Crash Report Analysis and Classification

Presenter: Wen Cui
About Crash Report Prioritization

Crash Report Overview

- Large amount of crash reports
- First come, first served may delay fix of important crashes
- Sometimes, prior knowledge is not enough

Need of Crash Report Prioritization Tools
To build the gap, this project

**S**  STUDY  Characteristics of crash report

**A**  CONSTRUCT  Use Few-shot learning, Similarity match, CNN for crash report classification

**C**  COMPARISON  Compare crash report classification tools

**P**  PROPOSE  Propose future direction of crash report classification
About Crash Report

<table>
<thead>
<tr>
<th>Signature</th>
<th>AsyncShutdownTimeout</th>
<th>profile-change-teardown</th>
<th>ServiceWorkerShutdownBlocker: shutting down Service Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUID</td>
<td>7a33180d-5c83-4951-8c4f-f8672b238323</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date Processed</td>
<td>2023-02-23 19:37:02 UTC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uptime</td>
<td>265.740 seconds (3 days, 1 hour and 49 minutes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last Crash</td>
<td>17,444,190 seconds before submission (28 weeks, 5 days and 21 hours)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Age</td>
<td>265.740 seconds since version was first installed (3 days, 1 hour and 49 minutes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install Time</td>
<td>2022-03-20 17:29:54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>Firefox</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Channel</td>
<td>nightly</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>113.0a1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build ID</td>
<td>20220315092641 (2022-03-19)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>macOS 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS Version</td>
<td>13.0.1 22A400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build Architecture</td>
<td>arm64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Info</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU Count</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adapter Vendor ID</td>
<td>0x1066</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adapter Device ID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Startup Crash</td>
<td>False</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Type</td>
<td>parent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOZ_CRASH Reason (Sanitized)</td>
<td>[Parent 18736, Main Thread] ###!!! ABORT: file /builds/worker/checkouts/gecko/dom/serviceworkers/ServiceWorkerShutdownBlocker.cpp:118</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crash Reason</td>
<td>EXC_BAD_ACCESS / KERN_INVALID_ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crash Address</td>
<td>0x0000000000000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Available Physical Memory</td>
<td>18,635,128.832 bytes (18.64 GB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMCheckCompatibility</td>
<td>True</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Feature Extraction

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-related</td>
<td>Total Physical memory, thread count, processor notes, CPU count</td>
</tr>
<tr>
<td>Crash-related</td>
<td>Method signature, prior fixes. Startup crash, module count</td>
</tr>
<tr>
<td>Other</td>
<td>Crash type, last crash, frame count</td>
</tr>
</tbody>
</table>
Crash Report Process

CRASH
Firefox crashes

REPORT
Failing stack trace collected

CLASSIFY
Classify automatically generated crash report

FIX
Bug fixed

Apply classification Algorithms
CLASSIFICATION APPROACHES

Existing Techniques

Idea: Convert crash report to sentences and perform text classification.

Not suitable for small amount of training data
Not-convincing definition “top crashes”

Machine Learning

New Approaches

Few shot Learning
Similarity Match
CNN
## RESEARCH QUESTIONS

<table>
<thead>
<tr>
<th>RQ1</th>
<th>RQ2</th>
</tr>
</thead>
<tbody>
<tr>
<td>How do we classify crash reports when there is little training data?</td>
<td>How does few-shot learning perform compared with other approaches in terms of crash report classification?</td>
</tr>
</tbody>
</table>
RQ1: DATA RETRIEVAL

STEP1: Data Collection

Mozilla Crash Report

- Collected data for consecutive 30 days

Feature Extraction

- Compact crash report to text

Filter and Assign Labels

- Assign label based on occurrence and fix time

Example Crash Report:

```
OOM | large | js::AutoEnterOOMUnsafeRegion::crash |
js::AutoEnterOOMUnsafeRegion::crash | JS::CallbackTracer::onEdge 8249831424
30>>> Start processing: 2023-03-06 00:37:34.170139+00:00
(processor_ip=172-31-2-221_us-west-2_compute_internal_8);8;597;2802;39;32;1
```
RQ1: DATA RETRIEVAL

STEP2: Create training and test set

Select crash report with labels

Collect new crash data without labels

---

cnn_train_2.csv

text,label
js::gc::HeaderWord::get 8587350016 30=>> Start processing: 2023-03-06 08:26:08.547884+00:00 (processor_ip=172-31-24-69_us-west-2_compute_internal_8);6:281;106;39;45,1
js::ObjectGroup::sweep 6442450944 34=>> Start processing: 2023-03-02 00:08:39.963803+00:00 (processor_ip=172-31-23-172_us-west-2_compute_internal_8);2;124;774942;275;27,1
<unknown in SHCore.dll> | CDeviceBase::DevQueryCallback 4149985280
67=>> Start processing: 2023-03-10 03:33:08.718026+00:00 (processor_ip=172-31-32-95_us-west-2_compute_internal_8);2;62;2169;157;14140344;10,1
COM | Small 1721008248 99=>> Start processing: 2023-03-02 00:22:42.895393+00:00 (processor_ip=172-31-33-118_us-west-2_compute_internal_8);2;11060;68720;160;396262;35,1
mozilla::dom::quota::QuotaManager::Shutdown<::operator() 17128787968 63=>> Start processing: 2023-02-26 00:55:15.548001+00:00 (processor_ip=172-31-37-241_us-west-2_compute_internal_7);8;2795;231;113;43068027;22,0

---

cnn_test.csv

text
shutdownhang |
js::frontend::ExtensibleCompilationStencil::ExtensibleCompilationStencil 4171517952 52=>> Start processing: 2023-03-07 10:48:04.558791+00:00 (processor_ip=172-31-7-120_us-west-2_compute_internal_9);4;136;1061;116;31277933;7
shutdownhang | NTQueryVirtualMemory 5259399168 35=>> Start processing: 2023-02-26 00:12:36.934910+00:00 (processor_ip=172-31-29-60_us-west-2_compute_internal_8);4;87;107;97;87698;7
mozilla::dom::quota::QuotaManager::Shutdown<::operator() 8483495936 73=>> Start processing: 2023-03-10 03:43:58.728717+00:00 (processor_ip=172-31-11-157_us-west-2_compute_internal_8);4;2795;14986;156;25002;22
AsyncShutdownTimeout | IOUtils: waiting for profileBeforeChange IO to complete | JSON store: writing data for 'targeting.snapshot' 8527294464 60=>> Start processing: 2023-03-06 00:38:02.761447+00:00 (processor_ip=172-31-11-157_us-west-2_compute_internal_8)
SignatureShutdownTimeout: Signature replaced with a Shutdown Timeout signature; was: """"Abort | NS_DebugBreak | nsDebugmpl::Abort | XPC耐用keyIndex"""";""""1:1047;422347;178;16
sys_read 3174739968 28=>> Start processing: 2023-03-07 10:34:16.647637+00:00 (processor_ip=172-31-32-95_us-west-2_compute_internal_8) msgd did not identify the crashing thread"""”;2;332;80756;137;0

---

Edited
RQ1: APPROACH DESCRIPTION

SETFIT

SETFIT (Sentence Transformer Fine-tuning), an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers (ST)

[Diagram: ST Fine tuning and Classification head training processes]
RQ1: APPROACH DESCRIPTION

**Few-shot Learning**

1. Load Training dataset

```python
dataset = load_dataset('csv', data_files={
    'train': ['data/train_2.csv'],
    'eval': ['data/eval_2.csv'],
    'cache_dir': './data/
}
)
```

2. Load a SetFit model from Hub

```python
model = SetFitModel.from_pretrained(
    "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2",
    cache_dir='./models/

```
RQ1: APPROACH DESCRIPTION

Few-shot Learning

3. Create Trainer

def trainer = SetFitTrainer:
    model=model,
    train_dataset=dataset['train'],
    eval_dataset=dataset['eval'],
    loss_class=CosineSimilarityLoss,
    metric="accuracy",
    batch_size=16,
    num_iterations=20,  # The number of text pairs to generate for contrastive learning
    num_epochs=1,  # The number of epochs to use for contrastive learning
    column_mapping={"text": "text", "label": "label"}  # Map dataset columns to text/label expected by trainer
RQ1: APPROACH DESCRIPTION

Few-shot Learning

4. Train, Evaluate, Save

```python
# Train and evaluate
trainer.train()
metrics = trainer.evaluate()

# save
trainer.model._save_pretrained(save_directory="./output/")
```

5. Inference

Preds = model(test_list)
RQ1: APPROACH DESCRIPTION

SIMILARITY MATCH

Levenshtein distance

Measure string difference: min single-character edits required to change one word into the other

\[
\text{lev}_{a,b}(i, j) = \begin{cases} 
\max(i, j) & \text{if } \min(i, j) = 0, \\
\min \begin{cases} 
\text{lev}_{a,b}(i - 1, j) + 1 \\
\text{lev}_{a,b}(i, j - 1) + 1 \\
\text{lev}_{a,b}(i - 1, j - 1) + 1_{(a_i \neq b_j)} 
\end{cases} & \text{otherwise.}
\end{cases}
\]
RQ1: APPROACH DESCRIPTION

**SIMILARITY MATCH**

- Income Crash Report
- Set of Crash Report with Known Classification
  - Compute Levenshtein distance
- Similarity Score
- Assign ranking for highest score

Enlarge training data
RQ1: APPROACH DESCRIPTION

CNN MODEL

Create
Train-validation split

Create the model

Create embedding matrix

Train the model
RQ1: APPROACH DESCRIPTION

CNN MODEL

Input

Word Embedding

Conv2D(3)

MaxPool2D

Conv2D(4)

MaxPool2D

Conv2D(5)

MaxPool2D

Concatenate

Flatten

Dropout

Output: Softmax
RQ2: APPROACH COMPARISON

Step

01
Create two sets of train-test set

02
Run both sets using 3 approaches

03
Output Comparison
DATA COLLECTION

Two sets of training-testing data

Each set contains 5 training reports with 10 × 5 test reports
## RQ2: RESULTS

<table>
<thead>
<tr>
<th>set_1</th>
<th>expected_output</th>
<th>fs_output</th>
<th>fs_accuracy</th>
<th>sim_output</th>
<th>sim_accuracy</th>
<th>cnn_output</th>
<th>cnn_accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_1</td>
<td>0 0 0 1 0</td>
<td>1 0 0 0 0</td>
<td>0.6</td>
<td>1 1 0 0 0</td>
<td>0.4</td>
<td>1 1 1 1 1</td>
<td>0.2</td>
</tr>
<tr>
<td>test_2</td>
<td>0 1 0 1 0</td>
<td>1 1 0 1 0</td>
<td>0.8</td>
<td>1 0 0 0 0</td>
<td>0.4</td>
<td>0 1 1 1 1</td>
<td>0.6</td>
</tr>
<tr>
<td>test_3</td>
<td>1 1 1 0 0</td>
<td>0 1 0 0 0</td>
<td>0.6</td>
<td>0 0 1 0 1</td>
<td>0.4</td>
<td>0 0 1 1 1</td>
<td>0.2</td>
</tr>
<tr>
<td>test_4</td>
<td>1 0 1 1 1</td>
<td>1 0 1 1 0</td>
<td>0.8</td>
<td>0 1 1 0 0</td>
<td>0.2</td>
<td>1 0 0 1 1</td>
<td>0.6</td>
</tr>
<tr>
<td>test_5</td>
<td>1 1 1 1 0</td>
<td>1 0 1 1 0</td>
<td>0.8</td>
<td>1 0 0 1 0</td>
<td>0.6</td>
<td>1 0 1 1 1</td>
<td>0.6</td>
</tr>
<tr>
<td>test_6</td>
<td>0 1 0 0 1</td>
<td>1 1 0 1 1</td>
<td>0.6</td>
<td>0 0 0 0 0</td>
<td>0.6</td>
<td>1 1 0 1 1</td>
<td>0.2</td>
</tr>
<tr>
<td>test_7</td>
<td>0 1 0 1 1</td>
<td>0 1 0 0 1</td>
<td>0.8</td>
<td>1 0 0 0 1</td>
<td>0.4</td>
<td>1 1 0 1 1</td>
<td>0.8</td>
</tr>
<tr>
<td>test_8</td>
<td>1 1 0 1 0</td>
<td>1 1 0 1 0</td>
<td>1</td>
<td>1 0 0 0 0</td>
<td>0.6</td>
<td>0 1 1 1 1</td>
<td>0.4</td>
</tr>
<tr>
<td>test_9</td>
<td>1 1 1 1 1</td>
<td>1 1 0 1 0</td>
<td>0.6</td>
<td>0 0 0 1 0</td>
<td>0.2</td>
<td>1 1 1 1 0</td>
<td>0.8</td>
</tr>
<tr>
<td>test_10</td>
<td>1 1 1 1 0</td>
<td>1 0 1 0 1</td>
<td>0.4</td>
<td>0 0 0 0 0</td>
<td>0.2</td>
<td>1 1 1 1 0</td>
<td>1</td>
</tr>
<tr>
<td>AVERAGE</td>
<td></td>
<td></td>
<td>0.7</td>
<td></td>
<td>0.4</td>
<td></td>
<td>0.54</td>
</tr>
</tbody>
</table>

Training set #1  →  Few shot  >  CNN  >  Similarity Match
RQ2: RESULTS

<table>
<thead>
<tr>
<th>set_2</th>
<th>expected_output</th>
<th>fs_output</th>
<th>fs_accuracy</th>
<th>sim_output</th>
<th>sim_accuracy</th>
<th>cnn_output</th>
<th>cnn_accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_11</td>
<td>0 1 1 1 1</td>
<td>1 1 1 1 1</td>
<td>0.8</td>
<td>1 1 1 1 1</td>
<td>0.8</td>
<td>1 1 1 1 1</td>
<td>0.8</td>
</tr>
<tr>
<td>test_12</td>
<td>1 1 1 0 1</td>
<td>1 1 1 1 1</td>
<td>0.8</td>
<td>1 1 1 1 1</td>
<td>0.8</td>
<td>1 1 1 1 1</td>
<td>0.8</td>
</tr>
<tr>
<td>test_13</td>
<td>0 0 1 1 0</td>
<td>1 0 1 1 0</td>
<td>0.8</td>
<td>0 1 1 1 0</td>
<td>0.8</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
</tr>
<tr>
<td>test_14</td>
<td>0 0 0 1 1</td>
<td>0 0 1 1 1</td>
<td>0.8</td>
<td>0 0 1 1 1</td>
<td>0.8</td>
<td>0 0 1 1 1</td>
<td>0.8</td>
</tr>
<tr>
<td>test_15</td>
<td>1 0 1 0 1</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
</tr>
<tr>
<td>test_16</td>
<td>0 1 0 0 0</td>
<td>1 1 1 1 1</td>
<td>0.2</td>
<td>1 1 1 1 1</td>
<td>0.2</td>
<td>1 1 1 1 1</td>
<td>0.2</td>
</tr>
<tr>
<td>test_17</td>
<td>1 0 1 1 1</td>
<td>1 0 1 1 1</td>
<td>1 1 0 1 0 1</td>
<td>0.8</td>
<td>1 1 1 0 1</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>test_18</td>
<td>0 1 0 1 1</td>
<td>1 1 0 1 1</td>
<td>0.8</td>
<td>0 1 0 1 1</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>test_19</td>
<td>0 1 0 1 1</td>
<td>1 1 0 1 1</td>
<td>0.8</td>
<td>1 1 0 1 1</td>
<td>0.8</td>
<td>1 1 0 1 1</td>
<td>0.8</td>
</tr>
<tr>
<td>test_20</td>
<td>0 1 1 0 1</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
<td>0 1 1 1 1</td>
<td>0.8</td>
<td>1 1 1 1 1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Training set #1 → Few shot ~ Similarity Match > CNN
RQ2: RESULTS

Training 1

Training 2

Test 1 to Test 10

Test 11 to Test 20

fs_output
sim_output
Cnn_output
FINDINGS

1. Similarity-based approach performance highly depends on training data quality.
2. CNN does not work well when the amount of training data is low.
3. Few shot learning performance could be improved by hyper-parameter tuning.
4. More data could be used to make results more convincing.
5. Overall, few shot learning outperforms the other two approaches with less training data.

Similarity-based approach performance highly depends on training data quality.
Challenges

Data Collection
Difficulty in collecting old data

Approach Comparison
Difficulty in finding other classification techniques

Report Analysis
Difficulty in finding resources of crash report features

Generalizability
Hard to illustrate generalizability of data due to short amount of training data
WHAT TO DO NEXT ...

- Crash Report classification techniques → Improve few shot learning performance
- Technique comparison → Investigate into more approaches and use more data for comparison
SETFIT (Sentence Transformer Fine-tuning), an efficient and prompt-free framework for few-shot fine-tuning of Sentence Transformers (ST)

**ST Fine tuning**
- Few-shot training data
- Enlarge training data
- Generate sentence pairs
- Fine-tune pre-trained ST

**Classification head training**
- Encode sentences with fine-tuned ST
- Train classification head

**CNN MODEL**

**Input**
- Word Embedding

**Conv2D(3)**
- MaxPool2D
- Concatenate
- Flatten

**Conv2D(4)**
- MaxPool2D
- Dropout

**Conv2D(5)**
- MaxPool2D
- Output: Softmax

**SIMILARITY MATCH**

**Income Crash Report**
- Set of Crash Report with Known Classification
- Compute Levenshtein distance
- Similarity Score
- Assign ranking for highest score

Overall, few shot learning outperforms the other two approaches with less training data.

More data could be used to make results more convincing.

CNN does not work well when the amount of training data is low.

Few shot learning performance could be improved by hyper-parameter tuning.

Similarity-based approach performance highly depends on training data quality.