Распознавание ботов

Контекст

Представьте, что у вас есть сервис, который поддерживает регистрацию новых пользователей. Вам стало известно, что вместо реальных пользователей на вашем сайте создается множество бот-акканутов. Боты – это пользователи, которые созданы при помощи автоматических скриптов. Обычно аккаунты, зарегистрированные одним автоскриптом, при незначительных вариациях очень схожи. Мало того, что на ботов тратятся вычислительные ресурсы, так они еще зашумляют бизнес-метрики. Жить так больше нельзя! Поэтому решено построить ML-систему, которая для каждой регистрации ставит в соответствие вероятность: насколько этот пользователь "похож на бота".

Описанная ситуация – это реальная задача Антиспама Почты, и в данном соревновании вам необходимо ее решить. Но есть нюанс: в процессе вам нужно обучить TLS-эмбеддинг!

Что такое эмбеддинг?

Эмбеддинг – это векторное представление нечисловых слабоструктурированных данных в некотором многомерном пространстве. Важным свойством такого многомерного пространства является то, что если два исходных объекта похожи, то их эмбеддинги должны быть "близки" в этом пространстве. Наиболее известными примерами являются текстовые эмбеддинги, но эмбеддинги можно обучать для абсолютно разных данных, в том числе для TLS-шифров.

А почему TLS-эмбеддинг?

Чтобы “общаться” с использованием TLS, клиент и сервер должны договорится о том, как обмениваться ключами и какими алгоритмами шифровать трафик. Список доступных клиенту алгоритмов определяется платформой, операционной системой, браузером, поэтому боты, которых регистрирует один и тот же автоскрипт, обычно разделяют похожие наборы шифров. Это первая причина.

Вторая причина заключается в следующем: чтобы понять, с какого устройства происходит регистрация, мы используем информацию из HTTP-заголовка User-Agent. Но этот способ не является надежным, так как клиент может вставить абсолютно произвольную строку в данный HTTP-заголовок и это не возможно достоверно выявить. С другой стороны, TLS-шифры, которые предоставляет клиент для сервера, труднее подделать, так как это требует намного более сложных манипуляций на уровне ОС и SSL/TLS-библиотек. Дополнительным аргументом в копилку TLS-шифров будет исследование, в котором по TLS-фингерпринтам достаточно успешно удается определить семейство браузеров.

Поэтому и встает вопрос об обучении векторного представления – эмбеддинга, который описывает поддерживаемые клиентом TLS-шифры.

Задача

Основная задача соревнования – это бинарная классификация на два класса "Бот" (метка 1) и "Не Бот" (метка 0), используя следующие данные:

1. юзер-агент клиента;
2. названия TLS-шифров, которые поддерживает клиент;
3. названия TLS эллиптических кривых, которые поддерживаются TLS-шифрами клиента.

Данные из пункта 2 и 3 получены в ходе установки TLS-соединения из сообщения ClientHello.

Обязательное условие

Мы ожидаем, что для обучения финального классификатора командами будет построен TLS-эмбеддинг, который обучен на TLS-шифрах и TLS-кривых. Командам потребуется загрузить не только предсказания своей модели, но также и код/ноутбук, который генерирует решение для того, чтобы мы могли убедится, что в процессе обучения downstream-классификатора действительно используется предварительно обученный TLS-эмбеддинг. TLS-эмбеддинг может быть обучен

1. совместно на шифрах и кривых:

$$F(\text{Ciphers}, \text{Curves}) = \text{TLS-embedding};$$

2. отдельно на шифрах и отдельно на кривых, а финальный TLS-эмбеддинг получен их конкатенацией:

$$\text{Concat}(F_1(\text{Ciphers}), F_2(\text{Curves})) = \text{TLS-embedding}.$$

Использование данных о юзер-агенте клиента в итоговой классификации не является обязательным, но скорее всего позволит получить более высокое место в лидерборде. На то, как команды будут использовать строку юзер-агента, требований не накладывается, но мы по своему опыту рекомендуем обучить отдельный эмбеддинг (UA-эмбеддинг):

$$G(\text{User-Agent}) = \text{UA-embedding}.$$

В таком случае финальный классификатор ботов – f – будет выглядит следующим образом:

$$P_{bot} = f(\text{UA-embedding}, \text{TLS-embedding}).$$

Оценка

Результаты подводятся на private-лидерборде, который открывается по завершении соревнования. Метрика оценки классификации ботов – площадь под ROC-кривой – **ROCAUC**. У **топ-3 команд** из private-лидерборда проверяется код на предмет обучения TLS-эмбеддинга. При обнаружении отсутствия построения TLS-эмбеддинга, мы исключаем данное решение и берем следующее по топу.

Данные

Данные для обучения представлены в формате Apache Parquet – это бинарный колоночный формат данных, который позволяет хранить данные более оптимально, чем CSV. Для загрузки этих файлов можно использовать привычный Pandas, но потребуется установить либо fastparquet, либо PyArrow.

```
$ pip install pandas fastparquet
```

Далее read_csv() заменяется на read_parquet(), и можно работать привычным образом:

```
import pandas as pd

df = pd.read_parquet("train.parquet")
```

Командам доступны 3 файла с данными:

1. **train.parquet** – размеченные данные для обучения;
2. **test.parquet** – неразмеченные данные: для каждого объекта из этой выборки необходимо оценить "вероятность бота";
3. **unlabelled.snappy.parquet** – большая неразмеченная выборка: эти данные команды могут использовать опционально для обучения TLS и UA эмбеддингов.

В виду ограничения платформы на размер файла, выборку **unlabelled.snappy.parquet** можно скачать из облака по ссылке.

Пример TLS-шифров

```
[
  'ECDHE-RSA-AES128-GCM-SHA256', '0x1301', 'AES256-SHA',
  '0x1303', 'ECDHE-RSA-AES256-SHA', '0xcca8',
  'ECDHE-ECDSA-AES128-GCM-SHA256', 'AES128-SHA',
  'ECDHE-ECDSA-AES256-GCM-SHA384', 'AES256-GCM-SHA384',
  '0x9a9a', 'ECDHE-RSA-AES128-SHA', 'AES128-GCM-SHA256',
  '0x1302', '0xcca9', 'ECDHE-RSA-AES256-GCM-SHA384'
]
```

Пример TLS-кривых

```
['secp384r1', '0xdada', 'prime256v1', '0x001d']
```

Важные факты о данных

Для того чтобы сделать задачу более интересной, данные были выбраны с некоторыми нюансами. Основной нюанс касается временного промежутка, за который данные были собраны.

1. **train.parquet** в основном составляют данные за 2021 год;
2. **test.parquet** собран в большей мере за 2022 год;
3. **unlabelled.snappy.parquet** – это большой неразмеченный сэмпл данных за 2022 год.

Есть второй нюанс, который следует иметь в виду. В выборках присутствуют объекты с трех платформ Почты: Web, iOS, Android. К какой платформе относится объект можно понять по юзер-агенту.

Подсказка

Скорее всего, если вы будете использовать неразмеченный сэмпл для обучения эмбеддингов, то получите более высокое место в лидерборде.

Как обучить эмбеддинги?

Раскрывать все возможные варианты не будем, так как именно здесь команды могут проявить все свое творчество и креативность. Дадим лишь общие направления для исследования.

1. TLS-эмбеддинг можно обучать как supervised (классификация ботов), так и в unsupervised (self-supservised) режиме, а можно комбинировать оба подхода вместе.
2. Особо искушенные могут попробовать методы semi-supervised обучения.
3. Для unsupervised/self-supervised обучения рекомендуем использовать большую неразмеченную выборку.
4. Для self-supervised обучения советуем присмотреться к таким задачам, как Masked Langauge Modeling (MLM) и др.
5. Вы можете разделить между участниками команды задачу обучения TLS/UA-эмбеддингов и задачу обучения downstream-классификатора над предобученными эмбеддингами. А можете обучать все в единой модели end-to-end.

Загрузка решения

Решение неободимо загружать в виде CSV-файла submission.csv следующего формата:

```
id,is_bot
12,0.55
21,0.73
33,0.96
...
```

До закрытия соревнования командам необходимо отметить то решение, которое они считают лучшим, и именно по отмеченным решениям будет строится private-лидерборд. После того как команда отмечает итоговое решение, следующий шаг – это загрузить код который сгенерировал submission.csv. Допустимые форматы: **.zip, .ipynb, .py.** Ожидаем, что в приложенном коде будет понятно описана структура обучаемых моделей, а также самого процессе обучения. Напомним, это необходимо, чтобы жюри могло проверить обучения TLS-эмбеддинга. Если код отсутствует или невалиден, то такие решения рассматриваться не будут.

Важно

1. До конца соревнования вам обязательно надо отметить итоговое решение и прикрепить для него код. Если этого не сделать, то мы не сможем оценить ваше решение.

2. Прикрепляйте файл/архив для **итогового** решения. К сожалению, платформа устроена так, что при каждом новом сабмите кода файл будет перетераться. Поэтому когда соревнование завершится в прикрепленном файле должен быть код для выбранной посылки.

Baseline

К соревнованию мы приложили ipython-ноутбук с простым бейзлайном. Что в нем происходит:

1. TLS-шифры и TLS-кривые рассматриваются как единая строка текста с [SEP]-разделителем.
2. На TLS-текстах из прошлого пункта обучается BPE-токенайзер.
3. На всей train-выборке обучается простая нейронная-сеть. TLS-эмбеддинг – это усредненный эмбеддинг каждого токена.
4. Перебора параметров нет.

Решение достаточно простое и поэтому выбивает не очень высокий ROCAUC – около 0.68, но с этого можно начать.