Webcompat ML

Mar 03, 2020

Contents

CHAPTER 1

About

Webcompat ML is the machine learning initiative with the goal to support the process of collecting web compatibility issues. This can vary from collecting data, building ML models, running automation and deploying the infrastructure behind this.

1.1 About

Webcompat ML is the machine learning initiative with the goal to support the process of collecting web compatibility issues. This can vary from collecting data, building ML models, running automation and deploying the infrastructure behind this.

1.2 Repositories

All of our current codebase is hosted on github in the following repositories

- ML models
  - mozilla/webcompat-ml
- Deployment / Infrastructure as Code
  - mozilla/webcompat-ml-deploy
- Documentation
  - mozilla/webcompat-ml-docs
- Search
  - mozilla/webcompat-search

1.3 Installation

`webcompat-ml` is distributed as a python package. Here are the steps to install it:

```
$ git clone https://github.com/mozilla/webcompat-ml
$ cd webcompat-ml
$ python -m venv env
$ . env/bin/activate
$ pip install .
```

1.4 Elasticsearch

Currently our source of truth for data related to webcompat ML is our Elasticsearch database. It's purpose is both to allow the webcompat team query and visualize webcompat data, but also to be a source for building the model datasets.

The indexing of new data is handled by a cron-job that fetches new issues and events using the GitHub API.

The Kibana web UI is available here and is secured using Mozilla IAM.

For more information: mozilla/webcompat-search

1.5 Datasets

Our dataset is based in the webcompat GitHub issues submitted on webcompat/web-bugs. In order to build the dataset we used the GitHub API and specifically the following endpoints:

- List issues for a repository
- Get a single issue
- List events for an issue

Datasets are distributed in `CSV` format in the same repository with the rest of the codebase as Git-LFS objects under /datasets.

1.5.1 Available datasets

All events/issues

See also:

- datasets/all-issues-events.csv

This dataset includes all the issue-related historic data available in GitHub. It combines all the issue data available from the API combined with all the events from each issue. This allows us to go through the lifecycle of each issue and extract features.

Needs diagnosis

See also:

- datasets/needsdiagnosis-balanced-original-titles.csv

- datasets/needsdiagnosis-full-original-titles.csv

This dataset is used to train the `needsdiagnosis` model. Based on `All events/issues` we select all the `closed` issues and we extract the `needsdiagnosis` feature based on their events. An issue marked as `needsdiagnosis` is an issue that through its lifecycle (issue events) reached the `needsdiagnosis` milestone.

Because of the nature of the webcompat issues, the data is unbalanced and have much more data points for `needsdiagnosis = False`. Trying to factor this in our model we tried 2 different approaches. The one was to use all the data we have and the other was to balance the entry points so he wave the same number of `needsdiagnosis = True` and `needsdiagnosis = False` entries.

While going through the data, we noticed an inflation in our metrics which looked suspicious. It turned out that some of the titles get changed as part of the triaging process which hinted that `needsdiagnosis` target. In order to fix that we went through the events and extracted the original titles.

The metrics for the ML model gave more balanced results for the balanced dataset but since we are trying to tackle the problem of having too much noise in our input we currently use `datasets/needsdiagnosis-full-original-titles.csv`.

1.6 Metrics definitions

- `Accuracy` is the number of correct predictions made as a ratio of all predictions made.
- `Precision` is the number of correct positive results divided by the number of positive results predicted by the classifier.
- `Recall` is the number of correct positive results divided by the number of all relevant samples
- `f1` is a metric that combines precision and recall (mean of the 2 numbers)

1.7 Models

1.7.1 `needsdiagnosis`

Given a set of webcompat issues, this model classifies if an untriaged issues needs to be diagnosed.

Metrics

Report

Confusion matrix

Usage

```
$ Usage: webcompat-ml-needsdiagnosis [OPTIONS] COMMAND [ARGS]...

Options:
  --help  Show this message and exit.

Commands:
  evaluate
```

(continues on next page)

(continued from previous page)

```
  predict
  train
```

Classify issue

```
Usage: webcompat-ml-needsdiagnosis predict [OPTIONS]

Options:
  --data TEXT    Path to input CSV
  --model TEXT   Path to binary model
  --output TEXT  Predictions output
  --help         Show this message and exit.
```

Train model

```
Usage: webcompat-ml-needsdiagnosis train [OPTIONS]

Options:
  --data TEXT    Path to dataset CSV
  --output TEXT  Path to model binary path
  --help         Show this message and exit.
```

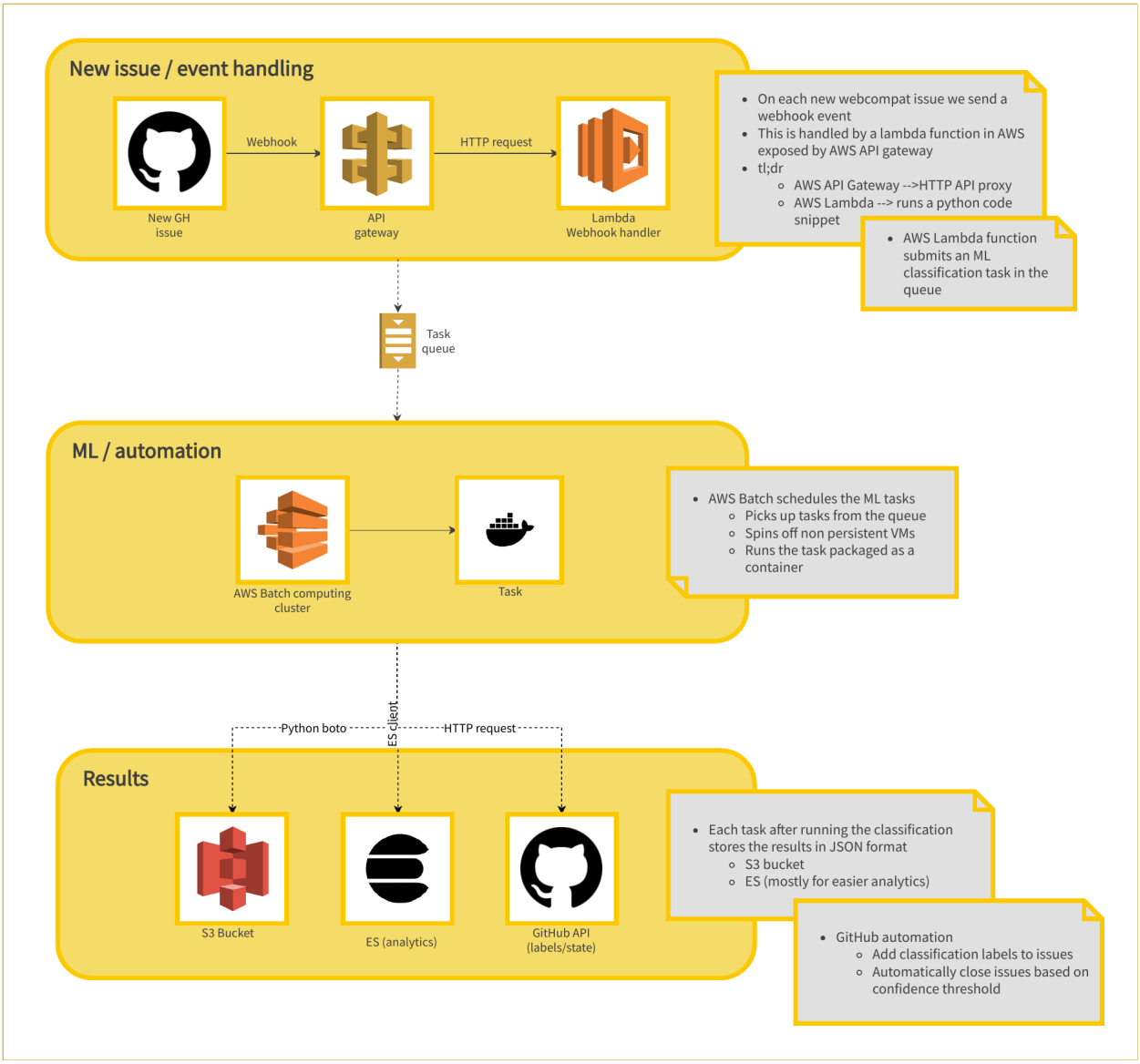
Evaluate model

```
Usage: webcompat-ml-needsdiagnosis evaluate [OPTIONS]

Options:
  --data TEXT  Path to input CSV
  --help       Show this message and exit.
```

1.8 Deployment

1.8.1 Architecture diagram



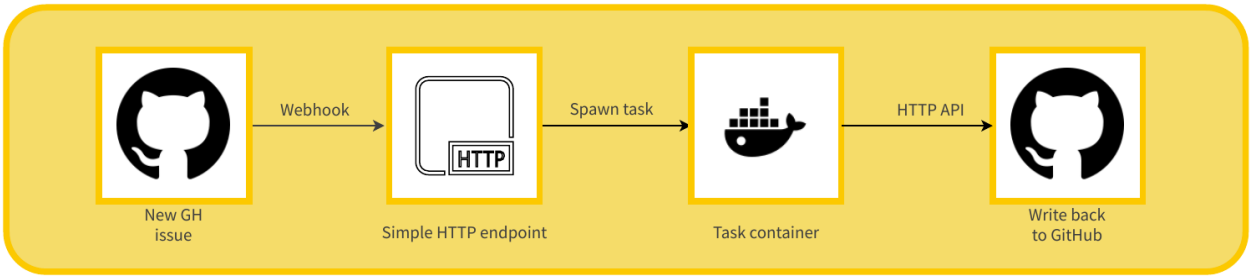
1.8.2 Technology stack

Current deployment is using the following technologies

- Docker
  - Package models and dependencies in a single container
  - Distribute docker images in Docker Hub
- AWS
  - AWS Lambda

        * Handle GitHub web hooks
    - AWS API Gateway
        * Expose lambda handler as API
    - AWS Batch
        * Schedule `webcompat-ml` tasks
    - AWS S3
        * Store `webcompat-ml` tasks results
- GitHub API / webhooks
    - Extract data to build datasets
    - Consume webhook events to trigger the automation

Even though most of the services are deployed in the cloud, all the primitives can be self hosted. The idea is that a webhook from GitHub triggers the automation and a simple HTTP API handles the request and spawns a task.



1.8.3 Infrastructure as Code

Dependencies

- terraform
- docker
- git-crypt
- webcompat-ml

About

All the infrastructure is managed as code and the codebase lives under mozilla/webcompat-ml-deploy.

To avoid over-complicating things, terraform is maintained in the git repository encrypted using git-crypt.

Important: For each change maintainers should make sure that the state is also checked in the repository. The state also leaks credentials so its important to always make sure that the state is encrypted before pushing.

All ML tasks should be described as a `Dockerfile` under `docker/` and should have the ML model prebundled.

1.8.4 Examples

Regular maintenance tasks

Build the `needsdiagnosis` model dataset

```
$ webcompat-ml-needsdiagnosis build-dataset --es-url "<URL>" --es-index-name="<INDEX>
↪" --es-doc-type="<TYPE>" --output "</path/to/dataset.csv>"
```

Train the `needsdiagnosis` model

```
$ webcompat-ml-needsdiagnosis train --data "</path/to/dataset.csv>" --output "</path/
↪to/model.bin>"
```

Releasing a new `needsdiagnosis` task image

```
$ cd webcompat-ml-deploy/docker/needsdiagnosis
$ docker build . -t ml-task:needsdiagnosis --build-arg MODEL_PATH="</path/to/model.
↪bin>"
$ docker tag ml-task:needsdiagnosis mozillawebcompat/ml-task:needsdiagnosis
$ docker push mozillawebcompat/ml-task:needsdiagnosis
```

Applying a terraform change

```
$ git-crypt unlock
$ terraform plan
$ terraform apply
$ git add .
$ git add terraform.tfstate
$ git add terraform.tfstate.backup
$ git commit -m '<change applied>'
```

CHAPTER 2

Indices and tables

- genindex
- search