Auto Content Moderation in C2C e-Commerce

Shunya Ueta, Suganprabu Nagarajan, Mizuki Sango
Mercari Inc.

Abstract

Consumer-to-consumer (C2C) e-Commerce is a large and growing industry with millions of monthly active users. In this paper, we propose auto content moderation for C2C e-Commerce to moderate items using Machine Learning (ML). We will also discuss practical knowledge gained from our auto content moderation system. The system has been deployed to production at Mercari since late 2017 and has significantly reduced the operation cost in detecting items violating our policies. This system has increased coverage by 554.8 % over a rule-based approach.

1 Introduction

Mercari[1], a marketplace app is a C2C e-commerce service. In C2C e-commerce, trading occurs between consumers who can be both sellers and buyers. Unlike standard business-to-consumer (B2C) e-commerce, C2C buyers can buy items that are listed without strict guidelines. However, C2C e-commerce has the risk of sellers selling items like weapons, money and counterfeit items that violate our policies, intentionally or unintentionally. Our company needs to prevent the negative impact such items have on buyers, and we also need to comply with the law regarding items that can be sold in our marketplace. In order to keep our marketplace safe and protect our customers, we need a moderation system to monitor all the items being listed on it. Content moderation has been adopted throughout industry by Microsoft [3], Google [2] and Pinterest [1] in their products. Rule based systems are easy to develop and can be quickly applied to production. However the logic of rule based system is hard to manage and it is difficult to cover the inconsistencies in (Japanese) spellings. It is also infeasible for human moderators to review all the items at such a large scale.

Machine Learning (ML) systems can overcome the limitations of rule based systems by automatically learning the features of items deleted by moderators and adapting to spelling inconsistencies. The moderators review the items predicted as positive by our system and we continuously re-train our models with the latest annotated data. Figure 1 shows an overview of our content moderation system. The moderator functions as a Human In The Loop to review the results of ML inference, rule based logic and reported items from customers, and helps regulate the items that are deleted.

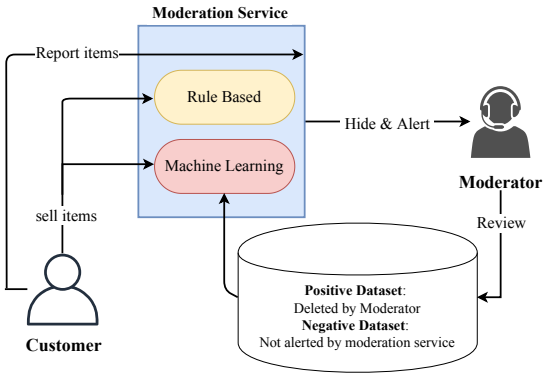

Figure 1: C2C e-Commerce content moderation system overview

The main contributions of this paper are (1) implementing multi-modal classifier models for imbalanced data in the wild (2) introducing and updating models in production (3) and preventing concept drift. In this paper, we also discuss more specifics about how we moderate items in our marketplace using ML.

2 Method

2.1 Model Training

Item listings in our marketplace consist of multi-modal data (e.g. text, image, brand, price), so we use multi-modal models to improve model performance. All models are trained in a one-vs-all setup since alerts from different models can over-

[1] Mercari. https://about.mercari.com/en/

lap. One model corresponds to one violated topic. One-vs-all models can also be easily re-trained and deployed independently. We made a pipeline using Docker [9] container-based workloads on a Kubernetes [6] cluster for model training workloads. We write manifest files containing requirements like CPU, GPU and Storage which are deployed using this pipeline.

For ML algorithms, we used Gated Multimodal Units (GMU) [5] and Gradient Boosted Decision Trees (GBDT) [7]. GMU potentially provides the most accuracy using multi-modal data. GBDT is efficient to train and use for prediction when training dataset size is not large. We train the GMU models using PyTorch [10] and deploy them using PyTorch to ONNX [4] to Caffe2 [8] conversion.

The system then automatically evaluates the new model against the current model in production using offline evaluation (Sec. 2.2).

2.2 Evaluation

We propose offline and online evaluation to avoid concept drift [11].

Offline evaluation. We use the precision@K of the model as our evaluation metric since it directly contributes to the moderator's productivity, where K is the bound on the number of alerts in each violated topic and is defined by the moderation team. We evaluate the new model based on back-tests (the current model's output on test data is known). The back-tests guarantee that the new model is not worse than the current model which prevents concept drift. However, this test is biased towards the current model because the labels were created based on it. Thus, we also evaluate online for a predetermined number of days by using both models for predictions on all items.

Online evaluation. In our scenario, A/B testing is slow for decision making because the number of violations is much lower than valid item listings. As a result, A/B testing can take several months. This results in concept drift occurring and does not meet our business requirements. For faster decision making, we deploy the current and new models in production, and both of them accept all traffic. We set the thresholds of current and new models to alert half the target number each in that violated topic. The current and new model send $\frac{K}{2}$ alerts each to the moderator. If the new model has better precision@$\frac{K}{2}$ during online evaluation, we deprecate the old one and expand the new model to the target number of alerts. Table 1 shows the relative performance gain of the model based on precision@K. It shows our back-test reflecting the performance in production.

3 System Design

Figure 2 describes the system architecture. Our deployments are managed using Horizontal Pod Autoscaler which helps to maintain high availability and cut down production costs. The system has a proxy layer which gets messages from a queue and makes REST calls to the prediction layer. The prediction layer is responsible for preprocessing and inference, and returns a prediction result to the proxy layer. The proxy layer aggregates the responses from all the models and publishes messages for those items predicted as positive by at least one model, to a different queue where these messages are then picked up by a worker, and sent to the moderators for manual review of items. In online and offline evaluation, the proxy layer logs the predictions from all models and these logs are exported to a Data Lake.

Table 1: Percentage gains of GBDT and GMU compared to Logistic Regression in offline and online evaluation on one violated topic.

Algorithms	Offline	Online
GBDT	+18.2 %	Not Released
GMU	**+21.2%**	**+23.2%**

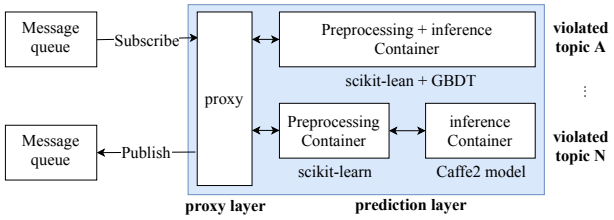

Figure 2: Auto content moderation system architecture. **GBDT**: One container contains the preprocessing and inferences. **GMU**:GMU has two containers. i) preprocessing. ii) inference using Caffe2.

4 Conclusion

Content moderation in C2C e-Commerce is a very challenging and interesting problem. It is also an essential part of services providing content to customers. In this paper, we discussed some of the challenges like new ML model introduction into production and how to efficiently prevent concept drift based on our experience. Our Auto Content Moderation system successfully increased moderation coverage by 554.8 % over a rule-based approach

Acknowledgments

The authors would like to express their gratitude to Abhishek Vilas Munagekar and Yusuke Shido for their contribution to this system and Dr. Antony Lam for his valuable feedback about the paper.

References

[1] Getting better at helping people feel better. https://newsroom.pinterest.com/en/post/getting-better-at-helping-people-feel-betterl. Accessed on 2020.04.14.

[2] Google maps 101: how contributed content makes a more helpful map. https://www.blog.google/products/maps/google-maps-101-how-contributed-content-makes-maps-helpful/. Accessed on 2020.04.14.

[3] New machine-assisted text classification on content moderator now in public preview. https://azure.microsoft.com/es-es/blog/machine-assisted-text-classification-on-content-moderator-public-preview/. Accessed on 2020.04.14.

[4] Open neural network exchange format (onnx). https://github.com/onnx/onnxl. Accessed on 2020.02.24.

[5] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*, 2017. https://arxiv.org/abs/1702.01992.

[6] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. Borg, omega, and kubernetes. *Queue*, 14(1):70–93, 2016.

[7] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[8] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014.

[9] Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[11] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big data analysis: new algorithms for a new society*, pages 91–114. Springer, 2016.