AutoFR: Automated Filter Rule Generation for Adblocking

Hieu Le*, Salma Elmalaki*, Athina Markopoulou*, Zubair Shafiq†

*UCI  †UCD Davis
Adblockers

2022 PageFair Adblock Report
<table>
<thead>
<tr>
<th>Type</th>
<th>Filter Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>ad.com</td>
<td></td>
</tr>
<tr>
<td>eSLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FQDN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element Hiding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop JS Execution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Filter List Authors

Filter Rules

EasyList (ads)
## Filter List Authors

**EasyList** (ads)

### Filter Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Filter Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hostname</td>
<td>ad.com</td>
<td></td>
</tr>
<tr>
<td>eSLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FQDN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Element Hiding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop JS Execution</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What makes filter rule generation challenging?

**Blocking ads vs. Avoiding breakage**

This trade-off causes the need to have different rule granularities.
How does the site behave normally?
Are ads blocked with some breakage?
Are ads blocked with minimal breakage?
Ideal Region: Filter rules that block all ads with minimal breakage
Acceptable Region:
Filter rules that block some ads with minimal breakage
## Filter Rules

<table>
<thead>
<tr>
<th>Type</th>
<th>Filter Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eSLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FQDN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Path</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Preferred but may overblock → **breakage**
- Avoid breakage more easily → **easier to evade**
What makes filter rule generation challenging?

**Blocking ads vs. Avoiding breakage**

This trade-off causes the need to have different rule granularities.

**High Human Effort**

Affects scalability of creating and maintaining rules across the web and over time.
Filter List Author (FLA) Workflow

- **Blocks based on URL**
  - Network Requests
  - Tests by visual inspection
  - FLA
  - Browser
  - Site
Filter List Author (FLA) Workflow

1. Site
2. Browser
3. Blocks based on URL
4. Tests by visual inspection
5. FLA
6. Potential Filter Rules
7. Network Requests

- eSLD → ||example.com^ (example.com)
- FQDN → ||ads.example.com^ (ads.example.com)
Filter List Author (FLA) Workflow

1. Site
2. Browser
3. Network Requests
4. Tests by visual inspection
5. FLA
6. Potential Filter Rules
7. eSLD → ||example.com^e
8. FQDN → ||ads.example.com^e
9. Final Filter Rules
High Human Effort

Blocks based on filter rule

Tests by visual inspection

Browser

Site

FLA

Potential Filter Rules

Final Filter Rules

Network Requests

eSLD → ||example.com^
FQDN → ||ads.example.com^
We address the following challenges:

**What makes filter rule generation challenging?**

- **Blocking ads vs. Avoiding breakage**
  This trade-off causes the need to have different rule granularities.

- **High Human Effort**
  Affects scalability of creating and maintaining rules across the web and over time.

**AutoFR**

What makes filter rule generation challenging?
Outline

● Challenges to Filter Rule Generation
● **AutoFR: Formulation using Reinforcement Learning**
● AutoFR: Implementation for Scalability
● Evaluation
● Future Directions
AutoFR: Formulation as a Multi-Arm Bandit

- Site
- Browser
- Blocks based on filter rule
- Tests by visual inspection
- FLA
- Potential Filter Rules
- Site Dynamics
- Independent
- Final Filter Rules
AutoFR: Agent

Site → Browser → FLA

Blocks based on filter rule

Tests by visual inspection

Agent (t = 1 to T)

Potential Filter Rules

Final Filter Rules
AutoFR: Policy

- Site
- Browser
- Blocks based on filter rule
- Tests by visual inspection
- Agent
- Policy UCB
- Potential Filter Rules
- Final Filter Rules
AutoFR: Actions

- Site
- Browser
- Action (a) (filter rule)
- Tests by visual inspection
- Agent
- Policy UCB
- Action Space (Filter Rules)
- Final Filter Rules
AutoFR: Deriving Actions

Site → Browser

Action (a) (filter rule)
Tests by visual inspection

Network Requests

Policy UCB
Action Space (Filter Rules)

Agent

Final Filter Rules

eSLD → ||example.com^ FQDN → ||ads.example.com^ w/ Path → ||example.com/ads.js
AutoFR: Hierarchical Action Space

**Intuition:** try coarser grain filter rules first. If it causes too much breakage, then try more specific rules.

**Hierarchical Structure:** Reduces the number of actions that the agent needs to explore.
AutoFR: Environment

Environment

Inputs
Browser

Action (a)
(filter rule)

Tests by
visual inspection

Agent

Policy

Action Space
(Filter Rules)

Final
Filter Rules
AutoFR: Reward

Environment

Inputs

Browser

Action (a) (filter rule)

Reward

Agent

Policy

Action Space (Filter Rules)

Final Filter Rules

Reward considers trade-off blocking ads vs. avoiding breakage
AutoFR: Reward Environment

Inputs

Browser

AdHighlighter

CA 3
CI 11
CT 20

Ads
Images
Text

JS Injection
Difference:
Normal behavior vs. when rules are applied
\[ R_F(w, \hat{C}_A, B) = \begin{cases} 
-1 & \text{if } \hat{C}_A = 0 \\
0 & \text{if } \hat{C}_A > 0 , \ 1 - B < w \\
\hat{C}_A & \text{if } \hat{C}_A > 0 , \ 1 - B \geq w 
\end{cases} \]
Some breakage

Ideal

Normal

\[ R_F(w, \hat{C}_A, B) = \begin{cases} 
-1 & \text{if } \hat{C}_A = 0 \\
0 & \text{if } \hat{C}_A > 0, 1 - B < w \\
\hat{C}_A & \text{if } \hat{C}_A > 0, 1 - B \geq w 
\end{cases} \]
Some breakage

Ideal

Normal

Blocking Ads

Avoiding Breakage

1 - B

\( C_A \)

\( R_F(w, \hat{C}_A, B) = \)

\[
\begin{align*}
-1 & \quad \text{if } \hat{C}_A = 0 \\
0 & \quad \text{if } \hat{C}_A > 0, \ 1 - B < w \\
\hat{C}_A & \quad \text{if } \hat{C}_A > 0, \ 1 - B \geq w
\end{align*}
\]
Some breakage

Ideal

Normal

Blocking Ads

Avoiding Breakage $1 - B$

$\hat{C}_A$

$R_B(w, \hat{C}_A, B) = \begin{cases} 
-1 & \text{if } \hat{C}_A = 0 \\
0 & \text{if } \hat{C}_A > 0, 1 - B < w \\
\hat{C}_A & \text{if } \hat{C}_A > 0, 1 - B \geq w 
\end{cases}$

(3a) \hspace{5cm} (3b) \hspace{5cm} (3c)
$$R(w; A, B) = \begin{cases} 
0 & \text{if } \hat{C}_A = 0 \\
-1 & \text{if } \hat{C}_A > 0, 1-B < w \\
& \text{if } \hat{C}_A > 0, 1-B \geq w 
\end{cases}$$

(3a)

(3b)

(3c)
Some breakage

Ideal

Normal

$$R_F(w, \hat{C}_A, B) = \begin{cases} -1 & \text{if } \hat{C}_A = 0 \\ 0 & \text{if } \hat{C}_A > 0, 1 - B < w \\ \hat{C}_A & \text{if } \hat{C}_A > 0, 1 - B \geq w \end{cases}$$
\[ R_F(w, \hat{C}_A, B) = \begin{cases} 
-1 & \text{if } \hat{C}_A = 0 \\
0 & \text{if } \hat{C}_A > 0, 1 - B < w \\
\hat{C}_A & \text{if } \hat{C}_A > 0, 1 - B \geq w 
\end{cases} \]
\[ R_F(w, \hat{C}_A, B) = \begin{cases} -1 & \text{if } \hat{C}_A = 0 \\ 0 & \text{if } \hat{C}_A > 0, 1 - B < w \\ \hat{C}_A & \text{if } \hat{C}_A > 0, 1 - B \geq w \end{cases} \]
Some breakage

Ideal

Normal

\[ R_F(w, \hat{C}_A, B) = \begin{cases} 
-1 & \text{if } \hat{C}_A = 0 \\
0 & \text{if } \hat{C}_A > 0, \ 1 - B < w \\
\hat{C}_A & \text{if } \hat{C}_A > 0, \ 1 - B \geq w 
\end{cases} \]
Outline

- Challenges to Filter Rule Generation
- **AutoFR**: Formulation using Reinforcement Learning
- **AutoFR**: Implementation for Scalability
- Evaluation
- Future Directions
AutoFR: Real-world Bottleneck

⚠️ RL applied in the real world → x100-1K visits: → take hours per-site
AutoFR: Site Snapshot

Example

AdGraph: Iqbal et al. IEEE S&P 2020
AutoFR: Site Snapshot

Example

AdGraph: Iqbal et al. IEEE S&P 2020
AutoFR: Site Snapshot

Example

AdGraph: Iqbal et al. IEEE S&P 2020
Collect, Annotate, and Reuse

**Capture Site Dynamics:**

1. Collect: 10 raw graphs, each represents a visit to the site (no rules applied)

2. Annotate them and save locally as .graphml files

3. Reuse them
AutoFR: Reuse Site Snapshots

Environment

Inputs

Browser

Action (a) (filter rule)

Reward

Agent

Policy

Action Space (Filter Rules)

Final Filter Rules
AutoFR: Reuse Site Snapshots

For each action (filter rule), randomly select a site snapshot, apply rule

AutoFR → efficient and scalable
Outline

- Challenges to Filter Rule Generation
- AutoFR: Formulation using Reinforcement Learning
- AutoFR: Implementation for Scalability
- Evaluation
- Future Directions
AutoFR: Efficiency

13 hours
One site, 10 rules to explore, 1K visits to site (toy example)
(47 sec / visit * 1K visits)

1.6 minutes
One site, collect and use site snapshots (averaged on Top-5K sites)
AutoFR: Efficiency

1.6 minutes

One site, collect and use site snapshots (averaged on Top-5K sites)
AutoFR: Performance (Top–5K)

(b) AutoFR (In the Wild)

86%
AutoFR: Rule Granularity (Top–5K)

AutoFR generates rules with similar type distribution as EasyList on the Top-5K.
(Extensive) Evaluation

- Efficiency
- Performance vs. EasyList
- Rules vs. EasyList
  - Rule Type Distribution
  - Differences in Rules
- Per-site to Global Rules
- Longitudinal Analysis
  - Robustness after 6 months
  - Robustness every four-days
Future Directions

AutoFR

Different Rewards
Explore different rewards to generate rules

AutoFR for Element Hiding
Create rules that hide ads

AutoFR for Tracking
Create rules to block tracking

AutoFR for Mobile
Create rules for other platforms
AutoFR
A framework and tool that generate URL-based filter rules to block ads while avoiding visual breakage automatically

Paper and Artifacts

Github: [https://github.com/UCI-Networking-Group/AutoFR](https://github.com/UCI-Networking-Group/AutoFR)
Dataset: [athinagroup.eng.uci.edu/projects/ats-on-the-web/autofr-dataset/](athinagroup.eng.uci.edu/projects/ats-on-the-web/autofr-dataset/)
Credits

Special thanks to all the people who made and released these awesome resources for free:

▷ Presentation template by SlidesCarnival
▷ Photographs by Unsplash