let alone webpages. The AI training system can only see numbers in the range of 0 to 1. Training data must be converted into lists of these numbers that we call features. Features are the characteristics of a machine learning model that significantly influence its output. In the case of a web page, we might create features such as:

- # Queries in Document: Number of times the query string appears in the document
- % Queries in Document: Percent of words in the document that match the query string
- # Queries in Title: Number of times the query string appears in the Page Title
- % Queries in Title: Percent of words in the title that match the query
- % Spammy: Percent of raters that think the result might be spam

Engineers will write features that take the text of a web page and compute the corresponding training values for every training sample. The training process is then simply a loop:

1. Presenting these feature values to a randomly configured model
2. The AI guesses at the correct output value (score between 0-1)
3. That guess is compared to the raters assessment of how good that web result is given the query:
4. If the AI guesses incorrectly, it will reconfigure itself for the next try
5. If the AI gets guesses correctly, the AI configuration is reinforced as it is guessing well

After many loops the AI gets smarter and smarter at guessing through small, semi-random changes in the model.

These features were used because they are intuitively good indicators of the quality of the search result, but there are an almost infinite number of features we could have written. The very process of a human (or system in the case of AutoML) deciding which features to use introduces bias in that it decided what aspects of the data the AI can and cannot see.

An often overlooked form of bias can happen when the code that computes the feature values is numerically incorrect. Perhaps the algorithm misses the first word or last word on the page. Perhaps the math is off and the feature never produces a value greater than 0.00001--this would effectively turn off that feature. The functional code used to generate features for training needs to be extensively tested, much like traditional code. If the AI can't see correctly, it can bias the training of the AI in unexpected ways.

A machine learning engineer's job is to evaluate the quality of the resulting AI and iterate by creating or removing features to help the AI perform better. A well-intentioned machine learning engineer may notice that the AI is performing poorly on pages with pictures of flowers, as the pages have very similar page layouts and very similar looking images of flowers. So she creates a new feature called AveragePageColor(), which looks at all the pixels on the page and determines the average color, and normalizes this to 0-1 values for the training system. She then retrains the AI and gets a better score because the AI can now realize the difference between two flower web pages that differ almost only on the color of the image of the flower.

She has made a better search engine according to the mathematical analysis, but on deeper inspection she should measure which other queries were also affected by this change. It is easy to imagine the AI might have rated other pages better or worse because the AI is now also considering the color of the page, and those need to be inspected for unintentional bias. It is easy to imagine that giving the search engine the ability to see color could lead to additional unwanted, unethical or offensive bias as the AI-engine might now also be able to detect correlations in skin color, or hair color, or clothes on non-flower search queries.

Remember that we have demographic bias in the training data from raters. If those raters have their own bias, and the machine learning engineer writes features that expose the features the raters used for demographic bias, she is enabling the AI to easily reflect those

same biases. Interestingly, the inverse is also true. If the machine learning engineer considered adding a feature for color, but realized that feature might add the possibility of demographic bias, she might choose not to add the feature. The search engine might not be as good at discriminating flowers, but might also be less likely to have unwelcome bias elsewhere!

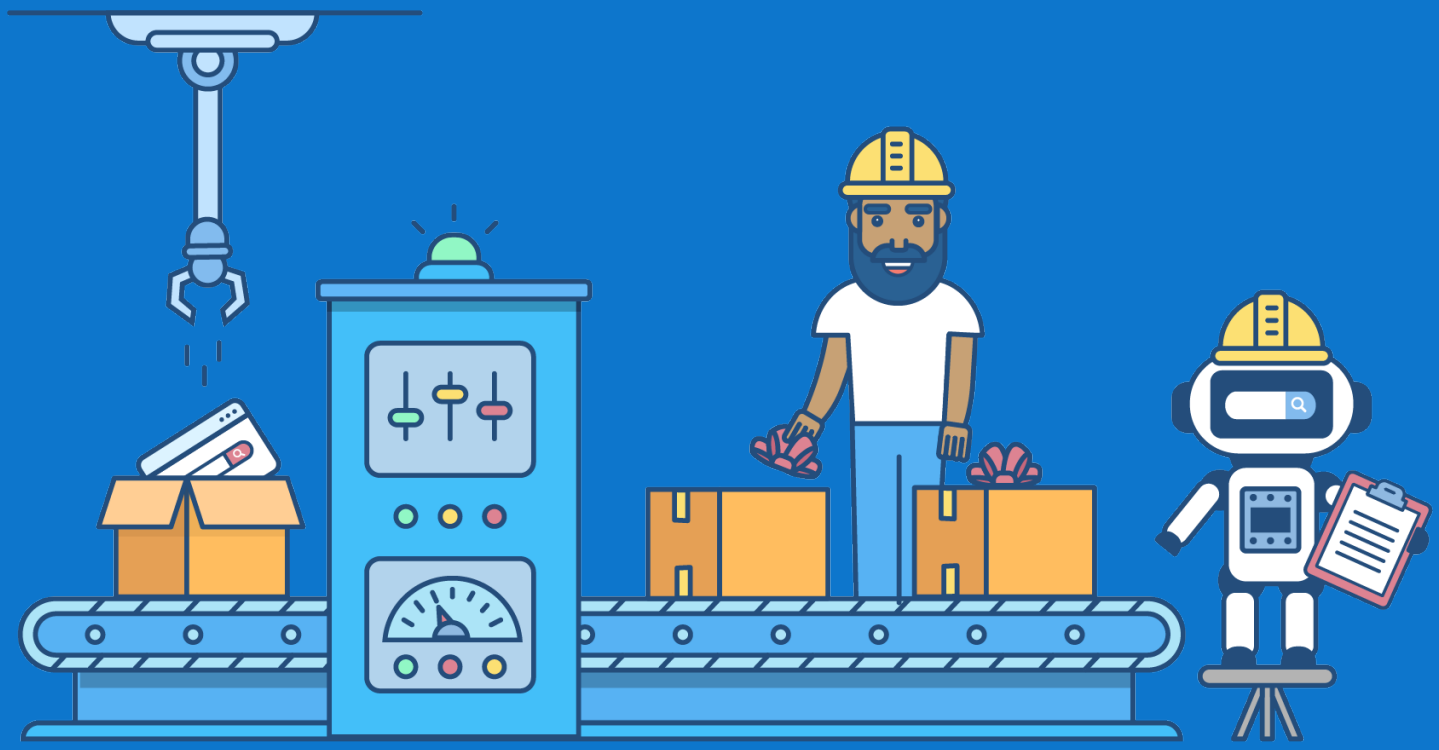
We can mitigate for this type of feature selection bias by:

- Considering the types of discriminations the feature may enable in the resulting AI
- Proactively analyzing the queries heavily influenced by specific features
- Proactively considering test queries that might reflect even the perception of unwanted bias
- Generating synthetic training data based on variations of existing examples for training to balance the data. This can be very dangerous and add its own bias, so be careful!

The training process also enables engineers to literally add their own initial bias, or weights, to the different input features. The machine learning engineers are continually trying to coax the AI to get better by messing around with all these little training bias hints. The hints reflect the intuition, judgement and bias of the engineer resulting in the AI having a heavy bias for color, or spamminess, or any other feature. Care should be taken when adding initial weights to feature inputs in training as the AI might get stuck in a local maximum that directly reflects unwanted bias from the engineer.

# A NOTE ON DRIFT

It is important to note that every new AI training session can result in a search engine with the same overall score but have very different characteristics and biases. This is often called drift. Due to inherent randomization in the training process, changes in the training data, training configuration, hyperparameters, etc, each new search engine with the same score might behave very differently on any given query. On average, the two AI's are equally good, but may be better or worse on different types of queries. Today's trained AI engine may be better at acronyms but worse at proper names versus yesterday, but on average it is the same relevance. Today's AI engine may also be better at getting the top result in the top spot, is far worse with results in position 4 and 5, but still have the same net score. For some AI-based applications, consistency may be incredibly important (e.g. medical, legal, financial) and might need a great deal of time spent on identifying drift in the sub-topical behavior of the system.



# Stage 3 : PRODUCTIZATION

The last stage in building an AI-powered app is productization. Ultimately, we build most AI systems for people like us. Whether it's a search engine, medical analytics tool, or autonomous driving system, these AI systems all try to produce the right data. But what really matters is how that data or those predictions are translated into real world user interfaces and actions. We need to understand the bias that exists in the end-to-end user experience.

In search engines, the user experience manifests in a list of search results in a ranked order. People see a list of ten blue links and read these results from top to bottom. We rarely look at results past the fourth one on the web page. On the other hand, machines (and often data scientists) can see all of the results. They can even see when the first result is 10x better than the second. They can see if the first and second results are virtually indiscernible. Some of that context and texture is lost when translating the underlying numbers output by the AI-based search ranker into a simple, clean user interface.

The underlying AI system is only optimized to score each individual website for a given query. The AI isn't smart enough to measure itself from the user's perspective. Rather than rely on the AI's own assessment of its correctness, we need a way to score how well the AI's output is perceived by the user.

**Quantifying End-to-end User Experience**

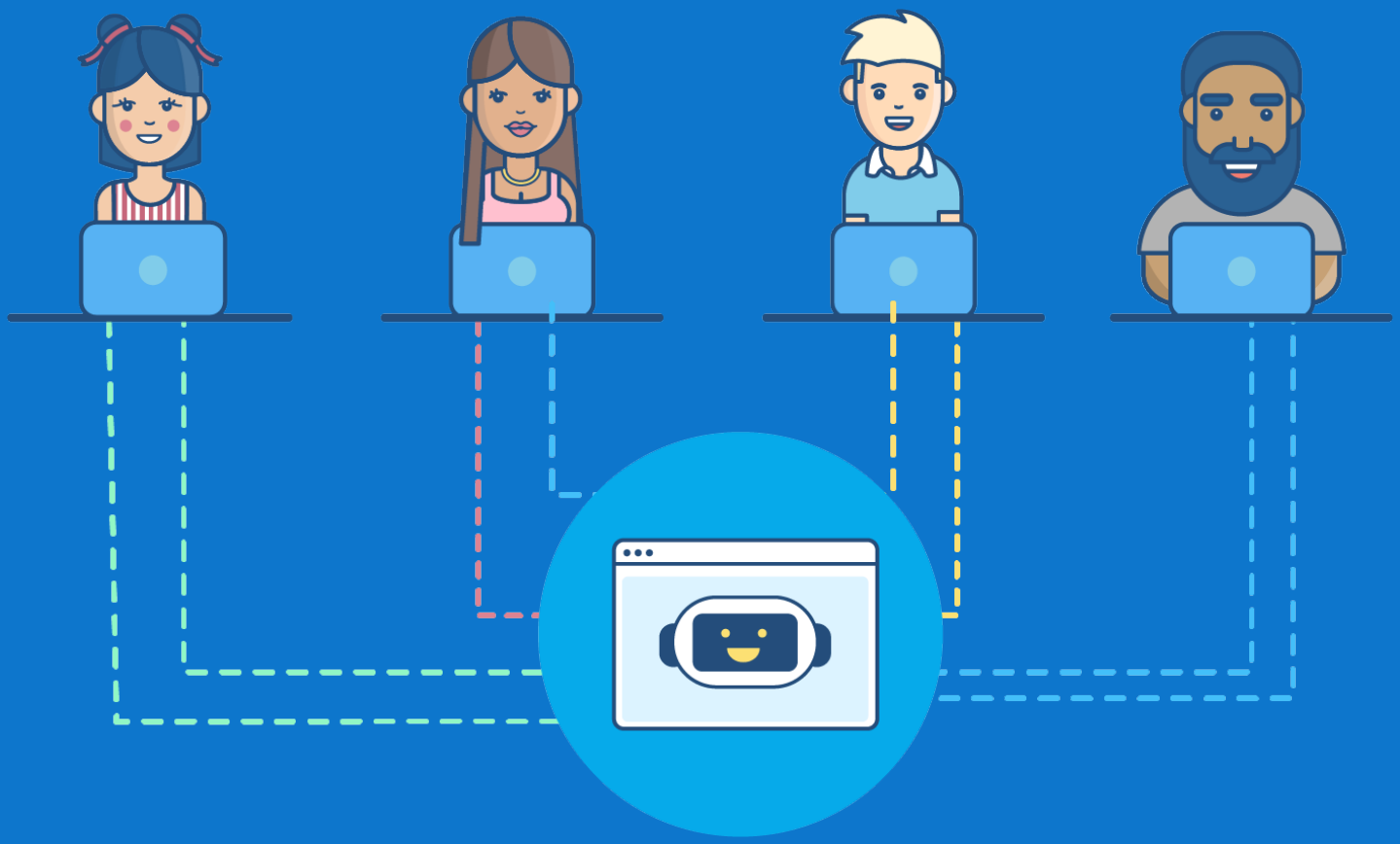
One simple approach to measure the end-to-end user experience of AI systems that rank (i.e. search engines) is called Normalized Discounted Cumulative Gain (NDCG). It sounds complicated, but for our search engine example, it is simply a mathematical way to quantify the perceived search engine quality to humans using the search engine's user interface. NDCG gives the search engine a score for a search query and set of search results that favors getting the best results at the top of the page. This measurement penalizes the search engine more for getting the top results in the second row than it would penalize the engine for getting the fourth best results in position ten. This encapsulates the understanding that if a good but not best result is in the top position, many users will never see the best link. It also encapsulates the notion that users rarely get to the results below position four or five. For AI systems, it is important to measure the relevance of the system as it is productized, deployed and factored into the human perception of relevance.

However, be wary that these productized measurements of bias can also add their own bias to the AI system. NDCG doesn't account well for the fact that some queries are informational. Medical queries are often of this type where the user really wants to read all of the top results, and they are almost all equally relevant. NDCG is optimizing on the assumption that there are right answers and the result positions are exponentially more valuable at the top. Using NDCG as a test for the relevance of the end to end search product, and for deciding which AI-based rankers to deploy into production means there is a bias against informational types of queries.

**Bias from Performance**

Another aspect of end-to-end relevance of the AI-based search engine is reliability. The

way search engines often work is a machine receives the user's query, and that machine talks to over a hundred other machines and asks them for the best results they have. These machines are given as little as 100ms to reply back to the querying server and return the merged results to the end user. If the network is slow, or a machine that happens to have the best URL has a functional bug and crashes, the search results might be missing the best URL, dramatically impacting NDCG. It is often important for large scale AI systems to be functionally reliable and performant, or the net relevance of the AI based system can be compromised.



# AI BIAS AND TESTING

By now it should be clear that testing AI systems is really the process of identifying, understanding and making a judgement on the bias of these systems. Training AI systems and testing those systems are really one and the same. They are both systematically trying to understand the bias inherent in the system. The testing of AI based systems is just glorified testing and quality work. At Google Search, the AI and machine learning engineers aren't called Software Engineers, or even AI Engineers, they are called Search Quality Engineers. These engineers are constantly running experiments and testing their hypotheses. Most of their time is spent testing and building testing infrastructure and analytic tools.

If you are a functional software tester today and are asked to test an AI system, your primary job will be to identify bias and judge whether it is good or bad. If you are a classic, functional software engineer transitioning to an AI role, much of your new work should be focused on quantifying and managing the bias in the system. Transitioning into an AI role means becoming more of a tester than a standard engineer. Testing in this new world of AI is looking for bias. Looking for bias in AI systems means testing.

While we primarily used a search engine as an example of an AI-powered app, the examples of bias above are generally applicable to most AI software projects. Search is a complex and large scale project and is vulnerable to most of the biases that smaller systems will also encounter. I ask that engineers working on AI-based projects to consider which of these biases and testing issues should be considered for their own projects.

test.ai

OUR MISSION

# AUTOMATE THE WORLD'S APPS

We improve the quality of software apps by making app development easier through the power of AI and automation.

For more information, visit https://test.ai