**Feedback Loops in Machine Learning Systems**

Designing feedback loops in machine learning system design



Photo by [Tine Ivanič](https://unsplash.com/@tine999?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

**Introduction**

In machine learning systems, we often receive some kind of feedback from the model’s environment, that is then fed back into the system. This can take many forms, such as using the model’s output to train newer versions of the model, or using user feedback on the model’s decisions to improve the model.

**While many feedback loops are useful and will improve the performance of your model over time, some feedback loops will actively degrade the performance of the machine learning system over time.** Designing helpful feedback loops is an important component of machine learning system design, and needs to be done thoughtfully in order to ensure that your system is sustainable.

In this post I will go over examples of both helpful and harmful feedback loops, and explain why each type has the result that it does.

**Helpful feedback loops**

A beneficial feedback loop typically involves **bringing in unbiased, external information** into your machine learning system. This often occurs in the form of obtaining labels through a source that is not strongly correlated to the outputs of the machine learning models.

These feedback mechanisms allow the system to get information that is not self-reinforcing with respect to the system’s current behavior. This ensures that, over time, the models do not degrade due to being retrained on biased data that does not reflect the true data distribution.

In many cases, this external data provides **feedback on when the model was right and when it was wrong**. This often occurs by implicitly or explicitly allowing the user to correct the mistakes of the system in a structured manner.

**User reports**

On platforms with user-generated content, there is often functionality for users to report content as being irrelevant content, spam, or offensive content. This allows end users to provide the machine learning ranking and**content moderation systems** with feedback on pieces of content that the algorithm sent them.

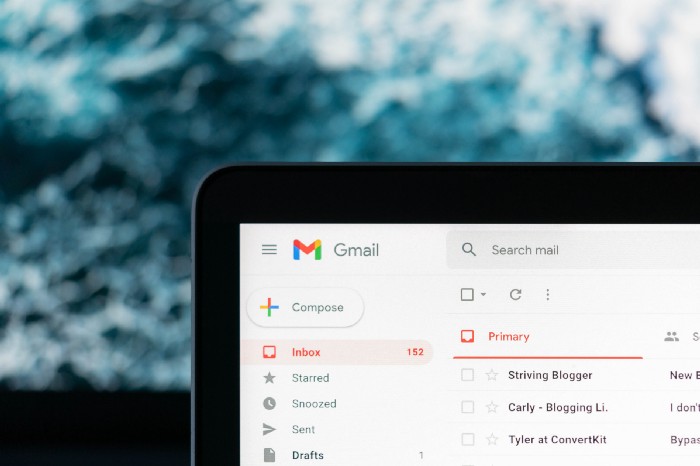


Photo by [Justin Morgan](https://unsplash.com/@justin_morgan?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

An example of this is how **Gmail** allows you to report emails in your inbox as being spam. If the email landed in your inbox and not your spam folder, then it implies that Google’s spam classification systems did not believe the email was spam. By reporting the email as spam yourself, you provide Google with valuable feedback that this email was a false negative. The next time their model is retrained, it can now take this feedback into account, and hopefully perform better on this type of spam in the future.

**User interaction logs**

On any platform involving **content ranking or shopping**, there is typically some form of content ranking system that decides which items best match the user’s search query or historical engagement. However, in many cases, the algorithm’s suggestions are not perfect, and users may choose to either engage with items that were not recommended, or engage with low ranked items over high ranked ones.

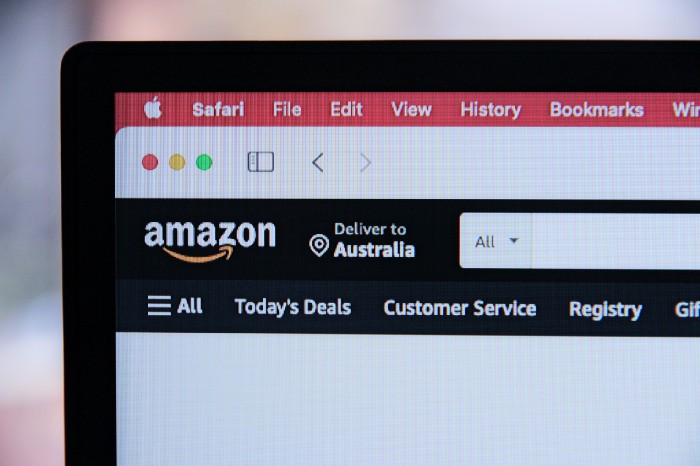


Photo by [Marques Thomas](https://unsplash.com/@querysprout?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

The logs from these user interactions provide the ranking system with ground truth for what the user was actually interested in. When a user interacts with something that was not ranked highly by the algorithm, it provides the system with valuable feedback for its next retraining. In addition, the more user interaction data that the platform has for a user, the better it is able to build profiles for their specific interests and target their specific needs.

**Harmful feedback loops**

A harmful feedback loop in a machine learning system typically **prevents the system from improving over time** by removing the presence of unbiased data.

When the data used for training the model is determined solely by the previous model’s outputs, the system has no way of improving over time. In fact, this scenario will most likely lead to the system degrading in performance over time, as it will continue to be trained on data that is more and more biased towards its own outputs.

In this situation, **the system is not able to correct its mistakes** at all. Over time, these errors accumulate and have a larger and larger impact on the system’s performance, gradually lowering its performance as the model is retrained.

**Interventions**

In many machine learning systems, the model triggers some type of intervention that alters the user experience. When the intervention is triggered, some users will naturally drop off if they are unable or unwilling to bypass the intervention. Over time, this population of users dropping off may significantly change the data distribution for labeled data.

For example, in a **fraud detection** system, users with a high risk score may be required to complete additional verification. If the verification is effective and leads to fraudulent users giving up without solving it, then those users will never get the opportunity to actually commit fraud and get labeled as such.

This video from [Stripe](https://stripe.com/) explains how they dealt with this type of feedback loop. They adopted the holdout group method and were able to unbias their machine learning training and evaluation data.

Without explicitly handling this, the model’s future training data will be biased, and will only include positively labeled data points on fraudulent users who either 1) solved the verification challenge, or 2) were below the model’s threshold and were never challenged. This gradually biases the model’s training data towards more “difficult” fraud over time, as the “easy” fraud gets stopped by the intervention. It also decreases the amount of (already scarce) positive examples in the training data, as the existing system is likely to prevent most fraud from occurring.

**Solution:** We can fix this by using a small (e.g. 1% of traffic) [holdout group](https://www.analytics-toolkit.com/glossary/holdout-group/) that is not subject to the intervention. In the holdout group, we do not intervene, and let all users go through without challenge, even if they are risky. We can then see what happens when we do not require any verification; some users will go on to actually commit fraud while others won’t. This small population provides us with unbiased data for training and measurement, without causing significant harm to the business due to its small size.

**Position bias**

**Search ranking** systems typically use user interaction logs to create labeled training data. Under a binary classification framing, if a user decided to interact with something, then the label is positive, and if not, the label is negative.

This labeling method leads to bias because the items that were ranked highly by the previous model are far more likely to be interacted with than other pieces of content, even if the highly ranked items were not actually relevant. Without adjusting for bias, this will lead the model to largely just memorize its past decision logic (and therefore its past mistakes), rather than learning the users’ true tastes.



Photo by [Christian Wiediger](https://unsplash.com/@christianw?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

**Solution:** This issue can be addressed by making changes to the model. There are several ways to model the effect of position bias, and then remove this effect at inference time. For example, the [Recommending what video to watch next: a multitask ranking system](https://dl.acm.org/doi/10.1145/3298689.3346997) paper explains how Google implemented this for YouTube. By learning to model the impact of position, these methods can pass in empty values or static values at inference time to signal to the model that all pieces of content have the same position factors, and should instead be ranked based on the content itself.

This video from the authors of the Google paper goes into more detail on the paper, which describes the overall content recommendation system for YouTube.

**Conclusion**

Feedback loops are critical elements of real-world machine learning systems. They enable the model to improve our time, and help collect feedback on where the model is making mistakes. In most cases, this improvement requires that your system’s feedback loop includes obtaining unbiased data that is not purely produced by the model itself.

Hopefully this article has given you a better idea of what kinds of feedback loops can help or hurt the performance of your system. **Designing feedback loops is an important component of machine learning system design in any industry**, so the learnings from this post should apply in a wide variety of contexts.

If you are interesting in learning more broadly about machine learning system design, [Stanford has a great course](https://stanford-cs329s.github.io/syllabus.html) that covers many aspects of building and deploying machine learning systems.