Model Evaluation Pipeline

Extensive Automatic Reporting for Project Detox

Author	Ajinkya Sheth (ajinkya@uw.edu)
Reviewer	
Version	1.0
Date	June 6, 2020

In collaboration with

Abstract:

Toxicity is a huge problem in online gaming industry. Toxicity in this context refers to any content posted on social media which may indicate swearing, profanity, bullying, sexual abuse, and racism. In order to combat toxic content and promote pro-social behavior, Data Scientists at Microsoft Xbox Gaming Safety team have state-of-art machine learning models as a part of Project Detox.

Presently, two models have been deployed Detox API v1 and Detox API v1.1. The Detox API 1.0 is currently in production and scores Xbox live messaging data. On the other hand, Detox API 1.1 is currently in development phase and is an iteration over Detox API 1.0, with improvements in misspellings & internet slangs.

However, the models have been tested on limited data and existing process for model evaluation and performance measure is a manual activity. To streamline these activities, Microsoft Gaming Safety Team is collaborating with iSchool at University of Washington as a part of the 2020 iSchool Capstone Event.

Team Fan-STATS-tic 4 of the iSchool is delivering the prototype project – Model Evaluation Pipeline to Microsoft Gaming Safety Team as a part of this Capstone Experience.

Team Roles and Responsibilities:

Microsoft Gaming Safety Team

John Sherwin – Project Manager
Kate Ansolis – Project Lead & Lead Software Development Engineer
Monica Tongya – Lead Data Scientist

iSchool Team Fan-STATS-tic 4

Ajinkya Sheth – Solutions Architect & Data Engineer
Prakirn Kumar – Data Scientist & Developer
Ishita Bhandari – Data Scientist & Developer

iSchool Mentor

Dr. Phil Fawcett – Capstone Manager

Table of Contents

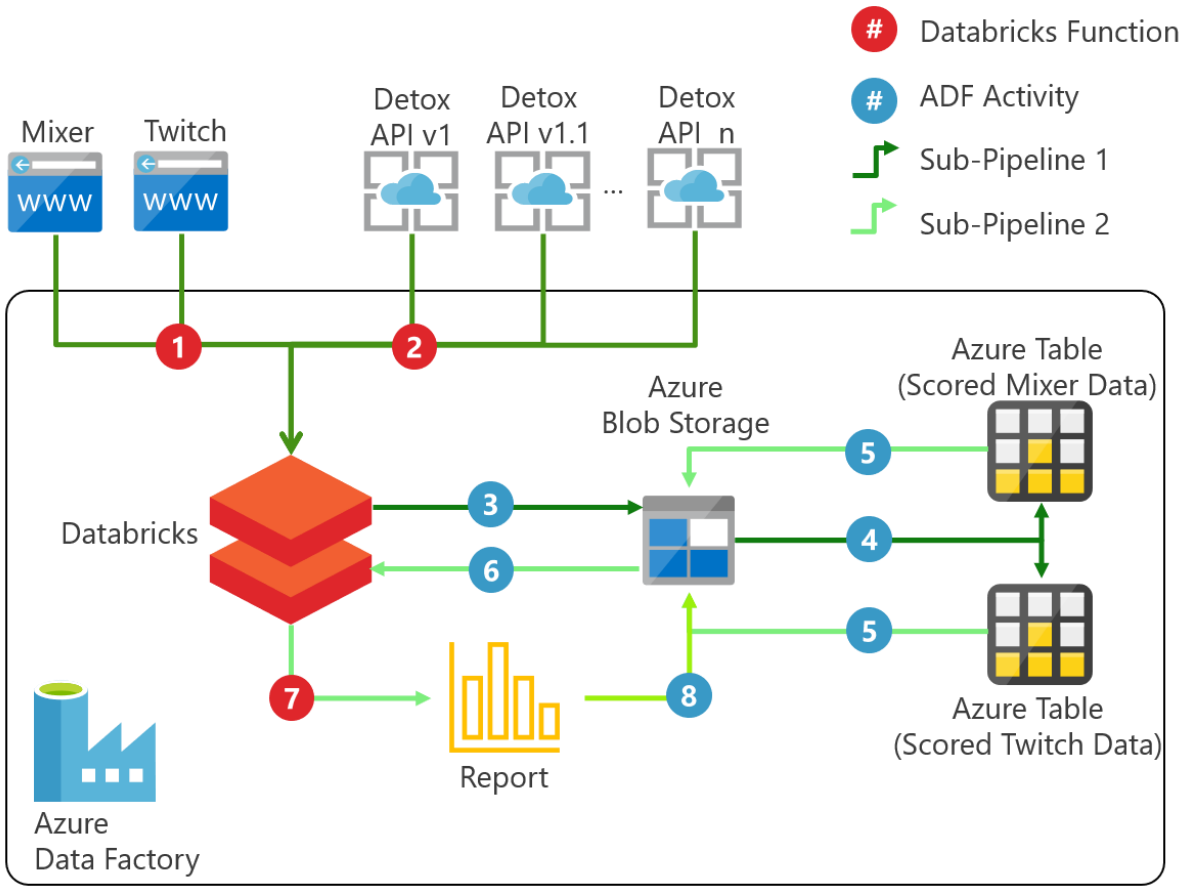
3. The Pipeline

3.1 Block Diagram

The pipeline has been designed in Azure Data Factory. Databricks (DB) Cluster is the compute in the pipeline, and we are using Azure Blob storage and Table storage for storing the data. ADF Activities available in Azure Data Factory to perform data movement between compute and storage.

Note: *The tables in pipeline have been named as *_V1 or *_V2 these naming conventions correspond to Detox API v1 and v1.1. The reason these tables are named thus is because we have been referring to API v1 and v1.1 as V1 and V2 respectively at the very start. Apologies for this misrepresentation.*

Data Flow Diagram



#	Sub-pipeline 1: Get Incremental Data	#	Sub-pipeline 2: Reporting using entire data
1	Web Scraping	5	Copy Data – Azure Table to Blob
2	Batch Scoring	6	Copy Data – Blob to Databricks
3	Copy Data – From Databricks to Blob	7	Generate Reports
4	Copy Data – From Blob to Azure Table	8	Store Reports in Blob

Sub-pipeline 1 captures live or newly generated chat data from game streaming websites - Twitch and Mixer through web scraping (discussed in section ____) The scraped data is scored with Detox API v1 and v 1.1 via Batch Scoring Process (discussed in section ____). Azure Blob storage is mounted on the Databricks cluster to provide temporary storage for newly captured and scored data. The data is incrementally maintained in Azure Table.

In Sub-pipeline 2, data in Azure table is first copied to the blob storage which is mounted to Databricks cluster. The DB cluster reads this data and generates interactive JS/HTML reports which are stored in the Blob for easy access.

3.2 Specifications

3.2.1 Azure Specifications

Following table covers various Azure components used to implement the pipeline.

Azure Component	**Component Name**	**Comment**
Resource Group	capstone_rg	
Storage Group	Capstonestorage3	
Data Factory	MLTestingPipeline	
Pipeline	DetoxPipeline_ScrapeAndStore	Sub-pipeline 1
	DetoxPipeline_Reporting	Sub-pipeline 2
Dataset	PipelineDatastore_*	Described in details in the following table
Databricks Cluster	Detox Cluster	Compute of the pipelines, Runs Python notebooks

3.2.2 Datastore Specifications

Azure Tables: All Azure tables are designed to hold the data incrementally.

Table Name	**Description**	**Utility**
PipelineDatastore_MixerInn	Input Mixer Data	
PipelineDatastore_MixerOut_V1	Scored Mixer Data, Detox 1	
PipelineDatastore_MixerOut_V2	Scored Mixer Data, Detox 1.1	
PipelineDatastore_MixerOut_V1V2	Scored Mixer Data, Detox 1 & 1.1	Input Table for sub-pipeline 2
PipelineDatastore_TwitchInn	Input Mixer Data	
PipelineDatastore_TwitchOut_V1	Scored Twitch Data, Detox 1	
PipelineDatastore_TwitchOut_V2	Scored Twitch Data, Detox 1.1	
PipelineDatastore_TwitchOut_V1V2	Scored Twitch Data, Detox 1 & 1.1	Input Table for sub-pipeline 2

3.2.3 Blob Specifications

Blob Folders: Blob storage serves multiple purposes. It is used to hold data temporarily before processing in Databricks and after processing in Databricks. It also stores the pipeline reports. Additionally, the temporary files in blob storage are archived by default and manual deletion may be required to clear up the space. The blob folder structure is discussed below.

Blob Container: capstonestorage3/pipelinecontianer
ADF Dataset: PipelineDatastore_blob

Directory	Files	Description
csv_archive	YYYYMMDD_HHmm_{%%}.csv %% = {Mixer|Twitch+*+V1|V2|V1V2}	Archived data for each sub-pipeline 1 run
logdir	uid/*.csv	ADF Delete Activity Logs
pipelinedir		Temp storage to hold Sub-Pipeline 1 output before copying to Azure Tables (described above) and deletion
report_inp	mixerdata.csv twitchdata.csv	Copy of PipelineDatastore_MixerOut_V1V2 and PipelineDatastore_TwitchOut_V1V2, Input data for Sub-Pipeline 2
reportdir	YYYYMMDD_HHmm_{%%}_Report.html	
	{%%}_Detox_APIv1	Performance of Detox API 1 on Twitch and Mixer scraped data
	{%%}_MixerVsTwitch_Detox	Comparison between Detox API 1 and Detox API 1.1 on Twitch and Mixer Data
	{%%}_Detox_APIv1.1	Performance of Detox API 1.1 on Twitch and Mixer

3.2.4 Databricks Specifications

The Databricks cluster consists of notebooks running Python code to perform Data Engineering activities. Databricks cluster URL: https://adb-8195675336672924.4.azuredatabricks.net/

Python Libraries Installed from PyPI: dominate, wordcloud, plotly, chartstudio, emoji, requests_oauthlib

Databricks Notebooks	Pipeline	Functions
LoadAndScoreNB	ScrapeAndStore	Webscraping, Batch scoring
ReportDetoxApiComparisons	Reporting	Generates two reports: For API v1 and v1.1
ReportMixerVsTwitch	Reporting	Generates API v1 vs v1.1 report

3.3 Sub-pipeline 1

3.3.1 Databricks Activity

Notebook Name in Databricks Cluster: LoadAndScoreNB
Databricks Activity: DownloadAndScore

3.3.1.1 Parameters and Pre-Requisites

Set the following parameters before starting:

Parameters	Value
Mount Blob Storage	fs.azure.account.key.capstonestorage3.blob.core.windows.net
blob_path	wasbs://pipelinecontainer@capstonestorage3.blob.core.windows.net/pipelinedir
archive_path	wasbs://pipelinecontainer@capstonestorage3.blob.core.windows.net/csv_archive
detox_api_v1_url	https://detox-ppe.xboxlive.com/api/v1/service/aks-scoring-service/score
detox_api_v1_token	xxxx
detox_api_v2_url	http://168.62.219.72:80/api/v1/service/detox-demo-test/score
detox_api_v2_token	xxxx
twitch_username	
twitch_token	
twitch_channel	@PipelineParameter
twitch_server	irc.chat.twitch.tv
twitch_port	6667
mixer_client_id	
mixer_client_token	
mixer_channel	@PipelineParameter
twitch_scrape_time	@PipelineParameter (seconds)
mixer_scrape_time	@PipelineParameter (seconds)
mixer_hits	@PipelineParameter (#)

3.3.1.2 Web Scraping

Web Scraping in this context is about collecting newly generated live data. Since this model APIs have been tested on limited data, it is imperative for us to collect data from popular live streaming websites like Twitch and Mixer. Twitch has a huge user base and viewership and has been around for years.

Meanwhile, Mixer is Microsoft's in-house game streaming website launched recently. Moreover, Twitch has been considered more Toxic as compared to Mixer. By collecting publicly available data, we also attempt to measure model performance on competitors.

Web-scraping process is different for Mixer and Twitch as outlined below. However, both the processes grant the user enough control on the amount of data to be scraped and from which channel.

Twitch	**Mixer**
Scraping Protocol: IRC (get chat stream)	Scraping Protocol: Rest API (get chat history)
Choose a channel Set capture duration Log chat streams Store the data	Choose a channel Set capture Duration (d) Set number of Hits (n) Hit Mixer API n times in d duration Store the data

3.3.1.3 Batch Scoring

Gamer chat contains a large number of small messages. Detox API are deployed on cloud and hence for the scoring to happen the API needs to be hit via the internet. Hitting the API multiple times through the network, would result into a bottleneck.

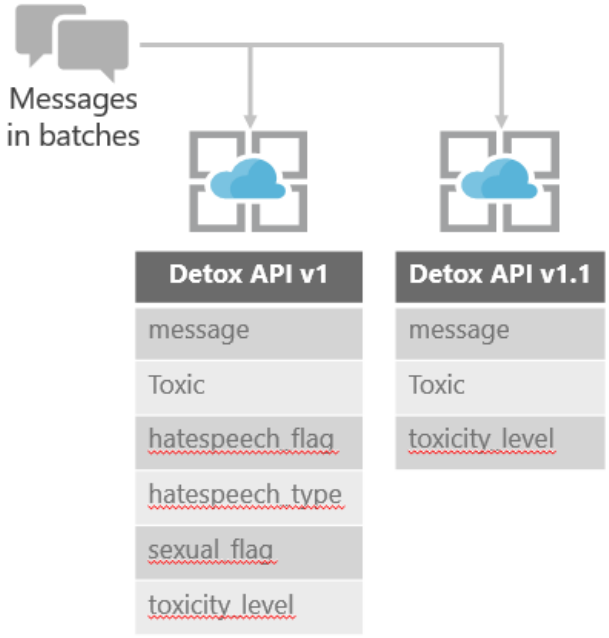
Hence, an optimal way is to score the chat text in batches. We figured a decent batch size is 2000. The Batch Scoring process sends messages in batches of 2000 and gets the scored data through REST API.

The scored data is then stored in Azure table incrementally for further processing.

Batch Scoring Function
Input:
series: Series of Messages to be scored
batch_size: Number of messages to score in a single API call
auth_token: Detox API token
url: Detox API endpoint
Output:
dataframe with toxicity scores and other API contracts

```
def batch_data(series_text,batch_size=2000, auth_token=detox_api_v1_token,
url=detox_api_v1_url):
    all_res=[]
    total_row_count=series_text.shape[0]
    if total_row_count>batch_size:
        batch_size=total_row_count
    for i in range(0, total_row_count, batch_size):
        j = i + batch_size
        ls = series_text[i:j].tolist()
        hed = {'Authorization': 'Bearer ' + auth_token}
        #data = ["This is my test sentence","Gonna win this match, you losers"]
        data = list(ls)
        response = requests.post(url, json = data, headers = hed)
        print(response)
        d = response.json()
        output = pd.DataFrame(d)
        #output['messag'] = ls
        all_res.append(output)
        final = pd.concat(all_res, ignore_index = True)
    return final
```

The APIs have different contracts as described in the figure below:



3.3.1 Blob to Azure Tables

Databricks cluster saves the processed files in csv format on the following locations of the blob storage.

Location	Purpose
pipelinedir	For temp location before moving to Azure Tables
csv_archive	For archival

There are 8 ADF copy activities which load the 8 output csv files in pipelinedir to 8 tables as described in 3.2.2 incrementally.

All Azure tables have the following columns common between them:

Column Name	Comment
PartitionKey	YYYYMMDD
RowKey	UID
RunID	UID
Timestamp	YYYY-MM-DDTHH:mm:SS.sssZ
channel	
username	

There are other columns as well based on the website the data has been scraped from and the API contracts

3.3.2 Delete Files in Temporary Location

After copying the files, the ADF Delete Activity discards the files in the temporary storage.

3.4 Sub-pipeline 2

3.4.1 Databricks Activity - ReportGen_MixerVsTwitch

Notebook Name in Databricks Cluster: ReportMixerVsTwitch

3.4.2 Databricks Activity - ReportGen_DetoxAPIComparisons

Notebook Name in Databricks Cluster: ReportDetoxAPIComparisons