MLTox: A Cloud-Based Machine Learning Toxicity Detector

Srihari Mohan
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
smohan12@jhu.edu

Benjamin Pikus
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
bpikus16@jhu.edu

Lalit Varada
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
lvarada1@jhu.edu

Parth Singh
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
psingh1@jhu.edu

May 10, 2019

Abstract

MLTox is a platform for supporting real-time toxicity analysis of large-scale textual data. The system is designed from the ground up with a focus on scalability, building on top of technologies such as Apache Kafka for stream processing, Apache Spark for distributed batch processing and machine learning (ML), Kubernetes for automated application deployment, and Google BigQuery for interactive queries over our data store. While existing solutions to toxicity detection permit a message-by-message analysis, MLTox provides a suite of analyses, packaged in a user-facing and continually refreshed public dashboard, that leverages state-of-the-art machine learning for toxicity detection with XGBoost, real-time toxicity forecasting with Seasonal Autoregressive state space models (SSMs), and the Local Correlation Integral (LOCI) for anomaly detection. We take inspiration from FiveThirtyEight, the data-driven news website, providing the public with statistical analyses of how the health of our online discourse evolves over time. We detail the architecture of MLTox, explore design decisions in its engineering and use of machine learning, and evaluate its effectiveness in pipelining upwards of 100,000 tweets per day through the system.

1 INTRODUCTION

The explosion of social media through the ubiquity of platforms like Facebook, Twitter, and YouTube has afforded society the prospects of free and open speech on an unprecedented scale. Yet, one look into The Comment Section, seemingly populated by denizens from the pits of Hell, is often all it takes to shatter one's faith in our stewardship of such privilege. The toxicity that permeates online discourse is assuredly apparent to anyone who has spent enough time on Twitter, where everything from politics to entertainment is polarized. "I HATE TRUMP! And you're all going to prison, Creep #trumpforprison", writes one Twitter user in a tweet critical of U.S. President Donald Trump. Not to be outdone, another responds in retribution, "how detached from reality are you demonrats [sic] at this point". In response to fan backlash over the recent Disney film Star Wars: The Last Jedi, one Twitter user writes "#thelastjedisucks thanks for showing us men that we can't do anything right without a female directing us". Within minutes, two others reply "LOL. Look at the pathetic manchildren in the comments proving [us] right. Star Wars fanboys are such fragile beings". The authors preface this research with the intention not to take sides in one form or another, but to rather assess critically toxicity as it manifests in online discourse on all sides of the political spectrum. Developing systems to assess the toxicity levels around real-time discussions is thus an interesting and useful social problem. Our intention is

not to moderate online discussions, but to simply provide a platform to assess trends in the health of our discourse as it evolves over time. This work is provocative on a philosophical level, for exploring toxicity entails introspection of human tendency towards deindividuation and the Troll Culture. Yet, it is just as fascinating from a technical perspective; architecting a system amenable to processing large quantities of textual data in real-time necessitates an exploration of scalable technologies at each layer of the application stack. Doing so successfully requires the integration of state-of-the-art machine learning and natural language processing.

Twitter is responsible for generating upwards of 500,000,000 unique messages per day. The advent of open-sourced cloud technologies such as Hadoop, Spark, and Kafka and their availability on infrastructure-as-a-service platforms such as Google Cloud Platform (GCP) permits the analysis of such big data at previously untenable levels. Narkhede et. al demonstrate in their initial whitepaper that Kafka, a high-throughput and low-latency software platform for stream processing, permits upwards of 2 million writes/second on as few as three commodity machines [1]. Zaharia et. al illustrate that Spark, a distributed cluster computing framework optimized through in-memory processing, can query a 39 GB dataset with sub-second response time [2]. Processing such large volumes of textual data in real-time requires developing on the cloud, as supporting the compute resources necessary for this computation is likely cost-prohibitive to small development teams like ours. Delegating responsibility of compute and storage to a service provider such as GCP affords us with the freedom to invest both time and money into developing the most critical features of our application that we describe below.

1.1 Existing Approaches

Toxicity detection is a branch of sentiment analysis, itself an offshoot of natural language processing and machine learning. Approaches to sentiment analysis have become popularized through open source contributions such as scikit-learn, nltk, and Google's TensorFlow. With the advent of social media, developer APIs on Twitter have facilitated near-unfettered access to real-time tweets and a bevy of projects aimed at emotion detection and sentiment analysis of them. As a consequence, current attempts at tagging the toxicity levels around online discussions have been fruitful. The Perspective API, a collaboration between Google's Jigsaw and Counter Abuse Technology teams, is the first attempt at large-scale toxicity detection. It seeks to explore the strengths and weaknesses of using machine learning as a tool for online discussion. Although the internal algorithms Perspective leverages are proprietary, users send HTTP requests through the API with a string of text and are returned toxicity scores between 0 and 1, reflective of how toxic that message was. Not relegated to the corporate sphere, academia has explored sentiment analysis on large-scale text data even more comprehensively. Agarwal et. al delve into the application of prior polarity scoring and support vector machines (SVMs) in classifying text sentiment as either positive or negative in [3]. Prior polarity scoring is an approach to assigning words a positive or negative score out of context. These scores are collected into a compact feature representation before being passed to machine learning models such as SVMs. Xu et. al apply variational autoencoders and Long Short Term Memory (LSTM) networks, a variant of recurrent neural networks (RNNs), to learn intermediate sequence embeddings for tweets used in sentiment analysis pipelines. The application of this network architecture has yielded an accuracy rate of 92.8% on benchmarked text classification datasets. Liu et. al in [4] and Timmaraju et. al in [5] demonstrated classification accuracies of 91.8% and 86.5% on similar benchmarked (IMDB) datasets leveraging variants of RNNs with multi-task learning and transfer learning, respectively.

1.2 Our Contribution

In this paper, we describe MLTox, a cloud-hosted machine learning toxicity detector. Our contribution is unique in that it extends existing approaches to ML toxicity detection by broadening its use-case. Namely, existing approaches focus exclusively on detecting toxicity on a message-by-message basis. MLTox, on the other hand, is a self-contained suite of tools to assess two additional aspects of toxicity detection: forecasting and anomaly detection. The inclusion of both distinguishes our solution through the integration of both real-time and batched processing in the machine learning pipeline. MLTox supports an analysis of the toxicity trends around a particular topic, both post-hoc and forecasted, distinguishing our pipeline from existing solutions that do not. MLTox uses Apache Kafka for continuous streaming of user tweets, Apache Spark for highly parallel and distributed machine learning, and Google's BigQuery and Kubernetes Engine for scalability on the cloud. The system's adoption of a cloud-first architecture at each step of the pipeline emphasizes scalability in the presence of continually increasing message volumes, which existing solutions do not explicitly address.

1.3 Intuition behind our construction

We seek to develop a platform that affords individuals around the world insights into real-time trends in toxicity levels around politics. We chose politics as a test-case topic for one reason: interest in the polarization of the political sphere

is both abundant and well documented. Articles such as When did Everything Get So 'Toxic' in the New York Times connects the Internet's promotion of virality to its toxicity [6]. Likewise, Brady et. al explore the phenomenology of how "emotion shapes the diffusion of moralized content in social networks" [7]. That such polarization is central to our national discourse is likely unsurprising to most Americans. According to Twitter, political movements such as the March for our Lives and the NFL anthem protests, and political figures such as Donald Trump and Barack Obama, were among the most tweeted about in 2018 [8].

Before outlining the rest of this paper, we describe a general use-case of the system. MLTox provides a user-facing dashboard with up-to-date toxicity analyses on politics-related tweets in the United States. The dashboard displays both a post-hoc toxicity trajectory with labeled anomalies and a forecasted trajectory. The learned toxicity scores and an aggregated summary of the volume of tweets processed in the analysis are provided for a user to explore. Rather than use existing solutions like the Perspective API to analyze tweets manually on a message-by-message basis, an individual can visit our publicly hosted URL to access MLTox's visualization and continually refreshed analyses. The dashboard continually updates its view in real-time with data streamed directly from Kafka into BigQuery.

1.4 Outline of this work

The rest of this paper proceeds as follows. In Section II, we formally define the MLTox architecture in each step of the pipeline. In Section III we explore the application of novel approaches to state space modeling and the local correlation integral to toxicity forecasting and anomaly detection. Next, we evaluate our system's utility in its stated goal of real-time toxicity analysis, in comparison to existing solutions. We finally close with a discussion of future work to transform MLTox into a more comprehensive platform for online toxicity analysis.

2 MLTOX ARCHITECTURE

The engineering infrastructure behind MLTox leverages cloud technologies at each layer of the application stack to maximize scalability in the presence of large volumes of textual data streamed through the system in real-time.

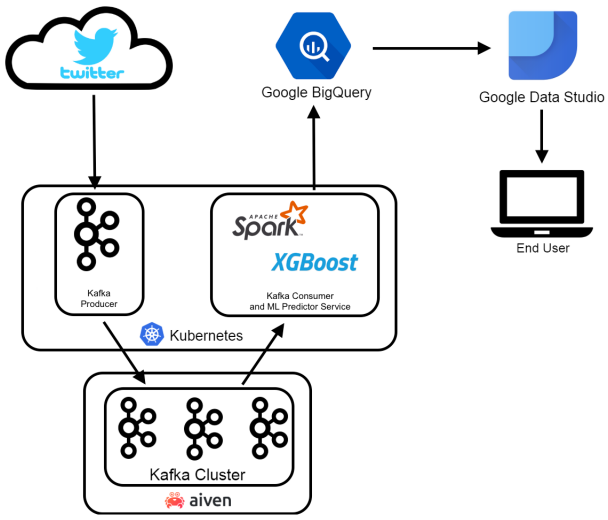

Figure 1: A summary of the MLTox architecture and pipeline.

2.1 MLTox: A Bird's Eye View

MLTox streams upwards of 100,000 tweets per day through a Kafka cluster hosted on Aiven. A microservice for the Kafka producer and Kafka predictor engine are spun up as containers run on Kubernetes. The producer microservice repeatedly fetches batches of politics-related tweets every 100s using the tweepy API, before publishing them to brokers in the Kafka cluster. The predictor microservice hosts our XGBoost ML predictor, seasonal autoregressive SSMs time-series forecaster, and LOCI anomaly detector. The microservices are hosted on Kubernetes and write their output in real-time to Google BigQuery. We finally connect BigQuery to Google Data Studio to generate a user-facing

and continually refreshed public dashboard of MLTox's learned toxicity trajectory, anomaly detection, and toxicity forecasting. The architecture is summarized in 1.

2.2 Apache Kafka

Apache Kafka is a distributed messaging system initially developed for collecting and delivering high volumes of log data with low latency. The technology has grown today into a full-fledged stream processing platform. In Kafka, a stream of messages of a particular type is defined by a topic. A client interacts with the Kafka system through the abstractions of the producer and the consumer. The producer is a client that publishes messages to a topic, while the consumer subscribes to a set of servers in the system called brokers, pulling messages conformant with the specifications of the Kafka API. Messages are serialized into a payload of bytes before being published to the Kafka servers. Kafka appends messages to the last seen segment file (approximately 1GB each) for each partition of a topic. The messages (in MLTox JSON representations of a tweet and its timestamp) do not have message ids, helping the system avoid the need to maintain auxiliary, seek-intensive random-access index structures between message ids and the messages themselves [1]. The consumer efficiently pulls up-to-date messages from the server by making asynchronous pull requests to the broker to maintain a buffer of sequential data ready for the consumer to fetch. Kafka leverages the notion of a stateless broker to mitigate much of the networking costs associated with network access for the consumers.

In our architecture, both the producer and consumer are abstracted as microservices spun up in Docker containers on Google Kubernetes Engine (GKE). The producer repeatedly fetches batches of tweets every 100s using the tweepy API, pushing them to brokers in MLTox's Kafka cluster hosted on Aiven's Kafka as a Service platform. The consumer iteratively fetches politics-related tweets in accordance with the Kafka client API, serializes a JSON message containing the tweet and its timestamp, before streaming the messages to a mounted disk in the Kubernetes pod on GKE. The consumer finally makes a call to the MLTox core predictor engine. The predictor engine reads the data mounted on the pod's local disk and routes it through the XGBoost predictor, forecaster, and anomaly detector before writing the output to BigQuery.

2.3 Apache Spark

We trained the toxicity predictor to regress toxicity scores on tweets using distributed XGBoost on PySpark MLLib. Spark is a cluster computing framework that supports distributed processing for a class of applications that reuse a working set of data across multiple parallel operations (including but not limited to many iterative machine learning algorithms). Spark supports these applications while retaining both scalability and fault tolerance through its abstraction of read-only collections of objects partitioned across clusters called resilient distributed datasets (RDDs). Spark supports the caching of RDDs in memory across machines in multiple parallel operations, achieving fault tolerance through the notion of lineage (e.g. if a partition is lost, the RDD knows enough about how it was derived to rebuild that partition). Spark MLLib is Spark's machine learning library, providing high-level tools for ML algorithm development, feature extraction, pipelining, and model persistence using RDDs [2]. The toxicity model learned offline is packaged into the container as part of the Docker build before deployment on Kubernetes. The utility of batch processing of tweets is useful in our context. As we stream high volumes of tweets through the system, Spark permits highly distributed parallel processing of this data through the predictor engine. If we scale up our application beyond the limits of our free trial quotas for Twitter's Developer API, we could feasibly stream from Kafka a batch of tens of thousands of messages through the system on the order of seconds. Distributed processing of these messages through our predictor engine in the Spark cluster allows us to feasibly update our UI in real-time.

2.4 Kubernetes

Kubernetes is an open-source platform for container orchestration and managing automated application deployment. Kubernetes facilitates the adoption of microservices, in that an overall application can separate services in accordance with crisp APIs that enable decentralized updating and scale up on the cloud. Kubernetes abstracts away the need for a user to manage scale up of an application by resolving DNS (for discovery) and provisioning new machines to be added to the cluster dynamically. Kubernetes' use of lightweight containers over fully virtualized machines permits faster startup, provisioning of machines to the cluster, and improved scalability. Containers spun in pods on a host machine all share the same host OS, rather than provisioning resources to copy the host OS themselves as in a VM. This means containers are lightweight in relation to VMs, such that more of them can be run on a single machine.

MLTox's use of Kubernetes affords our system both scalability and availability. Currently, 20 replicas of our predictor engine microservice are deployed onto pods on GKE, providing our application 24/7 availability.

2.5 Google BigQuery and Google Data Studio

Google BigQuery is an enterprise data warehouse hosted on GCP. It permits exceptionally fast interactive queries over massively large datasets. MLTox writes its toxicity predictions, detected anomalies, and forecasted trajectories into tables in BigQuery. The system currently writes upwards of 100,000 records per day into BigQuery, making use of the data warehouse reasonable in our context. We finally connect BigQuery to Google Data Studio to generate real-time visualizations of the toxicity trajectory, forecasting, and anomaly detection.

3 Machine Learning

MLTox uses machine learning to power its predictor microservice. Namely, machine learning is used to provide learned toxicity trajectories for the data streamed through the system, toxicity forecasts, and anomaly detection. In the training process we applied Google's Perspective API to a held-out training set of politics-related tweets fetched using tweepy to generate the training labels. We recognize that this is a point of contention. A model that learns to predict Perspective's toxicity scores cannot offer more robust toxicity predictions than Perspective. However, we deemed it a better use of our time for this research to build out the core infrastructure, rather than hand label thousands of training examples ourselves (which we likely will in future iterations of our service). During preprocessing, tweets are tokenized, stop words are removed, and TF-IDF is applied as a normalization technique before passing the resulting features to the XGBoost model (predictor).

3.1 XGBoost

MLTox currently uses XGBoost, a popular implementation of gradient boosted trees for the regression task (learning the continuous toxicity scores). As a boosting framework, XGBoost is an ensemble model that consists of a set of classification and regression trees (CART). These decision trees, denoted weak classifiers in the context of the boosting framework, recursively partition input examples into bins. XGBoost ensembles many such trees by requiring each to output a prediction for a given example, before weighting these predictions with respect to the confidence of each output and returning a final label[9]. Mathematically, we can write our model in the form:

$$\hat{y_i} = \sum_{k=1}^{K} f_k(x_i),\ f_k \in F$$

Where K is the number of trees in the ensemble, f is a function in the functional space F (here a decision tree), and F is the set of all possible CART trees. The objective function for the boosted ensemble is necessarily then (where G is a regularization function over the parameters of the weak learner).

$$J(\theta) = \sum_{i}^{n} l(y_i, \hat{y_i}) + \sum_{k=1}^{K} G(f_k)$$

Given this objective, XGBoost seeks to minimize it by learning optimal parameters of its candidate weak learners f_i. We use an additive strategy, iteratively building upon weak learners from each round of the training procedure to learn the ensemble gradient boosted tree (GBT). We first generate iterative predictions for each training example.

$$\begin{aligned}
\hat{y_i}^{(0)} &= 0 \\
\hat{y_i}^{(1)} &= f_1(x_i) = \hat{y_i}^{(0)} + f_1(x_1) \\
\hat{y_i}^{(1)} &= f_1(x_i) + f_2(x_i) = \hat{y_i}^{(1)} + f_2(x_i) \\
\hat{y_i}^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y_i}^{(t-1)} + f_t(x_i)
\end{aligned}$$

Each model the upweights the contribution of misclassified examples in one round on the model parameters of the model to be used in the next round. The complete training XGBoost training pipeline follows:

$$\begin{aligned}
F_0(x) &= \arg\min_{\theta} \sum_{i=1}^{n} L(y_i, \theta) \\
r_{im} &= -\alpha(\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)})_{F(x)=F_{m-1}(x)} \\
F_m(x) &= F_{m-1}(x) + \theta_m h_m(x)
\end{aligned}$$

3.2 Anomaly Detection

MLTox leverages the Local Correlation Integral (LOCI) for anomaly detection. The pipeline learns the toxicity scores for a batch of tweets pipelined through Spark and aggregated by the minute. Anomaly detection is applied over these learned toxicity scores to detect abnormal fluctuations in toxicity given the current trend. LOCI is an attempt at fast outlier detection. It computes the exact MDEF and σ_{MDEF} values for all objects, and reports an outlier whenever MDEF (multi-granularity deviation factor) is more than three times larger than σ_{MDEF} for the same radius. We refer readers to the official paper for more details on the method [10].

3.3 Toxicity Forecasting

MLTox implements toxicity forecasting through the integration of Seasonal Autoregressive state space models (SSMs). SSMs model observations as having been generated stochastically from an underlying continuous hidden state:

$$y_t = Z_t \alpha_t + d_t + \epsilon_t$$
$$\alpha_t = T_t \alpha_{t-1} + c_t + R_t \eta_t$$

Here, y_t is the observation vector at time t, α_t is the latent continuous state, and ϵ and η are normally distributed random noise with mean 0. The Seasonal Autoregressive SSM is an extends the general SSM functionality by including its estimation of additive and multiplicative seasonal effects, alongside potentially arbitrary polynomials [11]. To optimize the parameters for our SSMs, we performed a grid search on the actual data.

4 EVALUATION

We deployed MLTox using Google Data Studio. The user-facing, continually refreshed public dashboard can be accessed at `https://datastudio.google.com/u/0/reporting/19e6NORcUonzqnsiaQ7VTVPFzWML9eW_3/page/VgD`. The dashboard displays the MLTox learned toxicity trajectory in the upper left, alongside the forecasted toxicity trajectory in the upper right. Included in the dashboard are summary tables that mark whether a particular hour of online discussion was marked anomalous (bottom-left), as well as summary statistics indicating the volume of tweets streamed through the system per hour. A snapshot of the dashboard is provided in figure 2.

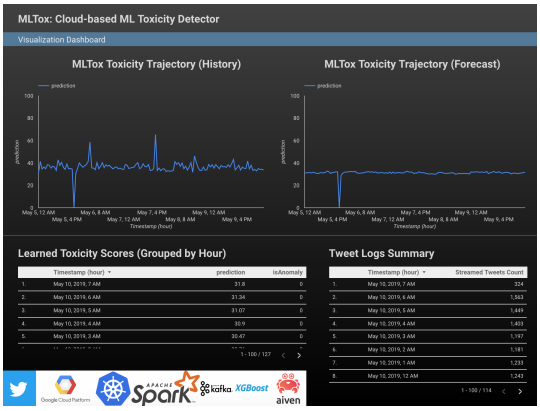

Figure 2: MLTox Public Visualization Dashboard.

4.1 High-Level Comparison with Existing Tools

Before evaluating particular metrics indicative of MLTox's performance in practice, we compare its features to open-source and closed-source tools that do similar tasks. Since these tools are proprietary, we do not have access to them to perform a comparative analysis with MLTox. Moreover, MLTox appears to be the first cloud-first system architected specifically for toxicity detection. Although a welcome surprise with respect to the uniqueness of our proposal, it does provide challenges in regards to feasibly comparing MLTox to a system similar enough to warrant comparison in the first place. The most similar tools to our project are Twitter Analytics, Brandwatch, and Mention. Twitter analytics is a service that comes with every Twitter account. Brandwatch and Mention are social media monitoring companies that provide a suite of tools to keep track of specific metrics associated with their account, including but not limited

to number of tweets, retweets, and audience impressions. We will be focusing on their twitter tools. These tools all focus on measuring the influence of an account, rather than the quality of discussion that the account or topic produces. Brandwatch Twitter, rather than provide data about the influence of the topic or account, provides data around the quality of discussion surrounding that account [12]. However, with the ability to track topics and hashtags rather than just accounts, MLTox permits a greater number of tweets and discussion threads to be analyzed. In turn, this large number of tweets provides MLTox the ability to forecast the quality of future discussion, something that other tools cannot.

4.2 Message Volumes

Although limited by our use of Twitter's free Developer API, the system still can be evaluated in the context of how many messages pass through it per day. Specifically, 368450 tweets have been streamed through the system in the less than 4 full days since we deployed it online. This is characteristic of over 100000 tweets streamed through the system (and the predictor engine) per day. When aggregated by the hour, the system streams on average 1699.54 messages through the predictor microservice per hour.

4.3 Machine Learning: XGBoost

For comprehensiveness, we evaluate the effectiveness of our machine learning models as well. The two primary components of the ML predictor service are the XGBoost model that learns the toxicity scores and the state space model (SSM) that uses these scores as a sequence history before forecasting ahead. The XGBoost model was trained on a training set of tweets separate from those it evaluates in real-time that are streamed through the system. On a held-out test set of tweets, taking the ground truth toxicity to be the output of the Perspective API, the MSE loss for the MLTox XGBoost model is 285.49. The MSE loss measures the average squared difference between the estimated values and the observed values. Taking the square root of the MSE, we have the $RMSE = 16.9$. The standard deviation of the learned toxicity scores is 19.65. The learned values are within at least one standard deviation of the mean of the ground truth values, which is reassuring, although reflective of the fact that we should pursue more robust cross-validation and hyperparameter tuning to better optimize our model. To make this comparison more concrete, we generate a cross-correlation scatterplot of the XGBoost toxicity predictions and the observed Perspective scores (Figure 3). The moderately strong, positive correlation (r=0.53) illustrates the efficacy of the learned scores in matching their associated ground truths. The plot does, however, exhibit fairly high variance. This could be indicative of either underfitting or overfitting in the model, something the authors will explore more carefully in future iterations as mentioned.

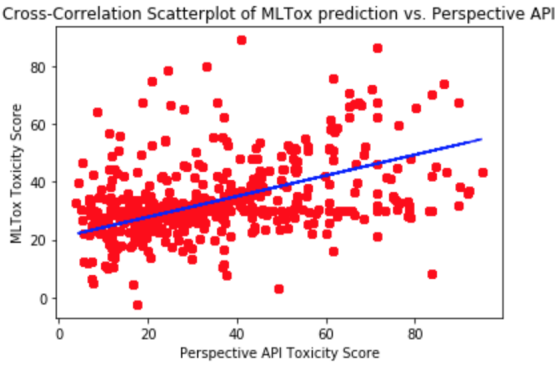

Figure 3: Cross-Correlation between MLTox and Perspective API

4.4 Machine Learning: Forecasting

We evaluate our toxicity forecasting with the help of Figures 4, 5, and 6 below. In each plot, the blue trajectory represents the observed history, while the orange represents our forecasted trajectory.

We generate forecasts using our SSM forecaster for three example trajectories. The first two are simulated trajectories to assess face-test validity of our forecaster. Namely, figure 4 takes a linear trendline with constant slope and forecasts ahead 20 steps. Clearly, the forecaster captured the dominant trend correctly. Figure 5 shows a sample sinusoidal trajectory. Again, the forecaster captured the dominant trend and forecasts ahead in accordance with the observed history. Figure 6 is a sample of a generated MLTox toxicity sequence. The trajectory is plagued with noise but the

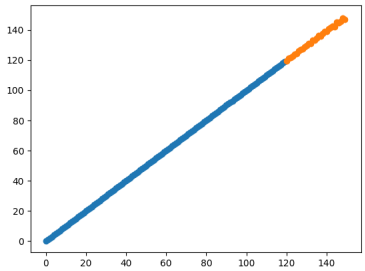

Figure 4: Simulated Linear Forecast.

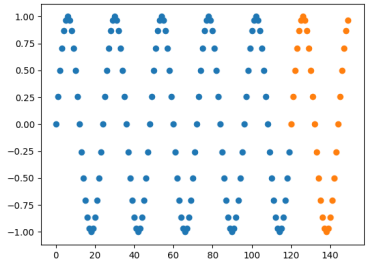

Figure 5: Simulated Sinusoidal Forecast.

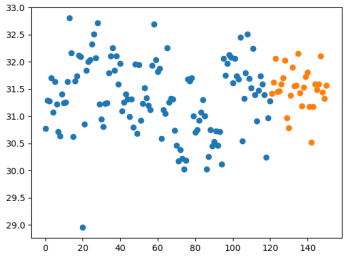

Figure 6: Observed MLTox Sequence Forecast.

forecast does appear to be in line with our expectations, bounded between 30 and 32 like most of the example points and exhibiting no clear trend behavior. Forecasting 30 hours forward based off of 3 days of data reports a mean absolute percentage error (MAPE) of 7.5.

5 DISCUSSION AND FUTURE WORK

MLTox's was developed to measure the toxicity of conversations online around a particular topic, and provide a transparent platform through which these analyses can be run. In this study, we explored the efficacy of our system in analyzing the toxicity associated with politics-related tweets. We analyzed properties of our system in practice, including the fact that it currently streams upwards of 100,000 tweets per day through our predictor service. The user-facing public dashboard, hosted at `https://datastudio.google.com/u/0/reporting/19e6NORcUonzqnsiaQ7VTVPFzWML9eW_3/page/VgD`, successfully fetches the output of our predictor and displays to users trends in toxicity in real-time. MLTox was effective in blending cloud technologies at every layer of the application stack (Kafka, Spark ML, Kubernetes, BigQuery); our decision to leverage these technologies was not made lightly or for the sake of ẗrying the next cool thing.̈ Rather, it was made out of a necessity to support scalability from the ground-up in the presence of a seemingly exponential increase in the volumes of textual data generated online.

Despite the inclusion of novel machine learning techniques packaged in the suite of tools MLTox provides to assess toxicity and the scalability our system affords (inherent in its cloud-first architectural design), its features are potentially limited. Specifically, we do not currently allow for comparison between multiple topics. Although our system could trivially be switched to listen to messages other than politics-focused tweets, the user does not currently have a way to interact with our predictor service actively. There is passive involvement between the user and MLTox's predictor engine, in that the user visits the IP of our dashboard and can view how toxicity develops over time. However, future work could permit users to interactively query the MLTox predictor, exploring the toxicity around topics in an ad-hoc fashion.

For future work MLTox could also be extended to other social media sites besides Twitter, such as Facebook and Reddit. Doing so could grant us with an even greater volume of messages to be streamed through the system, making MLTox's predictions of the toxicity around conversations more comparable with the contexts in which those conversations are being had. However, doing so likely requires us to retrain our model, since on average, tweets are shorter than Facebook or Reddit posts. Along with streaming from different social media sites, the streaming of multiple topics simultaneously can be implemented. This would allow for comparison between topics such as democrats and republicans, or different sports teams, providing greater insight into how discussion changes. More work could also be done by expanding MLTox's language classification to include semantic analysis. With semantic analysis, public perception of topics/accounts could be measured providing tools for companies or institutions. We also seek to incorporate more robust hyperparameter tuning by incorporating a variant of grid search over the model parameter values during training with cross-validation.

References

[1] Jay Kreps, Neha Narkhede, and Jun Rao. Kafka: a distributed messaging system for log processing. NetDB '11, pages 1–7. ACM, Jun 12, 2011.

[2] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. page 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[3] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. New York, NY, USA. Department of Computer Science, Columbia University.

[4] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. pages 2–2.

[5] Aditya Timmaraju and Vikesh Khanna. Sentiment analysis on movie reviews using recursive and recurrent neural network architectures. pages 2–2.

[6] Lauren Oyler. When did everything get so 'toxic'? New York Times, Oct. 2, 2018.

[7] William J. Brady, Julian A. Wills, John T. Jost, Joshua A. Tucker, and Jay J. Van Bavel. Emotion shapes the diffusion of moralized content in social networks. PNAS, July 11, 2017.

[8] Andrew Hutchinson. Twitter shares the most tweeted topics and celebrities of 2018. SocialMediaToday, Dec. 6, 2018.

[9] Tianqi Chen and Carlos Guestrin. Xgboost. KDD '16, pages 785–794. ACM, Aug 13, 2016.

[10] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. pages 7–12.

[11] statsmodels.tsa.statespace.sarimax.sarimax.

[12] Iris Vermeren. The best twitter analytics tools. Brandwatch, Feb. 15, 2019.