International Journal of Latest Engineering and Management Research (IJLEMR)
ISSN: 2455-4847
www.ijlemr.com || Volume 10 – Issue 12 || December 2025 || PP. 12-16

Practical Integration of Machine Learning Models into Automated Content Moderation Systems: Backend Design and Deployment Experience

Bogutskii Aleksandr
Bachelor's Degree, ITMO University, St Petersburg, Russia

Abstract: This article explores the practical integration of machine learning models into backend services for automated content moderation. It examines the important architectural design principles of such systems, including the use of microservices, message brokers, and asynchronous queues to ensure scalability and fault tolerance. The role of machine learning components as auxiliary modules within distributed infrastructures is analyzed, focusing on their functions in content classification, filtering, and prioritization. Particular attention is paid to the engineering aspects of the system interaction, such as request handling, load balancing, containerization, and orchestration.

Keywords: machine learning, automated content moderation, backend infrastructure, microservices, scalability, machine learning integration, fault tolerance.

I. Introduction

Contemporary digital platforms and services process massive amounts of user-generated content every day, from short comments and images to streaming video. At the same time, guaranteeing safe and policy-compliant interactions has become one of the key engineering problems. Manual review is impossible due to the volume of data, and real-time content filtering and evaluation systems are necessary for quick responses. In these circumstances, automatic moderation turns into a stand-alone field of backend infrastructure development, where model accuracy and architecture dependability to process millions of events per second.

The integration of machine learning (ML) into such systems has enabled dynamic content analysis that was previously impossible with simple rule-based filters. However, in practice, ML models rarely serve as the central element of a platform. More often they act as auxiliary components within a distributed pipeline. Their task is to improve the quality and speed of decision-making, as well as to optimize the work of human moderators.

Despite the abundance of research on ML algorithms, the engineering aspects of their practical use in moderation systems are much less explored. Large platforms share fragments of their solutions in technical blogs, yet systematic descriptions of architectural approaches and real-world challenges are rare. Meanwhile, for engineers designing backend services, what matters is not only the model itself but also how it is deployed, interacts with other components, scales, and updates without interrupting the main system. The purpose of this article is to analyze applied approaches to integrating ML models into automatic content moderation systems at the backend infrastructure level.

II. Architecture of Automated Content Moderation Systems and the Role of ML

Every day, the amount of user-generated content increases worldwide, and among it, content that is against platform policies invariably surfaces. Automated content moderation typically consists of filtering inappropriate images and videos and identifying and eliminating unwanted text [1]. The safety and integrity of online platforms are threatened by such content, which also flagrantly violates community guidelines.

Moderation systems use a variety of strategies to tackle these issues. Natural Language Processing (NLP) models are used to identify toxic language, threats, or harassment. Toxic language, threats, or harassment are detected using NLP models. By identifying nudity, violent scenes, or other forbidden elements in photos and videos, computer vision techniques, typically based on Convolutional Neural Networks (CNN), allow the detection of visual violations. Large-scale automated content moderation systems prefer transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), which exhibit noticeably higher accuracy and recall than traditional ML techniques (table 1).

Table 1: Comparison of model performance on hate speech and toxic comment datasets [2]

Model	Dataset	Accuracy	Precision	Recall	F1 score
Naive Bayes	Hate Speech	0.77	0.73	0.70	0.71
Support Vector Machine		0.82	0.80	0.77	0.78
CNN		0.86	0.83	0.82	0.82
Long Short-Term Memory		0.88	0.86	0.84	0.85
BERT Fine-Tuned		0.93	0.91	0.90	0.90
GPT Fine-Tuned		0.92	0.90	0.89	0.89
Naive Bayes	Toxic Comments	0.78	0.74	0.72	0.73
Support Vector Machine		0.84	0.81	0.79	0.80
CNN		0.87	0.84	0.83	0.83
Long Short-Term Memory		0.89	0.88	0.85	0.86
BERT Fine-Tuned		0.94	0.92	0.91	0.91
GPT Fine-Tuned		0.93	0.91	0.90	0.90

The accuracy of identifying policy violations in content that combines multiple modalities is greatly increased when multimodal models are used in complex cases. These models consider both textual elements, like captions and comments, and visual data [3].

A typical architecture of an automated content-moderation pipeline includes a distributed backend consisting of several sequential verification stages. When a new item is submitted for review, the system triggers a predefined chain of checks, each executed by a dedicated microservice. These checks may include analytical filters, ML-based classifiers, auxiliary services, such as Uniform Resource Locator inspection or text translation, as well as manual review when automated verdicts are insufficient.

This design ensures controlled progression through the moderation graph and guaranteed verdict. The verdict itself is non-binary and may encode region-specific restrictions, multiple flags, or additional metadata.

In such an architecture, microservices take on different tasks. For instance, one microservice can be designed specifically to process text, while the other can inspect images, with yet another inspecting videos. These microservices function independently, exchanging data via queue API or API. The data gathered from content evaluation can be saved in a database to facilitate future logging and analytical processes.

For example, on Reddit, a new content filtering system was implemented as a microservice-based pipeline. Incoming posts and comments are published to a Kafka topic, after which one service generates primary features (such as toxicity scores or the presence of offensive language). Another service uses these features to compute an integrated content risk score via a ML model, and the final component decides whether the material should be automatically filtered, that is, removed from the public feed and placed into a moderation queue (fig. 1).

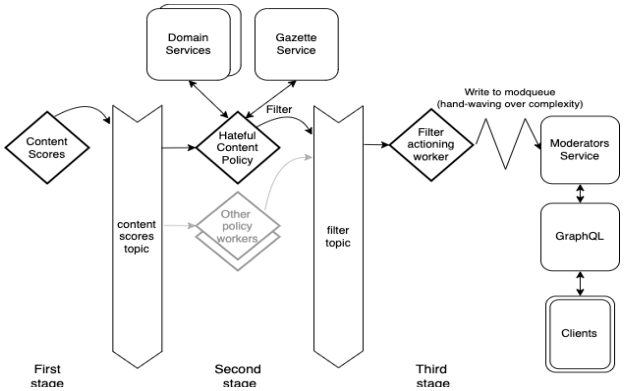

Fig. 1: Microservice architecture of Reddit's automated content moderation pipeline [4]

In automated moderation, ML models have become central components that substantially improve traditional rule-based systems. They typically perform three primary functions: content classification, violation filtering or blocking, and prioritization of items for human review. Classifiers based on ML can be trained on large labeled datasets, as it allows them to detect complex violation patterns that are inaccessible to static rule-based filters. A model may classify a comment as neutral, toxic, extremist, or spam with a certain confidence level. Based on this classification, the system decides whether to automatically hide or remove the content. This is precisely how major platforms operate. For instance, YouTube (Google LLC) reported that among all deleted

comments, 99.5 % were detected automatically by ML algorithms, while only 0.5 % were flagged by human users [5].

Two main approaches can be distinguished in the design of automated moderation models. The first is the classical pipeline, in which a platform's moderation policies are translated into a data annotation process followed by model training. The second is the Policy-as-Prompt approach, where the moderation policy itself is encoded directly as a prompt for a large language model (LLM) – fig. 2.

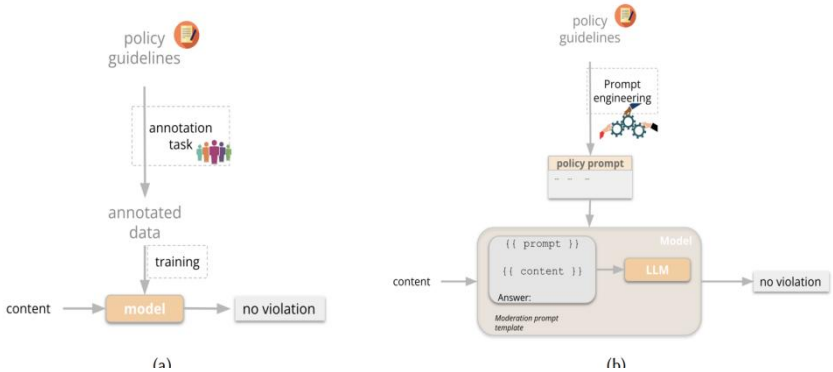

Fig. 2: Approaches to content moderation: (a) – traditional pipeline, (b) – policy-as-prompt [6]

At the same time, ML models assist in prioritization. When the model's confidence is low or when contextual nuances are required, the content remains on the platform but is flagged for human moderator review. Different systems employ AI not only to remove obvious violations but also to guide human moderators' attention toward the most potentially harmful cases. In this way, ML functions as a first-layer filter, rapidly processing massive data streams, automatically blocking clearly violating content. Without such an intelligent layer, no modern platform could effectively handle the scale of today's content flows, as ML-based systems enable real-time moderation.

The initial rule-based automated moderation systems were essentially hardcoded filters. Lists of prohibited words or regular expressions for identifying offensive phrases are a few examples of this strategy. Rule-based systems are quite simple to comprehend and apply. Moderators explicitly define unacceptable content, and each triggered rule is transparent.

Systems enhanced by ML eliminate many of the drawbacks inherent in traditional rule-based moderation. Instead of manually defining all criteria, they employ trainable models that automatically extract the features of undesirable content by learning from large datasets. As a result, these systems demonstrate much higher adaptability, as they can detect emerging trends and new types of policy violations after fine-tuning on fresh data.

III. Applied integration of ML models into backend services

There are several ways to integrate ML models into a platform's backend. One of the most common approaches is synchronous inference via «model-as-a-service» (MaaS) architecture. In this configuration, other backend modules send requests for predictions to the model, which is deployed as a web service with a REST or gRPC API. In this instance, user actions trigger real-time content verification. The moderation service's REST API can be called by the frontend or backend when a user submits a comment. The moderation service then returns a toxicity score, if the score is higher than a predetermined threshold, the comment is immediately blocked. The operational architecture of most MaaS methods is shown below (fig. 3).

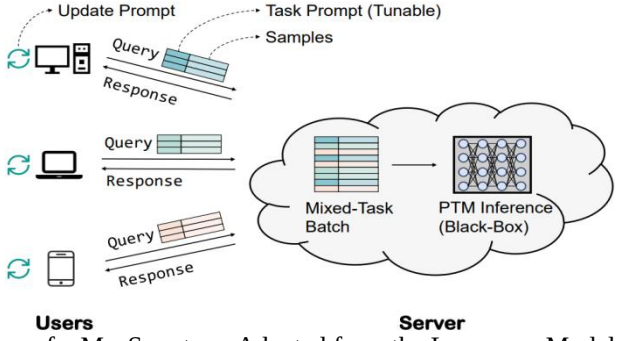

Fig. 3: General architecture of a MaaS system. Adapted from the Language-Model-as-a-Service (LMaaS) concept [7]

This approach's primary benefits are its conceptual simplicity and instant feedback, as developers can incorporate the API call in the same way they would any other external function. The request's synchronous nature, which raises latency for the end user, is the disadvantage. While the model processes the input in this scenario, a post submission might be a little delayed.

Overall, API-based inference is well suited for relatively simple moderation tasks that require immediate decision-making (e.g., blocking obviously prohibited content before publication). Many public moderation tools operate in this way, for instance, Google's Perspective API provides an open REST endpoint that returns a numerical toxicity score in response to a submitted text [8]. This serves as a classic example of synchronous REST-based NLP model integration within external platforms.

Another approach is asynchronous processing within an event-driven architecture. In this design, content publication does not wait for the model's response. Instead of this, the model processes the data in parallel, at its own speed, after an event is sent to a stream processing system or message broker.

Building multi-stage moderation pipelines is made possible by event-driven integration, which permits each model to function independently and concurrently. The asynchronous architecture increases system resilience, in which temporary failures or model overloads merely result in longer queues rather than complete service disruption.

Generally, ML integration can also be classified according to the type of model invocation. Inference on-demand refers to the case already discussed, where each object (such as a comment or image) is analyzed immediately upon creation. Batch inference applies the model to an accumulated set of data. Streaming inference is conceptually close to the event-driven approach but assumes continuous processing of an unbounded data stream (table 2).

Table 2: Inference modes in automated moderation systems

Inference mode	**Description**	**Key characteristics**	**Typical application cases**
On-demand	Synchronous model invocation for each content item. Blocks execution flow until a result is returned. Simple to integrate, may increase response latency.	Real-time processing; direct user interaction. Higher latency possible due to synchronous calls.	Reaction to real-time actions. Used for instant blocking of prohibited content. Effective with moderate request volume
Batch	Asynchronous processing of accumulated content batches. The model runs periodically on grouped data and outputs results for the entire batch.	Optimized throughput; reduced load on model instances. Not suitable for urgent responses.	Periodic content audits. Also used for A/B testing of new models on historical datasets.
Inference mode	**Description**	**Key characteristics**	**Typical application cases**
Streaming	Asynchronous continuous pipeline for real-time event processing. The model is integrated into a streaming platform, continuously receiving data.	High scalability; near-zero latency; event-driven architecture.	Continuous data streams. Suitable for detecting violations in real-time.

Integrating an ML model involves deciding when to invoke it and also how to deploy it in production. This is where modern infrastructure technologies such as containerization and orchestration hold a significant place. Containerization (Docker) has become the de facto standard for packaging ML services. The model with all its dependencies, including frameworks, libraries, and weight files, is bundled into a container.

Using containers allows each model and its environment to remain isolated. Another major advantage is scalability, since container images can be easily replicated. On large-scale platforms, models are typically served by clusters containing dozens or even hundreds of instances to handle large volumes of parallel requests. For example, Unitary, a startup providing AI-powered video moderation, processes up to 26 million videos per day by deploying its system on a Kubernetes cluster with over 1000 nodes, automatically scaling the number of model containers as the load increases [9].

Orchestration (Kubernetes) simplifies the management of such clusters by monitoring the state of containers, restarting failed ones, and distributing workloads across nodes. Finally, load balancing secures the even distribution of incoming requests among multiple model instances.

Overall, integrating ML models into backend systems is a multifaceted engineering task. It requires consideration of the architectural, the infrastructure and the operational levels. A well-designed automated moderation system combines these approaches. It applies various algorithms where complex content understanding is essential, uses asynchronous queues and microservices to scale to millions of events, and incorporates tools for full-lifecycle model management.

IV. Conclusion

The integration of ML models into automated content moderation systems has become one of the most significant directions in the evolution of modern platform backend infrastructures. The use of ML substantially expands the functional capabilities of such systems, transforming them from static filters into adaptive, context-aware tools for data analysis. At the same time, the key factor of success lies not in the complexity of the model itself, but in how processing queues are organized or events are routed.

The experience of major technology companies shows that an effective moderation architecture is built around the flexible integration of ML components with the rest of the backend ecosystem. Models act as intelligent first-layer filters, while the infrastructure ensures their stable operation, continuous updates, and scalability under high load. Thus, the applied use of ML in content moderation should be viewed primarily as an engineering challenge – the task of constructing a fault-tolerant, maintainable, and transparent backend system in which ML serves as a means of improving efficiency rather than an end in itself.

References

[1] G.Andreev. Integration of UGC into traditional digital media and its impact on media platform business metrics: from retention rate to user LTV assessment, *Cold Science, (19),* 2025, 4-15.

[2] Z.Khan. Natural Language Processing Techniques for Automated Content Moderation,*International Journal of Web of Multidisciplinary Studies, 2(2),* 2025, 21-7.

[3] J.Yuan, Y.Yu, G.Mittal, M.Hall, S.Sajeev, M.Chen. Rethinking multimodal content moderation from an asymmetric angle with mixed-modality, *In Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2024, 8532-8542.

[4] Building Better Moderator Tools / Reddit // URL: https://www.reddit.com/r/RedditEng/comments/uly8s4/building_better_moderator_tools/ (date of application: 02.11.2025).

[5] YouTube Community Guidelines enforcement visible changes / Google Support // URL: https://support.google.com/transparencyreport/answer/9198203?hl=en (date of application: 05.11.2025).

[6] K.Palla, J. L. García, C. Hauff, F. Fabbri, A. Damianou, H. Lindström, D. Taber, M. Lalmas. Policy-as-prompt: Rethinking content moderation in the age of large language models, *In Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, 2025, 840-854.

[7] T.Sun, Y.Shao, H.Qian, X.Huang, X.Qiu. Black-box tuning for language-model-as-a-service, *In International Conference on Machine Learning*, 2022, 20841-20855.

[8] M. S. Alexiou, J. S. Mertoguno. Not-in-Perspective: Towards Shielding Google's Perspective API Against Adversarial Negation Attacks,*In2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2023, 1-8.

[9] Unitary Scales AI Moderation to 26 Million Videos Daily with Amazon EKS / AWS Amazon // URL: https://aws.amazon.com/solutions/case-studies/unitary-eks-case-study/ (date of application: 10.11.2025).