Ofcom
making communications work
for everyone

Winder.AI
ML | RL | MLOps - Consulting | Development

Automated Content Classification (ACC) Systems

January 2023

An investigation into how social media platforms use MLOps software, systems, and processes to manage user-generated content

By Dr. Phil Winder, Luke Marsden, and Enrico Rotundo —Winder.AI

Foreword

As the UK's converged communications regulator, Ofcom oversees broadband and mobile telecoms, TV, radio, video-on-demand services, video-sharing platforms, post and the airwaves used by wireless devices. We have a statutory duty to promote media literacy under section 11 of the Communications Act 2003. As part of fulfilling this duty, we are required to take steps to: (a) bring about a better public understanding of the processes by which material is selected for publication by means of the electronic media (including on online platforms); and (b) encourage the development and use of technologies and systems for controlling what material is received that are effective and easy to use.

In recent years, a wide-ranging global debate has been taking place about the risks faced by internet users and the steps that can be taken to help protect them from harm. A key element of this debate has centred on the role and capabilities of automated approaches (driven by Artificial Intelligence and Machine Learning techniques) to enhance the effectiveness of online content moderation and offer users greater protection from potentially illegal and harmful material.

Ofcom has commissioned Winder.Ai to produce this report to improve our understanding of the end-to-end processes that support the creation and deployment of automated content classifiers used in moderating online content. We are publishing this report with the overall goal of improving awareness of how these systems work. We wish to thank those that participated in the interviews and helped provide the content for this report.

1. Introduction

Ofcom has been named as the new online safety regulator in the United Kingdom (UK), in the Online Safety Bill (OSB).[1] The OSB will give online platforms (such as social media platforms and other services enabling interaction between online users, as well as search services) new duties which are aimed at improving online safety.

One of the ways that online platforms seek to keep their users safe is by moderating content, which is the process of reviewing user-generated content against a platform's community guidelines and taking action against it if it violates those guidelines. Content is often moderated by people, but manual moderation processes can struggle to cope when the amount of content increases.

Platforms have therefore sought to scale their moderation processes using technology. When data is at the heart of that technology then it is often said to be powered by artificial intelligence (AI) although the implementation might be closer to traditional software engineering. For example, after a piece of content has been reviewed by a moderator, platforms might use technology to find duplicate content and automatically administer the same review.

More advanced automation might make decisions based on the predictions from models trained upon data. Here, it is the decision (or series of decisions) to allow content to be published for public consumption that represents the goal. The information used to decide whether the content is in violation is often complex and relies on the context, geography, demographic information, as well as content itself. The field of AI has many subdisciplines, but the task of training complex models to make decisions based on data is known as machine learning (ML).

1.1. Purpose of this Report

Ofcom commissioned a report from Winder.AI[2], industry recognized experts in AI consulting, to conduct interviews with a variety of platforms that engage in content moderation. This includes large, medium, and small-sized social media platforms, third-party moderation software vendors, and industry experts. The interviews were performed under the direction of Winder.AI and this report reflects those experiences. The views presented in this document are the views of the authors and not the view of Ofcom.

We investigated the processes, systems, and tools used to build and operate artificial intelligence-powered automating technology. We focused our interviews by conducting a semi-structured interview that asked hundreds of questions in the following broad areas:

[1] https://www.gov.uk/government/publications/draft-online-safety-bill

[2] https://Winder.AI

1. How do platforms build their content classifiers?
2. What techniques, tools, technologies, and hardware are platforms used to implement content classifiers?
3. What processes do companies use to manage the development and operation of classifiers?

The moniker machine learning operations (MLOps) is often used to describe the use of these AI-focused processes and systems. MLOps is an industry-standard term used to describe both the process and the tools used when creating and operating an algorithm or model — in this case, decisions related to preventing harmful content from appearing on a platform. MLOps also emphasizes empowering engineers to own and take responsibility for their creations by supporting them with productivity tools.

This report aims to present the development and operation of AI technologies inside various platforms and third-party vendors from a technical and cost perspective, whilst focusing on online safety. However, the interviews contained commercially sensitive information, so we removed confidential information, aggregated the remaining content, and anonymized all uniquely identifying information. Some of the content in this report is from public sources, for which we have provided references. More detail about how we did this can be found in section 1.3.

1.2. Document Organization and Executive Summary

To help constrain the size of this report, we have split it into groups of sections that roughly align with the industrially accepted version of the ML project lifecycle - note that the ML lifecycle is often highly iterative and non-linear, however. We asked questions surrounding these topics to all interviewees and recorded their responses. Although we strived to obtain good coverage across the ML lifecycle, we were time constrained and therefore some sections may have gaps.

Each sub-section of the report defines the elements involved, explains why it is important and discusses our findings in terms of similarities and diversities in implementation.

Below is a list of the main sections of this document and an executive summary of the content.

1.2.1. Working With Data

In section 2, Working with Data, the report investigates how platforms manage their data to ensure quality and fairness and how they implement their data engineering practices.

Because platforms have such vast amounts of data, their data architecture forms an important foundation for the rest of their technology stack. Platforms and vendors often create a bespoke system that makes it more efficient to work with their data. They then build bespoke database interfaces and pipelining technologies on top of their data architecture, many of which end up as publicized open-source projects.

Fairness and privacy are increasingly important topics in AI and as such platforms and vendors are investing in improvements. At the data level, these initiatives manifest as implementing testing and monitoring solutions to improve data quality and observability.

Extra tools to help people collect data and label content are built on top of this data stack. Some platforms and vendors also talked about how they work with their data via feature stores and version control systems.

1.2.2. Machine Learning Development

In section 3, Machine Learning Development, the report explains how platforms and vendors develop, train and evaluate the machine learning (ML) models used to make moderation decisions.

We found that perhaps unsurprisingly, many platforms and vendors are using highly sophisticated ML techniques to implement their models. We also found that they had well-established procedures to convert their requirements into a working model. Training and evaluation tended to be dependent on their respective technology stack. Much of the time data scientists reuse well-understood models and implement or improve solutions by altering how the input data is used. Evaluation is performed both during the experimentation phase on captured datasets and after deployment.

1.2.3. Operational Deployment

Section 4, Operational Deployment, discusses the procedures around deploying, logging and monitoring models as well as hardware requirements.

Platforms and vendors use well-known technologies in the deployment of their models. Platforms are often running at a large scale and therefore use advanced deployment techniques. Logging and monitoring are used throughout to trace execution and to help maintain reliability. Platforms have invested heavily in hardware and data centres to run their AI workloads.

1.2.4. Machine Learning in the Organization

Section 5, Machine Learning in the Organization, looks at how platforms and vendors structure their AI teams across the business, how they govern the use of AI, and how they leverage third-party tools.

An organization's size was the key factor. Larger platforms have a more defined structure with distinct responsibilities. Smaller platforms and third-party vendors tend to be more product-focused, rather than role-focused.

Most platforms leveraged third-party tools at some level, and we also found that third-party vendors also consumed other third-party tools. The report details different ways in which companies interact with vendors and some challenges that they encounter.

1.2.5. Costs

Section 6, Costs, briefly presents costing information gathered throughout the project. This includes information about how third-party vendors structure their prices and the costs involved with developing solutions in-house.

1.3. About the Information in this Report

This report presents aggregated, anonymized information provided by platforms, third-party vendors, and industry experts. We have also attempted to simplify the technical language to make the report more understandable to a general audience. However, the content of this report is still highly technical and assumes the reader has at least some experience in software engineering. The effect of this is that the topics in the report are summaries with little detail. Furthermore:

This report does not contain any confidential information.

All references to specific organizations or people have been removed, except for where information is publicly available. Public information is referenced.

The purpose of this report is to inform about current AI practices, not to specify or adjudicate. The remarks made in this report are not meant to be prescriptive and we do not recommend any actions.

Details about tooling and hardware were removed due to concerns about cyber security and identification.

The costs associated with third-party software, other software, hardware, and associated practices have been aggregated and anonymized.

The report does not discuss what or how models are used for specific use cases.

This report does not consider other areas where the public interacts with AI, for example, via search or recommendations, it solely focuses on the automation of the moderation of user-generated content using ML.

2. Working with Data

You can think of *data* in any modern organization as a very large pile of books. The books themselves are not particularly useful, other than for making impromptu guinea pig cages or tables. It is the information contained within that provides value.

For millennia, people have been using this information to build solutions to life's problems. Traditionally, engineers and scientists would use a mix of information sources to develop interpretations to solve a problem. Yes, the value creation process started with information, but it required people's interpretation to bear fruit.

More recently, people can develop solutions using the information directly. Insurance products, for example, use raw claims data to improve policy definitions and calculate premiums. More generally, technologies under the banner of artificial intelligence (AI) can automate decisions with machine learning (ML) and strategies with reinforcement learning (RL), using data.[1]

This section describes how organizations are managing and using their data to provide them with information for online safety.

2.1. Data Management

Data management considers the usability of the data that a platform or vendor owns. To successfully extract information and leverage data in downstream applications, data must be organized, understandable, and useful.[2]

There is a range of personas that interact with the data. Analysts talk of "exploring" the data to improve applications or establish whether solutions are viable. Data scientists extract information by training models on data – see section 3, Machine Learning Development for more information. Data engineers distribute, clean, and derive new data. These interactions, and many more, are maintained and controlled via data management processes.

Annotating the data with labels is an important phase because it provides the ground truth for downstream models. In other words, annotations inform applications about what is happening, from which they can learn. The decisions of annotators are often used as labels for machine learning, where labels are used to drive the training process via supervised learning.

Annotations are also important for measuring the performance of live applications. The results generated by models are frequently compared to human classifications to ensure models are performing adequately.

2.1.1. What Did We Learn?

Platforms and third-party vendors are investing in building data management tooling to support their data and their processes. Whilst many platforms leverage open-source building blocks, their systems are often bespoke.

Common Themes

Platforms and vendors have sophisticated internal tools for creating and curating datasets at a massive scale. These are often built on top of open-source storage or database technologies.

The sheer volume of data means that large platforms and vendors don't leverage training datasets in the traditional, static, sense. Training data often comes from a filtered subset of live production data. Engineers and scientists leverage unified tooling to access and manage data. Downstream applications access data via standardized interfaces. We found that in some cases engineers lacked some flexibility to experiment with new techniques.

Platforms discussed the difficulty in creating policies (for example, Community Guidelines) that precisely define specific harm, which means that organizing the data can be challenging. Most took a metadata-driven approach. They push the content through a dizzying array of systems and tag the data according to what was found. Business logic then decides whether the content should be sent for review, or automatically blocked.

Large platforms have policy teams who are tasked with writing Community Guidelines and working with product and operations teams to apply moderation at scale. Platforms and third-party moderation tool providers use dedicated internal teams of trained annotators and domain experts like psychologists to help improve the quality of the data and the downstream models. Moderators and external domain experts annotate samples of data from production environments that help to monitor and improve algorithms that are in use. Platforms also annotate data that algorithms are unsure about to help improve the quality of the model.

Differences

Some smaller platforms and third parties tend to have more ad-hoc processes that are suited to their niche. For example, third parties working with child sexual abuse material (CSAM) often need to work with data "blind." To avoid storing or observing illegal material, they submit requests for training jobs to an authorized third party, like local law enforcement. Others have bespoke tools to manage their data in unique ways, like tools to help them curate ontologies of harm.

Smaller platforms are also less likely to be creating models and therefore are less likely to curate data. They often use third parties and pass the burden of data management to them.

Third-party vendors tend to specialize in specific data domains, like video, or text. Therefore, the systems and processes used to manage data are quite different. For example, vendors working with video were more likely to have on-premise storage solutions, to reduce data transfer and storage-related costs. Whereas those working with structured data or text were able to leverage managed solutions.

The annotation process is also different. Those working with videos annotate data frame-by-frame, leading to many labels per individual media asset. Those working with text annotate using ontologies to build sophisticated structures of meaning. Annotations can be fuzzy, in the sense that sometimes annotations have a spectrum of valid values. Platforms and vendors commonly have billions of annotations. In some instances, platforms also derive annotations from metadata about content, like the number of likes or comments.

All platforms have the same goal, to prevent harmful content from appearing on their platform, but they achieve this in different ways. Some attempt to curate golden datasets that pertain to a particular policy, whereas others consider violation as a moving goalpost.

2.1.2. Why Is It Important?

Modern organizations take advantage of their data by first establishing a need. The need, at a high level, expresses the pain that an organization or its customers experience. This is then passed to research-oriented data teams to establish whether they can access data that pertains to this need. The data teams explore the wealth of internal and external data to establish whether: a) they can observe this need in the data, and b) they can use this information to help provide a solution.

2.2. Data Quality, Privacy, and Fairness

Data *quality* defines how good the data is for a particular task. Data scientists often require *clean* data, meaning that the data is free of errors and inconsistencies, to produce reliable results.

Data *privacy* is the collective term used to define whether users' information is safe and used appropriately. Many platforms and vendors have a privacy policy that defines what information is collected, how it is used, and how it is protected. Derivative works like trained models may also fall under the same protection.

Data also fundamentally defines the *fairness* of a model. Recently, there have been examples where organizations were found to have, unknowingly, discriminated against gender or ethnicity. The problem sometimes originates from the way the data was selected. For example, if a category is under-represented in a dataset, the model will likely make erroneous decisions. In other words, situations like this are a result of an inadequate, biased selection of the training dataset. It is worth stressing that bias can live unnoticed and be involuntary. Thus, extreme care must be taken at that stage to make sure a training dataset is constructed fairly.[3]

However, the goal of a classification model is to discriminate. That is, to make decisions that are based on the data, like whether to approve a loan or not. The issue isn't discrimination per se, but rather how certain protected groups like age, sex, race, etc., are treated differently.

2.2.1. What Did We Learn?

Data quality and bias are universally considered to be hard problems to solve. Platforms and vendors invest large amounts of time in improving their data.

Common Themes

Platforms use sophisticated techniques to sample data. They have procedures to check for common sources of bias. Some use sophisticated data-gathering techniques like active learning to help improve under-represented parts of the domain. They also perform bias sampling in production to monitor for degradation of the model.

Platforms and vendors have access control mechanisms to restrict data to those that need it. Reducing the number of people who have access to it helps reduce the risk of potentially sensitive data in these training sets leaking outside of the organization. They also use retention policies to discard data that is no longer needed or before within a certain time frame.

For illegal content, like CSAM, platforms forward raw data to third parties who are authorized to hold such data and immediately delete it.

Platforms and vendors often emphasized that the social context is an important factor in the moderation process. What may be appropriate within one group may not be appropriate in another. Even certain words have wildly different meanings depending on how and when they are used. Large platforms state that human moderation is a necessary and valuable step that helps decipher intent.

Differences

One interesting challenge, when it comes to training AI models, is that retention policies may make it difficult to make training reproducible. For example, imagine that there have been complaints about a production model. An investigation might find that the data is to blame, by tracking the lineage of that model back to the data. But if the data has been routinely deleted as part of another process, then it may not be possible to reach this conclusion. Reproducibility relies on reliably stored artefacts.

2.2.2. Why Is It Important?

Data is the raw material used to build AI solutions. The quality of the data directly affects the result. To help ensure a good outcome, organizations use pipelines to process and clean the data of inconsistencies, missing values, and other issues. The goal is to achieve a certain degree of quality, which is essential for training reliable downstream ML models. The ML industry often refers to the aphorism that an "ML model is only as good as the data it is fed."

2.3. Data Processing

Developing sound data processing code, to extract, transform, test, and store data is only part of the whole picture. Organizations also need the ability to connect and automate steps to create workflows capable of fulfilling complex needs.

Data *pipelines* package and run tasks in a timely fashion to cope with the dynamics of real-world scenarios. Nearly every organization, regardless of its size, relies upon pipelines to manage the way data and models are processed.

Pipelines typically consist of a DAG (directed acyclic graph) of tasks which process data or train a model. The jobs are connected in such a way that the output from one is used as the input to the next and fan-out or fan-in as needed.

2.3.1. What Did We Learn?

All platforms and vendors use pipelines to process their data. Differences arise when considering implementation; they use a variety of different tools.

Common Themes

Platforms mostly work with tabular data and store their data in large-scale databases. They also use technologies that sit on top of databases to orchestrate and enhance capabilities. Pipelines are often encoded within the database technology itself, but some also use dedicated pipeline orchestration solutions. Unification around pipelining technology across the organization is common.

Snapshots and ad-hoc datasets are commonly derived from production data and stored in binary large object (BLOB) storage for posterity or future use. These are often used to train models. Some smaller platforms and vendors work with data on an ad-hoc basis by creating one-off snapshots.

A significant amount of time is spent working within data engineering pipelines to improve performance – also see section 2.4., Feature Engineering.

Differences

There is a wide range of data processing technologies available. Each large platform has tended to develop bespoke solutions that suit its data processing needs; there was little overlap. Many of these have been open-sourced.

2.3.2. Why Is It Important?

All data-driven organizations rely on being able to process data in a scalable way. Automated pipelines that are version controlled and monitored enhance the robustness of the system. Some pipelines can proactively prevent data loss and ensure that reliable data is always available.

Using version control with pipelines allows updates to be made to the pipeline code in a controlled manner. Pipelines can be developed and released according to software engineering best practices, like GitOps. This helps to minimize the risk of mistakes and presents a useful location to test the pipeline. Tests might include end-to-end tests, unit tests, and integration tests. Some organizations may even add compliance tests to ensure that the pipeline conforms to their expected standards.[4]

Triggering pipelines automatically ensures that derivative data is up to date. Using stale data is one of the most common causes of model degradation over time.

Pipelines are also important when implementing data governance. They offer a reliable and repeatable mechanism to write metadata to a centralized system for auditing and tracking.

2.4. Feature Engineering

Simpler, more informative data makes it easier to train models. For example, training a presence detection model on a passive infrared sensor (PIR) sensor is a lot easier than training on a video stream. This is because the PIR data is simple and informative; heat signatures are there or not there. Video data may be just as informative, but it is much more complex and therefore harder for the model to interpret.

Feature engineering is the art of transforming raw data into informative features. This often involves translating data into a sensible numerical representation expected by the model. Traditionally, features are devised by data scientists and analysts who apply their domain knowledge and heuristics, but more recent ML techniques can extract them automatically.

2.4.1. What Did We Learn?

Platforms spend significant amounts of engineering time improving how information is extracted from data. They use domain experts to derive foundational features that are used by application teams to solve problems. They also use *embeddings* extensively to build unsupervised

understandings of the data – embeddings are simplified representations of the data learnt via a training process.

Platforms tend to build bespoke implementations of a feature store that builds on top of their data stores.

Smaller platforms tend to perform feature engineering on a per-application basis. The feature engineering pipelines are often specific to the application and the data.

Specialist third-party vendors are also leveraging generative techniques to augment existing or synthesize new data to improve the performance of the models they train.

Every organization performs feature engineering, but how they do it varies and usually falls into one of three categories:

1. **Implicit**: Feature engineering occurs in a data pipeline on the way to training a model.
2. **Organizational**: Feature engineering is fundamental to the structure or value proposition of the organization and is exposed as a dedicated service.
3. **Feature store**: Feature engineering occurs within an explicit feature store in their architecture where multiple teams can reuse each other's features.

2.4.2. Why Is It Important?

Two key reasons for the importance of feature engineering are:

1. A shared understanding of the domain helps the organization be more efficient by avoiding duplication of work.
2. Reusing informative features encourages robustness and scalability because more people are consuming them.

The benefit of sharing features depends on whether many teams are working on the same problem. If so, a feature store will be valuable. Otherwise, feature engineering can be performed within existing pipelining tools.

2.5. Data Version Control

Many problems in AI derive from poor data. Data can suffer from a variety of quality issues, like missing, biased, and corrupted data. Data can be incorrectly labelled. Data can contain personally identifiable information (PII) and other correlating features that cause models to erroneously infer patterns caused by socioeconomic contexts. Data can *leak* into a model – the act of using data to train a model that isn't available in real life.

Irrespective of the issue, post-mortems need to be able to access the exact data that was used to train or generate downstream artefacts. How this replica is stored is the topic of this subsection.

The term *version control* is quite broad but means that it should be possible to "time-travel" through past views of the data. This could mean movement in a streaming sense, where events are replayed, but most traditional version control systems present static snapshots of data. Git is

the canonical representation of a version control system for code and small text-like datasets, but it isn't designed to work with large amounts of data.

2.5.1. What Did We Learn?

Most platforms and vendors version their data in some way. But some are limited by the lifetime of the data. Others rely on the data being snapshotted.

Many follow the practice of versioning data via metadata. They do this by storing information about the data in a separate storage location like a database. They can keep track of how data is used by adding new metadata. This is a common practice, but it doesn't prevent data loss and requires other systems to maintain the state of the metadata. It is easy to lose or delete the source data via external processes. It is also hard to ensure that metadata is reliable because people are responsible for ensuring it is up to date.

Organizations that rely heavily on databases to store their data create versions by ensuring queries have fixed date ranges. Then, in the future, they can replicate the state of the data by repeating the query. Again, data could be inadvertently deleted or corrupted in the interim.

Some platforms and vendors used snapshots of data to ensure that downstream processes are entirely reproducible. Whilst being a gold standard in terms of reproducibility, this is challenging when needing to delete data, to comply with legal requirements, for example. Therefore, this process tends to suit vendors that have datasets that rarely change, not large platforms.

Larger platforms built have metadata-based versioning into their bespoke data and pipeline solutions. Whereas smaller platforms and vendors tended to version data using ad-hoc processes.

2.5.2. Why Is It Important?

When engineers are notified of a problem affecting users, their first task is to find and isolate the cause of that problem. The exact steps required depend on the specific issue, but often data has a role to play. Engineers would first analyse the lineage of an artefact that had an issue and that would, ideally, link back to the exact set of origin data. They can then pass on details about the issue and links to that exact data for deeper analysis. An analyst or expert in the domain would then attempt to establish a causative link between the data and the problem. Once the link is established, they pass this information back to data scientists and engineers that attempt to fix the problem.

An analysis of the data is only possible if the data exists in the same state as it was when first used. Furthermore, even if the data is not the problem, engineers still need it to replicate subsequent processes like training models or processing data.

3. Machine Learning Development

Data scientists develop *models* of the world to help make sense of it. They use models to automate decisions and investigate what might happen in the future. The word model is used throughout science and technology to describe an approximation of a system. Models are rarely 100% accurate because most systems are chaotic.

But the scientific and AI communities develop models in different ways. Scientists use the scientific method to help prevent fitting arbitrary models to random observations. They define a hypothesis, based on prior knowledge, and test this by observing the real world. In AI, however, data scientists start with the observations and set out to find a model that fits their data, the exact opposite. This can lead to overly optimistic models. But often they are useful because they approximate expert systems, which are systems that are good at doing one thing only, like playing chess. AI models are surprisingly capable of modelling a tightly defined problem if you have enough representative data to train them on.

In the context of online safety, models are used to predict various aspects of the content and its context. The end goal is that platforms or vendors classify content according to the ontology they have developed to describe their policies. For example, it could be offensive or not offensive. Or it could have a more fine-grained numeric representation describing how it was offensive or whom it would offend. In general, more complex domains with more complex data types require more detailed ontologies.

This section investigates how platforms and vendors develop, train, and evaluate their models.

3.1. Model Development

All AI projects have a reasonable chance of failure. The most likely cause of failure is not having enough or the right data. But quite often projects fail because data scientists can't produce a model that solves the problem well enough to be useful. For example, data scientists could create models to predict health complaints, but they may not be accurate enough to be deemed safe for use. Another common reason for failure is that the data scientist has solved the wrong problem or a problem that isn't useful or timely. To mitigate against wasting money due to failure, AI projects typically follow an iterative development procedure depicted in Figure 1.

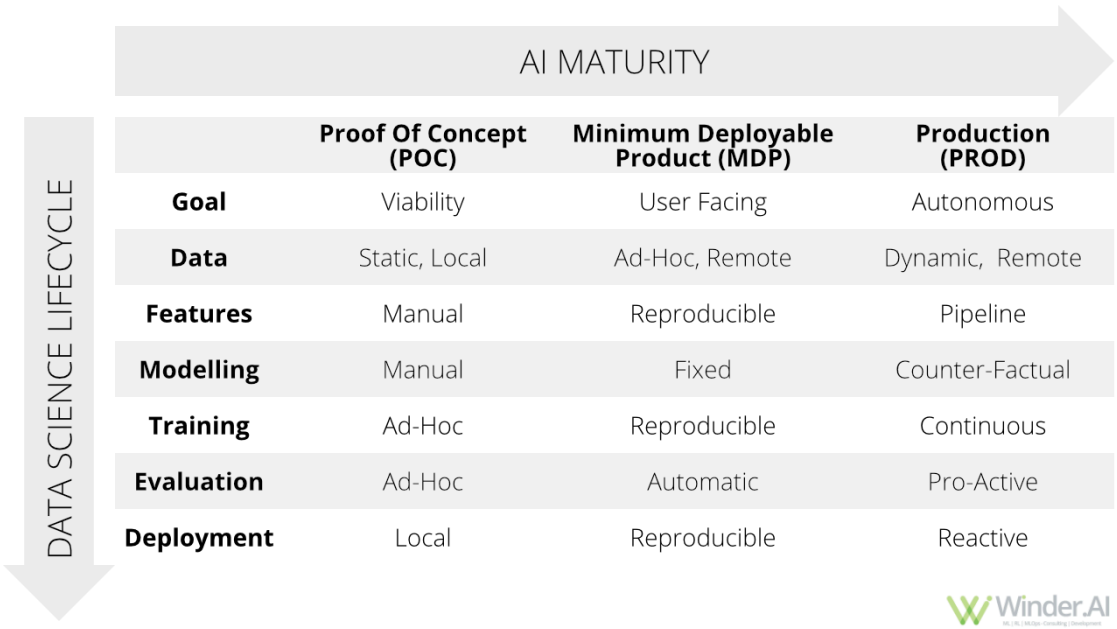

Figure 1: A representation of the AI project lifecycle mapped onto the maturity of the project and the data science lifecycle. Courtesy of the Winder.AI blog, with permission.

People often run a proof-of-concept (POC) phase to prove viability. After that, they move into subsequent phases that aim to deploy a user-facing model. In the final phase, the application is hardened and automated. Moving across the image from left to right represents how mature the project is.

Moving down, from top to bottom, represents the traditional ML lifecycle. But it's unlikely that the data scientist will design the right model, or the right features, the first time. It's equally unlikely that they have all the data they need too. So, the data scientist will work in a loop, improving the performance and robustness of the model over time. How they do this, however, depends on the phase of the project.

Both dimensions help to quantify the maturity of the project. Throughout this process, a range of personas is involved, like data scientists, data engineers, ML engineers, and MLOps engineers. For more information about the ideas contained within each cell please refer to the books in the bibliography.[5][6]

3.1.1. What Did We Learn?

Many of the platforms and vendors we spoke to are leaders in the AI industry. So, it was unsurprising to find that they use sophisticated processes and modelling techniques.

Common Themes

Data science teams usually begin a safety project when policy teams add or update a new policy. They then tend to gather labelled examples of content that fits within the policy definition. Throughout the lifecycle of the project, policy teams are often working together with engineering teams to help drive the model to be as close as possible to the policy definition.

Social media platforms generally invest heavily in developing efficient data capture and experimentation platforms. They can rapidly collect a sub-sample of production data and ask the experimentation platform to train several well-known models to quickly evaluate performance on offline data. They then deploy this model into production and submit requests for manual labelling and evaluation. This takes a matter of days to produce an actionable ML model. Platforms often have too much data and they need to build simple models that help to find and reduce the amount of data.

Larger organizations reported that they were using embeddings heavily, which are usually fed into task-specific models that are tuned towards policies. They use different embeddings to represent different types of content. Embeddings reduce the complexity of data by "embedding" that data into an abstract space. Embeddings are trained independently of models making decisions, which makes it attractive as an alternative to traditional feature engineering.

Another trend was the use of sophisticated neural network-based architectures within models. In terms of model implementation, all the popular neural network frameworks were cited. Participants described architectures that ranged from "vanilla" multi-layer perceptrons to transformers – architectural patterns that have been proven to work well at specific ML tasks.

Platforms and vendors also used simpler models like gradient boosting methods, random forests and regression, especially when the data was tabular. Typically, the complexity of the modelling technique is scaled with the complexity of the data. However, organizations tended to unify upon certain types of models that they had institutional experience with.

Platforms and vendors also discussed how other phases of the AI lifecycle can impact modelling decisions. For example, one suggested that if the application required fast decisions from a model, then that might rule out some of the more complex modelling approaches.

Differences

Smaller platforms tend to operate in a more "traditional" manner, collecting datasets and developing code offline.

Platforms repeatedly stated that most problems are unique and require representative training data. They suggest that off-the-shelf models are a useful baseline but require custom training and development.

Each platform and vendor have subtly different approaches to breaking down modelling problems. Some build a highly curated ontology to define what it is they are trying to achieve. This helps to shrink the complexity of the problem into bite-sized chunks. Others combine models using ensembles to produce higher-level capabilities. And some rely on huge amounts of data and complex models. Each approach has its strengths and weaknesses and is usually dictated by external constraints, like access to supercomputers or the business model.

Some platforms and vendors create models from content that they are not allowed to store or view, so they must submit model training and labelling requests to an external organization. The external party then performs the training and feeds back anonymized results. These runs are typically encoded pipelines that the external party can run on their hardware.

Large platforms tend to have a lot of models, which means that it is difficult for individual engineers to discuss the nuances of all the different models that are being used.

3.1.2. Why Is It Important?

Presenting the state of the art of machine learning is outside of the scope of this report. However, we would like to talk briefly about some ideas that emerged from our interviews.

Many organizations are using very complex neural networks in their solutions. This is required because the data and the domain are complex. Content moderation requires a model to achieve a human-level understanding of a piece of content and its context. However, this increase in complexity has several knock-on effects.

1. Complex models take more training time and resources. This makes them expensive to develop and run.

2. Complex models require more data. Large models have a higher dimensionality that leads to more "gaps." The model will make mistakes when there is no data to fill these gaps.

3. Complex models deliver complicated answers. It can be hard to explain why complex models made the decisions they did, although not impossible. It is hard to ensure fairness because complex biases can exist across complex combinations of dimensions in the model space.

However, it appears that simpler, community-derived metadata is still useful. Some platforms talked about using content metadata as the primary moderation mechanism. For example, if someone has been moderated in the past it doesn't take a complicated model to predict that they're likely to produce content that needs moderating in the future.

3.2. Model Training

Models consist of two types of artefacts. The first is the static definition of the model, which may include extra processing functionality, architectural details, and tests. This is typically represented by code leveraging machine learning frameworks. The second is the dynamic parameters of a model. These are a set of weights or parameters that effectively define how to combine features to produce a result. Sometimes the weights are stored inside a deployable asset – see section 4.1 – but more often they are stored independently in a model registry or as a file.

To *train* the model weights, engineers leverage the machine learning framework's training algorithms. This typically involves using the latest version of the code and parameters to make a prediction and then comparing those results to the labels. The framework then measures the amount of error in the prediction and nudges the model parameters to perform better on the next iteration. This repeats until no improvements can be made.

Developers can train locally on their laptops, but local conditions make it difficult to repeat elsewhere. For example, there may be different versions of libraries being used or out-of-date data. Instead, many platforms and vendors rely on repeatable managed pipelines to perform the training consistently. That way, if anybody else needs to retrain a model, like when the labels have been updated, they can do so with a click of a button.

3.2.1. What Did We Learn?

The use of pipelines was almost universal throughout our discussions, but we found a high level of diversity.

Common Themes

Most platforms and vendors said that they are using some form of pipelining technology to orchestrate the retraining of models. They also reported that malicious actors are quick to circumvent automated moderation systems and therefore it is necessary to retrain frequently.

Platforms and vendors reported a wide range of periods between retraining, anywhere between hourly and yearly.

For the models that were retrained regularly, the most common explanation was that the underlying nature of the data is highly dynamic, like in spam prevention applications, for example. One counter-explanation for the longer durations between training was that some models are well-constrained, and the nature of the data doesn't change much over time. Some smaller platforms and vendors said that they would like to retrain more easily.

Some use sophisticated techniques like continuous and active learning. Continuous learning is a technique that allows for automated retraining based on what is observed in production. Active learning is a way of improving models by allowing the model to choose which observations would be best to add to the model to improve performance.

Some integrate pipelines into their metadata management systems to improve the governance of the models. This helps to map what models are used in production and find what data they depend on.

Again, platforms and vendors often use open-source solutions to provide the backbone of their pipelines, often with custom overlays to help optimize workflows for their use cases and data. There was a surprising level of diversity with many technologies being cited.

Differences

One common comment was that some pipelining solutions tended to be hard to use. This was because the solutions still required a lot of "plumbing," and some had locked themselves into a particular technology framework which prevented them from experimenting with other solutions easily.

Some smaller platforms reported that they were using remote SSH connections to manually start retraining. They used scripts to encode the training process.

3.2.2. Why Is It Important?

If platforms and vendors wish to use machine learning, then they must train a model. But how the model training is performed is also important. If model training is automated and a mature process, then that makes it easier for engineers to keep models up to date. It also encourages experimentation which might lead to better models.

3.3. Model Evaluation

Model *evaluation* is the phase of the machine learning lifecycle where you quantify how well your model does the job it is supposed to do. Model *metrics* are used to describe performance.

Although evaluation tells you how well a model performs, it doesn't explain why it made the decisions it did. Model *explainability* is the practice of articulating the reasoning behind a decision. It often takes the form of predicting which parts of the data contributed to the decision. But because of the breadth of possible machine learning approaches, different algorithms require different techniques.

3.3.1. What Did We Learn?

Recall that models are simplified approximations of real life. They make broad assumptions to help automate some of the decisions that are made in the real world. But there are always edge cases, like those that lie on the boundary or situations that occur with odds of "one in a million." Therefore, it is important to stress that no model is perfect; you should always expect errors.

Common Themes

The type of metric used to describe model performance often falls into one of two categories:

- Technical metrics evaluate how well the model performs given the technical task at hand. Data scientists use technical metrics to compare models and monitor training.
- Business metrics measure how well the model achieves its goals, like the number of allowed posts that were eventually reported as harmful. This will be discussed in more detail in Operational Deployment.

Data scientists often use technical metrics as a proxy for business metrics because often business metrics are not known at training time. But business metrics are sometimes a trigger for retraining.

Some platforms made a distinction between metrics for training and governance. Technical evaluation metrics might describe how well a model performs, but governance metrics track risk. Governance metrics are often evaluated by trust and safety teams to help them decide whether a model is "good enough" to be deployed. Key performance indicators (KPIs) are often used to demonstrate that models are achieving their long-term goals.

Metrics are often generated automatically and built into the pipelining solution which helps to unify the view of performance across the organization.

It is hard to suggest figures that represent how well models perform in general because each use case is different. But we found that platforms and vendors tended to be aiming for technical metrics that approach 95% for a particular use case. At this level, models were generally performing well enough to be considered "very successful." However, given the difficulty of

some of the use cases involved in content moderation, scores were not always that high. Some reported that models with scores in the 70s were sometimes used.[3]

Also, note that it is easy to misrepresent the performance of a model by systematically altering the definition of success. For example, most platforms and vendors have a class in the middle of both positive and negative to represent "unsure." In use, content that falls into this category is sent for manual review. But during training, you could arbitrarily increase the width of this band and claim a "100% moderation rate." In real life, there aren't enough moderators to tackle this many unsure cases, so there are financial incentives that prevent this.

All platforms and vendors stressed that improving the performance of a model is an ongoing process and that only long-term initiatives obtained the best performance.

Some vendors emphasized the explainability of a decision. Using smaller, more focused models and symbolic approaches that encode business logic can help in this regard. But newer, powerful explainability tools are becoming available that work with even the most sophisticated models.

3.3.2. Why Is It Important?

Releasing a new version of any software carries risks and platforms don't want users to have a bad experience. In traditional software engineering, platforms and vendors enact tests with pass/fail thresholds to ensure the software works as expected. In machine learning, because the data is never 100% certain, there will be some failures. The difficulty involves deciding what type and how many failures are acceptable.

Model evaluation quantifies these failures based on predefined and randomized test datasets. These metrics estimate how many errors you would expect to see in real life. But they are only predictions and so model monitoring after deployment is just as important.

Data scientists will often improve their models to produce better metrics. They will alter algorithm types, model architectures, and thresholds to improve performance. They will then decide whether the performance of a model is acceptable.

Organizations must weigh up the risks between either improving solutions with new releases or sticking with the known, reliable status quo. Each organization will have a different risk profile. However, some organizations expose risk information via governance processes to standardize how, when, and why decisions are made.

Finally, explaining the decisions of models helps in two ways:

1. It helps debug models when they go wrong. If data scientists can see why a model made the decision it did, it is much easier for them to work on a solution.

---

[3] Note that these numbers are meant to imply the quality of the model when using balanced technical metrics like the F1-score or Matthew's correlation coefficient.

2. It helps the user. Explainability may form part of a product feature to aid user understanding or it may be used as part of a process, like a user complaint.

4. Operational Deployment

Models are only useful once they can be consumed by users, where a user could be another system or a member of the public. In large organizations, models likely form part of a larger process or are hidden behind an interface, so users don't interact with them directly.

In the case of moderating content, platforms and vendors talk about content being "tagged" and "flagged." This is when a model decides that a piece of content includes something the model has been trained to detect. There may be many models detecting different things, so content can be thought of as flowing through a pipeline with tags being attached at various points.

4.1. Deployment

Deployment refers to the process of taking a software or model artefact and publishing it to a location where it can be consumed. There is a range of common patterns that organizations can leverage to reduce the risk of failure.[7] These best practices emerge from cloud-native engineering, but data-driven models add complexity.

4.1.1. What Did We Learn?

Larger platforms have developed bespoke deployment solutions, which often leverage an open-source base. Indeed, large platforms may be the main contributor to open-source software.

Common Themes

The deployed artefact can take many forms and depends on the technology stack. Some platforms and vendors choose to work directly with model weights stored in file storage. Others compile weights into packages like containers or binaries.

At the largest scale, platforms act upon billions of events per day. To deal with this load they often attempt to pre-filter content so that it doesn't need to pass through all models. For example, content sent to small private groups has a different priority compared to something published globally. Even then, they need complex queueing structures to buffer messages if a failure occurs. Platforms and vendors also optimize the computational performance of their models by batching data together. The largest platforms deploy in the order of a thousand new models every day.

Platforms and vendors leverage cloud-native deployment strategies to reduce the risk of introducing bad models into production. They often run models in shadow deployments, which means that decisions made by the model are not acted upon, to ensure models operate as expected, both functionally and statistically. They also use canary deployments that are exposed to small proportions of live traffic to ensure that downstream systems are not affected. And if any issues are observed, models are rolled-back to previous versions. A/B deployments pit an incumbent model against a contender to ascertain which has the best performance in real-life. Some organizations perform these steps automatically.

Many platforms and vendors mentioned the use of a popular open-source orchestration platform to host their deployments and applications.

Public information shows us that Meta has built a variety of ML services covering the major use cases that internal teams use. The scale of the workloads is so vast that they designed their dedicated hardware and data centres.[4] Some parts of the organization use their Looper platform.[5] "Looper is currently used by 90+ product teams at Meta that deploy 690 models that make 4 million predictions per second."

Twitter has publicly reported the development of an internal ML platform named DeepBird. It allows their teams to experiment and productionize ML models built in a variety of frameworks.[6] Triggered workloads are orchestrated by Apache Airflow.[7] Twitter is built upon an event-driven architecture and consists of approximately 20,000 services. It is unclear how these services are deployed or where they reside, but Twitter has been a long-term user of Apache Mesos, an end-of-life orchestration system that predates Kubernetes.[8]

Public blog posts from Reddit show that they leverage a Kubernetes cluster dedicated to ML that hosts "Gazette," a model inference service with a single prediction endpoint that developers can send requests to. The architecture separates the request processing, run on Kubernetes, with the ML inference workload which is executed by an independent model pool. Although they support running TensorFlow models, they are moving towards supporting additional frameworks.[9]

---

[4] https://simplecore.intel.com/nervana/wp-content/uploads/sites/53/2018/05/IntelAIDC18_Kim-Hazelwood_Mainstage_5_23_Final.pdf

[5] https://ai.facebook.com/blog/looper-meta-ai-optimization-platform-for-engineers/

[6] https://blog.twitter.com/engineering/en_us/topics/insights/2018/twittertensorflow

[7] https://blog.twitter.com/engineering/en_us/topics/insights/2018/ml-workflows

[8] https://en.wikipedia.org/wiki/Apache_Mesos

[9] https://www.reddit.com/r/RedditEng/comments/q14tsw/evolving_reddits_ml_model_deployment_and_serving/

Differences

There is a strong distinction between those that store models inside packages, known as *baking*, and those that store model weights in file storage. Baking has the benefit of having a well-versioned artefact for lineage purposes but couples the machine learning process to a software development lifecycle. This has a knock-on effect where data scientists are forced to use software engineering tooling like CI/CD, but this is not necessarily a bad thing if done well. Storing model weights in file storage has the benefit of decoupling the data science asset from the MLOps/DevOps asset but can make provenance difficult if lineage metadata isn't guaranteed.

We also observed that there are still large barriers between data scientists and operations. We saw examples of data scientists "throwing models over the wall." See Machine Learning in the Organization for more discussion.

4.1.2. Why Is It Important?

Automating the process that moves an artefact into production reduces operational and temporal risk.

- Operational risks involve things like relying on specific people to fulfil processes when they might not be available or not having automated testing in place to ensure that deployments are successful. In these situations, it might not be possible to deploy an improved model, or it may not work as expected.
- Temporal risks revolve around updating deployments as quickly as possible to react to changes in the data. If processes are not automated, then it will take longer for improvements to be deployed.

4.2. Logging and Monitoring

Logging is the term used to describe the record of a series of events encountered whilst an application is running. *Monitoring* is used to measure the performance of an application. *Alerting* is used to inform engineers that the performance of an application is not acceptable.

Logging, monitoring, and alerting are discussed within the same breath because they all have a common goal: to keep the application running. But they are distinct concerns. Log data is usually in the form of a line of structured text. Monitoring data comes in the form of metrics like counters or timers. Alerting setup connects various log and monitoring sources to a mechanism to alert engineers.

4.2.1. What Did We Learn?

Monitoring tends to be the utmost concern for AI-driven use cases because models are likely to drift over time. This typically means that models have been trained on data in the past and the present data doesn't look anything like it did, so it's difficult for models to make accurate decisions.

The evaluation of production models is performed via feedback from moderators or community users. Reporting production performance over longer periods is increasing in popularity. Logging is less of a day-to-day concern as is used primarily for post-mortems.

Common Themes

As introduced in Model evaluation, platforms and vendors evaluate their production performance by obtaining a sample of predictions and sending them for human review. They then calculate the "true" technical and business metrics for their algorithms, typically using the same metrics used during model evaluation. Here we use quotation marks around "true," because human moderation decisions are still opinions. The opinions may vary to the extent that statistical testing techniques are necessary to determine the most likely correct "truth."

Evaluating models in production can be difficult because platforms and vendors need a constant source of quality labels to check that the model is producing the right answer. There are two hurdles to this practice.

1. The content must be available to perform an analysis. For example, this might not be possible with illegal content, data that contains sensitive information, or data that has been deleted.

2. Platforms and vendors need either a team of annotators or a way of empowering users to provide feedback, often both. Sometimes platforms design community feedback mechanisms into the product, like Reddit's upvote/downvote. But most platforms and vendors rely on in-house human moderation because they produce labels that more accurately align with their defined policies.

Some platforms and vendors are too small to spare capacity for continuous annotation and therefore only measure performance once every few weeks or months. Data scientists and other engineers would sometimes do this themselves.

Platforms and vendors also measure model and data metrics over time to ensure that the model and application are performing as expected. They suggest that changes in user behaviour are the most common cause of model failure, and that content is highly correlated with what is discussed in the news and online.

Platforms and vendors indicated that it is usual for models in certain use cases to drift quicker than others. Spam detection was one example provided where models can drift quicker than policy-breaking content. Many have technologies and systems to handle data drift automatically, but sometimes they do need to analyse the data and redevelop solutions when new threats emerge. Platforms reiterated the iterative nature of AI development and how they use monitoring to inform their development efforts.

Metrics are often viewed in dashboards for easy access and visualization. They help engineers watch for large shifts in distributions, trends, and outliers. Engineers often use measurements based on this data to identify problems and suggest improvements.

Thresholds are often used to define bounds around "normal" metrics, to determine when a model is misbehaving. When a metric crosses a threshold it sends an alert to engineers.

Logging typically includes information about the model, the content, the decision made by the model, and any related scores. Logging happens in several places. For example, some may store the raw data in one place, but the decisions in another.

Vendors often have contractual obligations to meet expected standards, like uptime or response times, but rarely offer guarantees about moderation performance.

4.2.2. Why Is It Important?

The goal of any application is to serve. If it cannot do that, for whatever reason, then it cannot fulfil its function. Logging, monitoring, and alerting are attempts to reduce the risk of failure.

Logging does this by providing the state of the application, which is used to explain why an application failed. Monitoring measures the performance of an application, to ensure it can continue to fulfil its use case. Alerting ensures that failures or potential failures are known, so engineers can fix them faster.

In the context of ML, monitoring is even more important than normal. Data tends to change over time, which results in degradation in model performance. Therefore, models are monitored for changes in the metrics about the data so that engineers can retrain the model on more recent data or make improvements to the design to mitigate it.

4.3. Hardware

Infrastructure is a term engineers use to describe the physical resources, or *hardware*, required to run applications. This might include network appliances like switches, routers or firewalls, storage arrays like blob-storage, physical disks, or databases, and computational elements like CPUs, GPUs, and RAM. Engineers can purchase this hardware and build a data centre, or they can rent these resources "in the cloud." Both options cost money, but the former is a capital expense whereas the latter is an operational expense. Purchasing hardware outright is cheaper in the long run compared to renting comparable equipment but burdens the organization and prevents it from quickly changing strategies. Usually, only larger platforms and vendors purchase hardware where resource requirements tend to be more static, but this isn't always the case.

Related to the hardware is the idea of having different environments for different purposes. For example, most platforms and vendors have an environment for production and an environment for testing. As the scale increases more environments will be dedicated to other purposes, like Meta has a cluster dedicated to testing mobile devices.[10]

[10] https://engineering.fb.com/2016/07/13/android/the-mobile-device-lab-at-the-prineville-data-center/

4.3.1. What Did We Learn?

Large platforms invest significant sums in building their own data centres to reduce the costs associated with long-term hardware requirements. Also, see Costs for a broader view of cost. See the bibliography for examples of leveraging cloud vendor services.[8][9]

Common Themes

Many platforms were unwilling to share specific details about production environments due to concerns that adversaries could exploit this information. However, they do offer a lot of public information about non-production clusters. For example, a recent blog post from Meta describes their on-premise AI research cluster consisting of 760 NVIDIA DGX A100 systems to present 6,080 GPUs.[11] They also say they are expanding this to eventually consist of 16,000 GPUs and handle 1 exabyte of local storage. Meta doesn't mention the cost of this, but each A100 retails for approximately 100-200 thousand USD, depending on scale.[12] If we assume the lower end of this range then the compute nodes alone currently cost 76 million USD. The final cost is likely to be of the order of approximately 200 million USD.

Note that this is just a research system. This does not include any of Meta's other development or production systems. Meta owns 18 other data centres, 14 of which are located in the US and 3 in Europe in Ireland, Denmark, and Sweden.[13] This has reportedly cost them 20 billion USD thus far, which demonstrates how small the research cluster is in comparison.[14] They don't specify how many nodes they own in total, but each data centre is likely to hold millions.[15] As a

---

[11] https://ai.facebook.com/blog/ai-rsc/

[12] https://www.engadget.com/nvidia-ampere-a100-gpu-specs-analysis-upscaled-130049114.html

[13] https://datacenters.fb.com/

[14] https://dgtlinfra.com/facebook-18-data-centers-20bn-investment/

[15] https://techcrunch.com/gallery/a-look-inside-facebooks-data-center/

comparison, as of April 2022, Google has 23 data centres to power itself and 88 for the Google Cloud Platform.[16]

There is a clear pattern of larger, more established platforms investing in their own data centres. This may have ancillary benefits like security and control, but the main reason they do this is that it is cheaper. Amazon reportedly has a gross margin of approximately 60% across all of Amazon Web Services (AWS) so platforms could save approximately half of their infrastructure costs by building their data centres.[17]

This manifests on a macro level too. Most platforms and vendors we interviewed had specialized on-premise hardware for development or training. They cite the availability and pricing of cloud GPUs as the core driver of this. AWS g5.48xlarge machines are roughly equivalent to the state-of-the-art NVIDIA DGX A100 machines - not quite, the DGX has 2TB of memory. You could recoup the costs of an on-premise machine after about 8 months of use. And given that AWS (and other cloud vendors) makes it difficult to apply to use these machines, because of availability issues, platforms and vendors see this investment as necessary. Those that can't afford the capital expenditure have developed associations with universities and other groups that are willing to share their equipment.

Despite the significant costs presented in this section, it is worth restating that it is hard to attribute online safety costs in isolation. These machines and clusters are often shared across the organization, which spreads the cost burden. Meta's new research cluster is a good example of this.

Most platforms and vendors reported that they use an environment hierarchy, like development, staging and production, which is a common pattern throughout the software engineering industry. All had production environments and larger platforms had multiple that are dedicated to specific use cases or teams. Nearly all had development environments to develop and test their applications.

Most, but not all had staging environments to perform final testing. This is complicated by the fact that some platforms have developed their platforms to a point where an application is just configuration, which they need to test on production data. In this case, there isn't a strict staging environment here because there isn't any new software for quality assurance. The result of the experiment is the test on a subset of production data, often in a shadow deployment.

---

[16] https://www.google.com/about/datacenters/locations/

[17] https://www.cnbc.com/2021/09/05/how-amazon-web-services-makes-money-estimated-margins-by-service.html

Differences

In general, the interviews presented a picture where smaller platforms and vendors tended to use managed cloud services and only those that are larger invested in dedicated data centres.

4.3.2. Why Is It Important?

Hardware use is important because it represents one of the main costs of running machine learning models, with the other being people costs. In general, people are more expensive than machines, so it makes economic sense to try and automate some of their work using machine learning. But designing, building, and operating machines still take considerable time, effort, and therefore money.

In machine learning, there is a particular emphasis on the amount and movement of data and the computational resources required by models. GPUs are by far the most expensive element to rent. The cheapest machine with a GPU on AWS costs approximately one dollar per hour of use. The most expensive is 32 USD / hour.[18] GCP and Azure have a similar range of prices. When it takes multiple days to perform a single training run, often described as "over the weekend" in our interviews, and engineers are performing many experiments at the same time, the total cost can be huge.

Moving data is also a challenge. Datasets can be incredibly large, and it would take an infeasible amount of time to copy them to another location. This limits where and how the data is used and has many knock-on effects like cloud vendor and geographical lock-in. Ingress and egress costs can also balloon out of control.

Within the machine learning lifecycle, there are differences in hardware requirements. Training models tend to be the most resource-intensive step in the process because not only does it take time to train complex models, but experiments are also often repeated to achieve the best performance, for example in hyperparameter tuning runs.

The hardware required for serving predictions to downstream users depends on the type of model and load. For some machine learning models, once trained on a GPU, they must be served on a GPU, although this is beginning to change with frameworks that optimize GPU-based models for use on a CPU. Without tools like this, the effort to convert GPU models to run on a CPU can be substantial. If there are many concurrent users, then engineers must "scale out" their model by replicating the hardware and the models. Therefore, models that are heavily used tend to have large hardware requirements.

There are many other bits of hardware that are required to perform machine learning, but they do not represent a significant cost. Examples include network-related infrastructure and compute resources to host ancillary platform components like experimentation management.

[18] https://aws.amazon.com/ec2/pricing/on-demand/

5. Machine Learning in the Organization

Conway's law suggests that in the long term, a software architecture tends towards the structure of the organization.[10] In short, this is because teams of people build solutions for their specific department. But today, data is more important than it has ever been, and organizations have been working hard to make it more accessible than it once was. This opens the potential for centralized or floating teams of data experts to work on solutions throughout the organization.

We wanted to learn more about how trust and safety teams were engaging the services of internal and external consultants to tease out organizational best practices. Our investigation also included speaking to third-party vendors of moderation services and tools. We asked these vendors a very similar set of questions to establish their AI practices. In addition, we obtained a view of their product offerings, investigated how they work, and how they integrate with platforms. Finally, the governance of AI is increasing in importance due to the increasing use of AI.

5.1. Organisational Structure

The structure that an organization creates around its MLOps and AI development activities can affect productivity. How individuals and teams interact impacts the quality and timeliness of AI applications.

5.1.1. What Did We Learn?

Large Organizations

Meta has publicly reported that their trust and safety teams have a dedicated engineering capability.[19] We found in our interviews that other large platforms follow a similar pattern of staffing engineering capability within specific business functions. But since engineering skills are often in demand, some said that they operated a model where "roving" AI experts would join specific teams who wanted to integrate an AI-driven feature. These "consultants" would support them during development and initial deployment but then take a step back. The team would be responsible for ongoing operations.

Small Organizations

Some smaller platforms and vendors stood out as having a separation between the teams that are training models and the teams operating them. The smallest tended to have one small data science team that was responsible for all AI initiatives.

Third-party providers also tended to have more separation between development and operation responsibilities, however, since some are publishing software for others to deploy, perhaps that is not surprising. They are also smaller in size when compared to the likes of the large platforms.

[19] https://www.metacareers.com/life/protecting-and-caring-for-the-facebook-community

5.1.2. Why Is It Important?

For a historical parallel, decades ago software development had a "wall" between software development and operations. Programmers would write software and "throw it over the wall" to operators (for example, sysadmins) who had to run it in production. This led to slow iteration times and more failures because of the communication barriers between these teams. The DevOps movement suggested an alternative by saying that software engineers should also operate their software in production. There might still be different roles within a cross-functional team (e.g., some DevOps-focused engineers and some purely development-focused engineers) but the team assumes long-term responsibility for a given service.

With the development and operation of ML models, there is a parallel between data scientists/ML engineers and DevOps teams - a "new wall," if you will. Ultimately, an efficient organization is better positioned to serve its users, because it can innovate faster and mitigate the impact of failures.

5.2. Governance

Governance is the activities involved to enforce the organization's policies over the management of its applications. For applications driven by AI, organizations tend to focus on data governance and model governance, although other forms exist.[4]

Compliance measures how well an application conforms to the organization's policies.

The processes and tools used to govern are often useful in an *audit*, which takes place when an interested internal or external party wishes to know more about the day-to-day operation of an AI solution.

5.2.1. What Did We Learn?

Asking about AI governance, in the context of trust and safety, was challenging. In our interviews, we wanted to investigate the practices that aim to reduce engineering risk. But we learned less about AI governance than we had hoped, because interviewees generally did not distinguish between AI governance and corporate governance, in which platforms and vendors also have a keen interest.

Common Themes

Governance is an inherently complicated subject because it is tailored to the policies and goals of each organization. This manifests as bespoke systems implementing custom processes. Furthermore, the governance of AI from a technical perspective is often conflated with corporate governance. However, the other parts of this document help to explain and investigate parts of the machine learning lifecycle that ultimately improve the technical posture of a platform or vendor.

The larger platforms suggested that they have governance processes in place but did not go into detail about what this involves. Some talked about approval gates between teams.

Compliance was hardly discussed, except in the context of other sections. For example, we talked about automated testing as part of the development and deployment sections. Auditing

was mentioned by a few organizations, in the context of logging. Most say that they do some form of logging to record access and decisions surrounding a classifier.

5.2.2. Why Is It Important?

Platforms and vendors use technologies described as AI to automate decisions. AI tools are orders of magnitude more efficient than their human counterparts. AI is the only viable solution in some cases because scaling human-centric processes is hard. Organizations that do this well have a competitive advantage, which leads to the fact that the use of AI is increasing.

Within any large organization, there is a diverse array of AI-driven applications. Organizations prefer not to completely centralize the development of AI because that would limit innovation. This leads to disparate teams working on distinct AI projects.

Many teams working on different projects in different ways is a technical risk. For example, if an organization used a dozen software languages, it would need a dozen style guides. Organizations can work to develop policies to mitigate risks. Engineers can then measure the level of compliance and prioritize work to improve it. Organizations can use governance to measure how effective their applications are at meeting their policies.

Ultimately, governance provides a way of implementing risk-related requirements, like legislation or regulation. But it also helps define the organization's core values and targets.

5.3. Use of Third-Party Vendors

Platforms offering services that publish public content could be helped by the tools offered by third-party vendors. As part of the interviews, we asked vendors how they developed their AI solutions to help their customers. We also asked platforms that were using third-party tools what they look for when evaluating them. Evaluation is one way of determining whether third parties are doing what they say they will.

5.3.1. What Did We Learn?

Vendors are already beginning to specialize in specific types of content or user segments. Multiple vendors stressed the importance of context and that an off-the-shelf solution usually required configuration or retraining to work successfully in the domain of a platform. Some offer a suite of tools to handle a moderation process, but others specialize in delivering signals that are integrated with a client's moderation platform. One provider said that they start by asking customers "what's the biggest problem you haven't solved yet," which demonstrates how much customization is required and how consultative the industry is. From a product perspective, the industry has not been commoditized. And as content types and interaction patterns change on the platforms themselves, it's unlikely to become commoditized any time soon.

At the beginning of the project, we were keen to learn how vendors express confidence in their models. We expected that some vendors would publicly expose the performance metrics of their models. However, it transpired that nearly all vendors tend to customize their models to client contexts and requirements. This makes the comparison unfair because the underlying context is different. Performance cannot be independently verified because it requires proprietary context.

This is a surprising finding. Each platform has a different user demographic, and they all have different needs. This demands a unique policy to protect them. These unique policies are the ultimate driver of requirements in a moderation system and lead to different implementations. Some third-party vendors actively articulate the complexity and build platforms to accommodate this hyper-customization.

It was interesting to note the diversity of integration options offered by third-party vendors. There was a mix of pure software-as-a-service (SaaS), hybrid, and fully on-premise deployment options. Many cited that this is what their customers had advocated for. In general, we found that if the data is destined to be public, a SaaS offering is an acceptable solution. Otherwise, the solution should be hosted in a secure environment, alongside the private data.

Most large platforms leverage third-party tools at some level, even if they have significant engineering capability. This makes sense because if there are viable commercial offerings it often makes sense to buy, not build. It was interesting to find that many third-party vendors themselves used third parties to build models and provide extra data. This creates a complex supply chain for the user-facing platforms.

When asked about the experience of integrating with a third-party provider, the responses overwhelmingly stated that they underestimated the time and effort it would take to obtain a satisfactory result. In one instance, a platform said that they had been cooperating for more than a year due to difficulties training the models on data representative of that ecosystem. They also said that a lot of time was spent in a loop between the legal and trust and safety teams and have invested a very large amount of money.

Another challenge we've already mentioned is that in some countries engineers may be prohibited from observing certain types of data. This means that third parties often must make ad-hoc changes to alter the result the model would produce, without being able to see the observation. This requires intimate knowledge of the model and the domain as well as sophisticated observational tools to be able to even attempt this.

5.3.2. Why Is It Important?

There are challenges when evaluating the claims of vendors. If the content vendors are trying to prevent is illegal, how can they justify their claims? How can platforms ensure value for money without the ability to evaluate them?

This type of content is an edge case, however. It is possible to evaluate the harmful but not illegal end of the spectrum, especially if the platform has the data ready to share. But platforms remind us that these systems need training or configuration to match the unique policies and the domain of the data. Speech in an adult domain has a completely different set of social norms compared to a domain where children reside, for example. Therefore, like in any other situation where AI is utilized, there isn't one definitive solution to all problems.

The result is that the need for customization prevents the widespread commoditization of AI-driven moderation services. This also means that small competitors can't simply buy and forget a third-party solution, which may lead to less innovation in the social platform market.

6. Costs

Overall, we found that the cost of third-party solutions is reasonable when compared to the cost of hiring engineers and building AI solutions in-house. There is also a wide ecosystem of providers willing to sell solutions across various industries and use cases.

Both on-premise and SaaS models are available, depending on the provider. On-premise is where you license the software to run in your data centre or cloud account. SaaS is where the provider runs the service for you, and you send requests to it via API.

During the interviews, we asked if vendors could inform us about their pricing structures. This information is presented below in an anonymized and generalized form:

- **Low-end/volume pricing**: These costs seem to be bounded at the low end by the services provided by the big cloud providers (Google/Microsoft/Amazon) whose hosted offerings are in the order of 1 dollar USD per 1000 images[20] (with an image size limitation) or 1000 messages (with a character limitation) for some basic filtering.[21] Some third-party providers with more specialist solutions offer per-user pricing of around five USD.
- **Fixed/annual pricing**: Several third-party providers offer fixed annual pricing based on customizable packages which range from five to six figures depending on options.
- **Enterprise pricing**: For large clients where there might be a request for proposal process, pricing was quoted between six and seven figures, depending on the complexity of the problem.

Overall, these costs are low compared to the cost of developing solutions in-house, where costs can easily exceed one million dollars for a small AI development team. The main drivers of this cost are:

- hiring a team of ML engineers where salaries are 200k USD/year for good engineers - at least in the US,
- collecting the data in-house where large datasets can cost 10k USD to store and process,
- training the models on expensive cloud GPUs (a single g5.48xlarge AWS instance costs around 150k USD/year[22]) or buying in GPU-heavy servers, and

[20] https://aws.amazon.com/rekognition/pricing/

[21] https://cloud.google.com/natural-language/pricing

[22] https://aws.amazon.com/ec2/instance-types/g5/

- operating the AI solutions which typically requires a dedicated MLOps team, with salaries like the above.

While the platforms did not reveal exactly how much they're spending on delivering their in-house ACC models, one platform did mention they were spending millions of USD per year on infrastructure alone, and tens of millions in total, which is consistent with the above. And public descriptions of the investments made by the largest platforms suggest they are spending billions - see Hardware for further details.

Although these costs appear to be significant, it is important to note that the larger platforms have revenues ranging from the sub-billion to a hundred billion USD. Also, the costs of hardware and people are often shared between business units.

7. Conclusions

The use of platforms to distribute content is nothing new. More recently, globally distributed internet-based social platforms have connected people to other people in a way that has never been possible before. Billions of individuals can post user-generated content that is observable to billions of other individuals.

Sidestepping outstanding questions surrounding the legal definitions and responsibilities for the content, it's reasonable to say that some percentage of content will cause harm to some percentage of users. It's also reasonable to assume that platforms are incentivised to create a safe environment for their users. Therefore, the goal of attempting to reduce the risk of harm to the public through moderation is also reasonable.

In practical terms, some platforms have so much content that it might be difficult to moderate at scale without automation. This is where AI technologies can help. The interviewees have reported that automated solutions are actively used in moderation. But these systems are not and may never be perfect. These models are effectively attempting to evaluate the probability and severity of harm, which is incredibly challenging when considering the diversity of a platform's users.

MLOps has evolved to mitigate some operational risks by providing tools and processes that can be customized to suit many situations. Efficient deployment strategies, model testing, drift monitoring, unified development practices, automated and replicable retraining procedures, etc. all help to reduce technical risk by making the AI solution more efficient and reliable. But MLOps isn't a silver bullet either. Instead, MLOps presents a set of ideas to help organizations improve the chances that AI projects will become successful.

Our observations show that although online platforms, vendors and their engineers are actively fighting to improve online safety- many have invested significantly in developing novel processes and technologies that may help improve safety- there are still a number of limitations and difficulties with the use of automated content classifiers in online content moderation. We would like to thank those that participated in the interviews again.

We hope that this report has shed some light on how organizations are using, operating, and investing in AI. In turn, we wish that platforms, vendors, and the whole AI industry can continue raising awareness, to work together to make online interactions as safe as practicable.

8. Glossary

Annotation
The process of adding ground truth to data.

Artificial intelligence (AI)
A moniker that refers to the use of machines to solve problems. Sub-disciplines include machine learning, reinforcement learning, data mining, analytics, etc.

Binary Large Object (BLOB) storage
A type of cloud storage for unstructured data such as media such as images, video, and audio files.

Container
A flexible, portable, encapsulated execution runtime for applications.

CSAM
Child sexual abuse material.

DAG
Directed acyclic graph, typically used to represent data or training pipelines.

Deployment
The process of moving a built artefact into a location where it can be consumed.

DevOps
A philosophy and set of working practices that encourage developers to take ownership of their products.

GitOps
The operation of systems using version control and continuous delivery.

Governance
The enforcement of an organisation's policies over the management of its applications and data.

Ground truth
The verified outcome of an event or situation. It is used for training in machine learning and evaluation.

Infrastructure
The physical or virtualised resources required to host applications.

Machine learning
The use of algorithms that learn to make decisions based on ground truth.

Metadata
Data that describes data.

Machine learning operations (MLOps)
The processes and tools that formalise the development lifecycle of AI.

Model
A (simplified) representation of a (constrained) phenomenon that uses observations to predict outcomes.

Orchestration
Processes and systems that execute tools and applications.

Perceptron
They are the fundamental (single layer) blocks upon which neural networks which are built.

Pipeline
A series of steps that are executed to produce a result.

Personally identifiable information (PII)
Information that can be used to identify a user.

Training
A process to build a model based on data.

Version control
A tool or system that tracks changes, to help audit and revert to previous versions.

9. Bibliography

[1] P. Winder, *Reinforcement Learning*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2020.

[2] P. Strengholt, *Data Management at Scale*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2020.

[3] A. Nielsen, *Practical Fairness*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2020.

[4] E. Eryurek, U. Gilad, V. Lakshmanan, A. Kibunguchy-Grant, and J. Ashdown, *Data Governance: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2021.

[5] F. Provost and T. Fawcett, *Data Science for Business*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2013.

[6] V. Lakshmanan, S. Robinson, and M. Munn, *Machine Learning Design Patterns*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2020.

[7] P. Reznik, J. Dobson, and M. Gienow, *Cloud Native Transformation*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2019.

[8] C. Fregly and A. Barth, *Data Science on AWS*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2021.

[9] N. Gift and A. Deza, *Practical MLOps*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2021.

[10] S. Newman, *Building Microservices, 2nd Edition*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2021.

10. About The Authors

Winder.AI are an AI consultancy based in the UK. They have helped some of the world's largest organisations develop AI solutions and products. Winder.AI continues to provide consulting and development expertise in the fields of machine learning, reinforcement learning, MLOps and AI in general. To find out more please visit https://Winder.AI.

10.1.1. Dr Phil Winder

Dr Phil Winder is a multidisciplinary software engineer and data scientist. As the CEO of Winder.AI, he helps start-ups and enterprises improve their data-based processes, platforms, and products. Phil specialises in implementing production-grade cloud-native machine learning and was an early champion of the MLOps movement. More recently, Phil has authored a book on Reinforcement Learning (RL) (https://rl-book.com) which provides an in-depth introduction to industrial RL for engineers. Phil holds a PhD and M.Eng. in electronic engineering from the University of Hull and lives in Yorkshire, U.K., with his brewing equipment and family.

10.1.2. Luke Marsden

Luke is a technical leader and entrepreneur who developed the end-to-end MLOps platform company Dotscience. He was Kubernetes SIG lead for cluster-lifecycle, creating kubeadm with Joe Beda, and worked on Docker plugins with Solomon Hykes. Luke is also the founder of MLOps Consulting where he specialises in helping product-based organizations scale.

10.1.3. Enrico Rotundo

Enrico is an exceptional young data scientist with 5 years of experience working on a variety of ML and MLOps projects. In the past, Enrico has lived and worked across Europe delivering machine learning projects in domains as diverse as IoT, B2B marketplaces, traffic optimisation, vehicle infrastructure, and fleet management. More recently Enrico has been leveraging his expertise in a variety of ML and MLOps projects for Winder.AI.