Feature Analysis Using TensorFlow Data Validation and Facets

Learning Objectives

1. Use TFRecords to load record-oriented binary format data
2. Use TFDV to generate statistics and Facets to visualize the data
3. Use the TFDV widget to answer questions
4. Analyze label distribution for subset groups

Introduction

Bias can manifest in any part of a typical machine learning pipeline, from an unrepresentative dataset, to learned model representations, to the way in which the results are presented to the user. Errors that result from this bias can disproportionately impact some users more than others.

TensorFlow Data Validation (TFDV) is one tool you can use to analyze your data to find potential problems in your data, such as missing values and data imbalances - that can lead to Fairness disparities. The TFDV tool analyzes training and serving data to compute descriptive statistics, infer a schema, and detect data anomalies. Facets Overview provides a succinct visualization of these statistics for easy browsing. Both the TFDV and Facets are tools that are part of the Fairness Indicators.

In this notebook, we use TFDV to compute descriptive statistics that provide a quick overview of the data in terms of the features that are present and the shapes of their value distributions. We use Facets Overview to visualize these statistics using the Civil Comments dataset.

Each learning objective will correspond to a **#TODO** in the student lab notebook -- try to complete that notebook first before reviewing this solution notebook.

Set up environment variables and load necessary libraries

We will start by importing the necessary dependencies for the libraries we'll be using in this exercise. First, run the cell below to install Fairness Indicators.

NOTE: You can ignore the "pip" being invoked by an old script wrapper, as it will not affect the lab's functionality.

In [1]:
```
# Fairness Indicators is designed to support in evaluating, improving, and comparing mo
!pip3 install fairness-indicators==0.1.2 --user
```

```
Collecting fairness-indicators==0.1.2
Downloading fairness_indicators-0.1.2-py3-none-any.whl (48 kB)
```

```
|████████████████████████████████| 48 kB 2.3 MB/s eta 0:00:011
Requirement already satisfied: witwidget<2,>=1.4.4 in /opt/conda/lib/python3.7/site-pack
ages (from fairness-indicators==0.1.2) (1.7.0)
Requirement already satisfied: setuptools>=40.2.0 in /opt/conda/lib/python3.7/site-packa
ges (from fairness-indicators==0.1.2) (49.6.0.post20200814)
Requirement already satisfied: tensorflow-data-validation<1,>=0.15.0 in /home/jupyter/.l
ocal/lib/python3.7/site-packages (from fairness-indicators==0.1.2) (0.24.0)
.
.
.
Installing collected packages: fairness-indicators
Attempting uninstall: fairness-indicators
Found existing installation: fairness-indicators 0.24.0
Uninstalling fairness-indicators-0.24.0:
Successfully uninstalled fairness-indicators-0.24.0
Successfully installed fairness-indicators-0.1.2
```

Restart the kernel after you do a pip3 install (click on the **Restart the kernel** button above).

Kindly ignore the deprecation warnings and incompatibility errors.

Next, import all the dependencies we'll use in this exercise, which include Fairness Indicators, TensorFlow Data Validation (tfdv), and the What-If tool (WIT) Facets Overview.

In [2]:

```
# You can use any Python source file as a module by executing an import statement in so
# The import statement combines two operations; it searches for the named module, then
# to a name in the local scope.
# %tensorflow_version 2.x
import sys, os
import warnings
warnings.filterwarnings('ignore')
#os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'  # Ignore deprecation warnings
import tempfile
import apache_beam as beam
import numpy as np
import pandas as pd
from datetime import datetime

import tensorflow_hub as hub
import tensorflow as tf
import tensorflow_model_analysis as tfma
import tensorflow_data_validation as tfdv
from tensorflow_model_analysis.addons.fairness.post_export_metrics import fairness_indi
from tensorflow_model_analysis.addons.fairness.view import widget_view
from fairness_indicators.examples import util

import warnings
warnings.filterwarnings("ignore")

from witwidget.notebook.visualization import WitConfigBuilder
from witwidget.notebook.visualization import WitWidget


print(tf.version.VERSION)
print(tf) # This statement shows us what version of Python we are currently running.
```

```
2.4.0-dev20200922
<module 'tensorflow' from '/home/jupyter/.local/lib/python3.5/site-packages/tensorflow/_
_init__.py'>
```

About the Civil Comments dataset

Click below to learn more about the Civil Comments dataset, and how we've preprocessed it for this exercise.

The Civil Comments dataset comprises approximately 2 million public comments that were submitted to the Civil Comments platform. Jigsaw sponsored the effort to compile and annotate these comments for ongoing research; they've also hosted competitions on Kaggle to help classify toxic comments as well as minimize unintended model bias.

Features

Within the Civil Comments data, a subset of comments are tagged with a variety of identity attributes pertaining to gender, sexual orientation, religion, race, and ethnicity. Each identity annotation column contains a value that represents the percentage of annotators who categorized a comment as containing references to that identity. Multiple identities may be present in a comment.

NOTE: These identity attributes are intended *for evaluation purposes only*, to assess how well a classifier trained solely on the comment text performs on different tag sets.

To collect these identity labels, each comment was reviewed by up to 10 annotators, who were asked to indicate all identities that were mentioned in the comment. For example, annotators were posed the question: "What genders are mentioned in the comment?", and asked to choose all of the following categories that were applicable.

- Male
- Female
- Transgender
- Other gender
- No gender mentioned

NOTE: *We recognize the limitations of the categories used in the original dataset, and acknowledge that these terms do not encompass the full range of vocabulary used in describing gender.*

Jigsaw used these ratings to generate an aggregate score for each identity attribute representing the percentage of raters who said the identity was mentioned in the comment. For example, if 10 annotators reviewed a comment, and 6 said that the comment mentioned the identity "female" and 0 said that the comment mentioned the identity "male," the comment would receive a `female` score of `0.6` and a `male` score of `0.0`.

NOTE: For the purposes of annotation, a comment was considered to "mention" gender if it contained a comment about gender issues (e.g., a discussion about feminism, wage gap between men and women, transgender rights, etc.), gendered language, or gendered insults. Use of "he," "she," or gendered names (e.g., Donald, Margaret) did not require a gender label.

Label

Each comment was rated by up to 10 annotators for toxicity, who each classified it with one of the following ratings.

- Very Toxic

- Toxic
- Hard to Say
- Not Toxic

Again, Jigsaw used these ratings to generate an aggregate toxicity "score" for each comment (ranging from `0.0` to `1.0`) to serve as the label, representing the fraction of annotators who labeled the comment either "Very Toxic" or "Toxic." For example, if 10 annotators rated a comment, and 3 of them labeled it "Very Toxic" and 5 of them labeled it "Toxic", the comment would receive a toxicity score of `0.8`.

NOTE: For more information on the Civil Comments labeling schema, see the Data section of the Jigsaw Untended Bias in Toxicity Classification Kaggle competition.

Preprocessing the data

For the purposes of this exercise, we converted toxicity and identity columns to booleans in order to work with our neural net and metrics calculations. In the preprocessed dataset, we considered any value ≥ 0.5 as True (i.e., a comment is considered toxic if 50% or more crowd raters labeled it as toxic).

For identity labels, the threshold 0.5 was chosen and the identities were grouped together by their categories. For example, if one comment has `{ male: 0.3, female: 1.0, transgender: 0.0, heterosexual: 0.8, homosexual_gay_or_lesbian: 1.0 }`, after processing, the data will be `{ gender: [female], sexual_orientation: [heterosexual, homosexual_gay_or_lesbian] }`.

NOTE: Missing identity fields were converted to False.

Use TFRecords to load record-oriented binary format data

The TFRecord format is a simple Protobuf-based format for storing a sequence of binary records. It gives you and your machine learning models to handle arbitrarily large datasets over the network because it:

1. Splits up large files into 100-200MB chunks
2. Stores the results as serialized binary messages for faster ingestion

If you already have a dataset in TFRecord format, you can use the tf.keras.utils functions for accessing the data (as you will below!). If you want to practice creating your own TFRecord datasets you can do so outside of this lab by viewing the documentation here.

TODO 1: Use the utility functions tf.keras to download and import our datasets

Run the following cell to download and import the training and validation preprocessed datasets.

In [3]:
```
download_original_data = False #@param {type:"boolean"}
```

```
# TODO 1

# Downloads a file from a URL if it is not already in the cache using the `tf.keras.uti
if download_original_data:
  train_tf_file = tf.keras.utils.get_file('train_tf.tfrecord',
                                          'https://storage.googleapis.com/civil_comment
  validate_tf_file = tf.keras.utils.get_file('validate_tf.tfrecord',
                                             'https://storage.googleapis.com/civil_comm

  # The identity terms list will be grouped together by their categories
  # (see 'IDENTITY_COLUMNS') on threshould 0.5. Only the identity term column,
  # text column and label column will be kept after processing.
  train_tf_file = util.convert_comments_data(train_tf_file)
  validate_tf_file = util.convert_comments_data(validate_tf_file)

# TODO 1a

else:
  train_tf_file = tf.keras.utils.get_file('train_tf_processed.tfrecord',
                                          'https://storage.googleapis.com/civil_comment
  validate_tf_file = tf.keras.utils.get_file('validate_tf_processed.tfrecord',
                                             'https://storage.googleapis.com/civil_comm
```

```
Downloading data from https://storage.googleapis.com/civil_comments_dataset/train_tf_pro
cessed.tfrecord
488161280/488153424 [==============================] - 8s 0us/step
Downloading data from https://storage.googleapis.com/civil_comments_dataset/validate_tf_
processed.tfrecord
324943872/324941336 [==============================] - 3s 0us/step
```

Use TFDV to generate statistics and Facets to visualize the data

TensorFlow Data Validation supports data stored in a TFRecord file, a CSV input format, with extensibility for other common formats. You can find the available data decoders here. In addition, TFDV provides the tfdv.generate_statistics_from_dataframe utility function for users with in-memory data represented as a pandas DataFrame.

In addition to computing a default set of data statistics, TFDV can also compute statistics for semantic domains (e.g., images, text). To enable computation of semantic domain statistics, pass a tfdv.StatsOptions object with enable_semantic_domain_stats set to True to tfdv.generate_statistics_from_tfrecord.Before we train the model, let's do a quick audit of our training data using TensorFlow Data Validation, so we can better understand our data distribution.

TODO 2: Use TFDV to get quick statistics on your dataset

The following cell may take 2–3 minutes to run. Please ignore the deprecation warnings.

NOTE: Please re-run the below cell if you are not getting the TensorFlow Data Validation widget in the output.

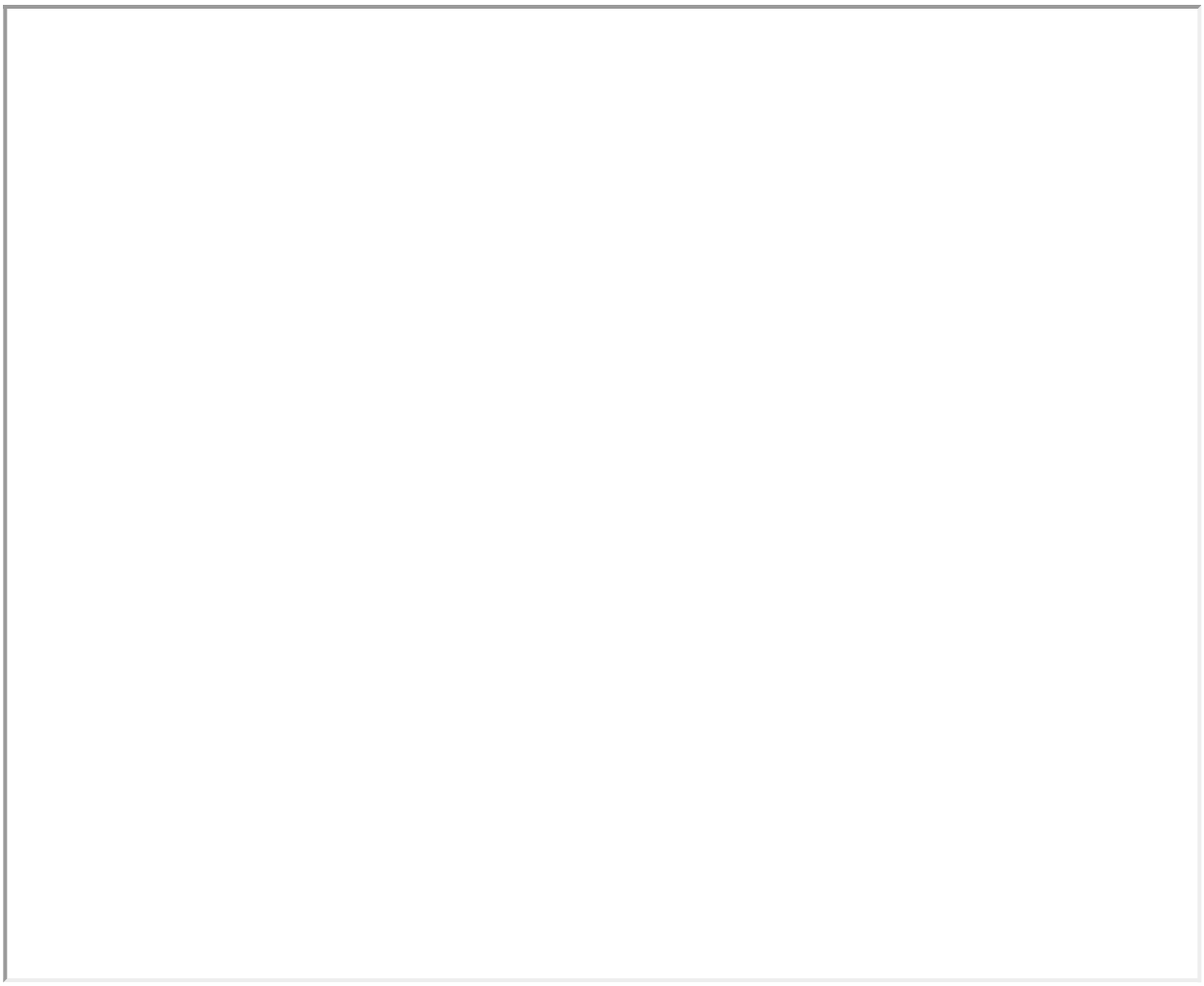
In [4]:

```
# TODO 2

# The computation of statistics using TFDV.  The returned value is a DatasetFeatureStat
stats = tfdv.generate_statistics_from_tfrecord(data_location=train_tf_file)

# TODO 2a
```

```
# A visualization of the statistics using Facets Overview.
tfdv.visualize_statistics(stats)
```



TODO 3: Use the TensorFlow Data Validation widget above to answer the following questions.

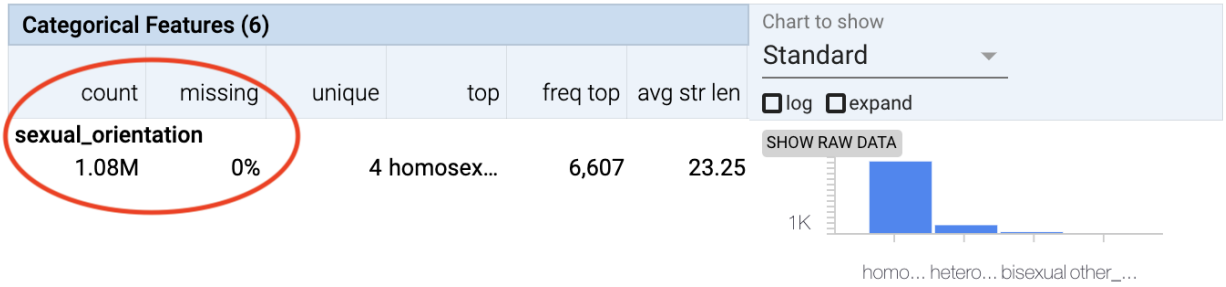
1. How many total examples are in the training dataset?

Solution

See below solution.

There are 1.08 million total examples in the training dataset.

The count column tells us how many examples there are for a given feature. Each feature (`sexual_orientation` , `comment_text` , `gender` , etc.) has 1.08 million examples. The missing column tells us what percentage of examples are missing that feature.



Each feature is missing from 0% of examples, so we know that the per-feature example count of 1.08 million is also the total number of examples in the dataset.

2. How many unique values are there for the `gender` feature? What are they, and what are the frequencies of each of these values?

NOTE #1: `gender` and the other identity features (`sexual_orientation`, `religion`, `disability`, and `race`) are included in this dataset for evaluation purposes only, so we can assess model performance on different identity slices. The only feature we will use for model training is `comment_text`.

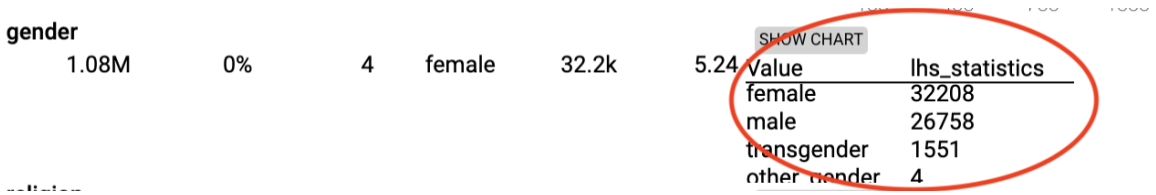
NOTE #2: *We recognize the limitations of the categories used in the original dataset, and acknowledge that these terms do not encompass the full range of vocabulary used in describing gender.*

Solution

See below solution.

The **unique** column of the **Categorical Features** table tells us that there are 4 unique values for the `gender` feature.

To view the 4 values and their frequencies, we can click on the **SHOW RAW DATA** button:



The raw data table shows that there are 32,208 examples with a gender value of `female`, 26,758 examples with a value of `male`, 1,551 examples with a value of `transgender`, and 4 examples with a value of `other gender`.

NOTE: As described earlier, a `gender` feature can contain zero or more of these 4 values, depending on the content of the comment. For example, a comment containing the text "I am a transgender man" will have both `transgender` and `male` as `gender` values, whereas a comment that does not reference gender at all will have an empty/false `gender` value.

3. What percentage of total examples are labeled toxic? Overall, is this a class-balanced dataset (relatively even split of examples between positive and negative classes) or a class-imbalanced dataset (majority of examples are in one class)?

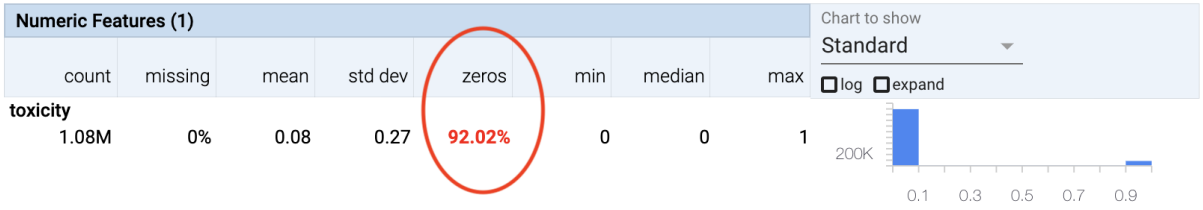
NOTE: In this dataset, a `toxicity` value of `0` signifies "not toxic," and a `toxicity` value of `1` signifies "toxic."

Solution

See below solution.

7.98 percent of examples are toxic.

Under **Numeric Features**, we can see the distribution of values for the `toxicity` feature. 92.02% of examples have a value of 0 (which signifies "non-toxic"), so 7.98% of examples are toxic.



This is a **class-imbalanced dataset**, as the overwhelming majority of examples (over 90%) are classified as nontoxic.

Notice that there is one numeric feature (count of toxic comments) and six categorical features.

TODO 4: Analyze label distribution for subset groups

Run the following code to analyze label distribution for the subset of examples that contain a `gender` value

NOTE: *The cell should run for just a few minutes*

In [5]:

```
#@title Calculate label distribution for gender-related examples
raw_dataset = tf.data.TFRecordDataset(train_tf_file)

toxic_gender_examples = 0
nontoxic_gender_examples = 0

# TODO 4

# There are 1,082,924 examples in the dataset
# The `take()` method returns the specified number of elements starting from the first
for raw_record in raw_dataset.take(1082924):
  example = tf.train.Example()
  example.ParseFromString(raw_record.numpy())
  if str(example.features.feature["gender"].bytes_list.value) != "[]":
    if str(example.features.feature["toxicity"].float_list.value) == "[1.0]":
      toxic_gender_examples += 1
    else:
      nontoxic_gender_examples += 1

# TODO 4a

print("Toxic Gender Examples: %s" % toxic_gender_examples)
print("Nontoxic Gender Examples: %s" % nontoxic_gender_examples)
```

```
Toxic Gender Examples: 7189
Nontoxic Gender Examples: 41572
```

What percentage of `gender` examples are labeled toxic? Compare this percentage to the percentage of total examples that are labeled toxic from #3 above. What, if any, fairness concerns can you identify based on this comparison?

There are 7,189 gender-related examples that are labeled toxic, which represent 14.7% of all gender-related examples.

The percentage of gender-related examples that are toxic (14.7%) is nearly double the percentage of toxic examples overall (7.98%). In other words, in our dataset, gender-related comments are almost two times more likely than comments overall to be labeled as toxic.

This skew suggests that a model trained on this dataset might learn a correlation between gender-related content and toxicity. This raises fairness considerations, as the model might be more likely to classify nontoxic comments as toxic if they contain gender terminology, which could lead to disparate impact for gender subgroups.