

DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps

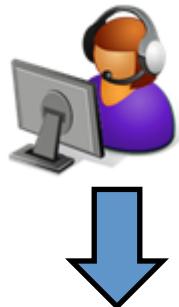
Bin Liu

Suman Nath, Ramesh Govindan, Jie Liu



The Mobile Ad Ecosystem

App Developer



Paid by Impressions

Paid by User Clicks



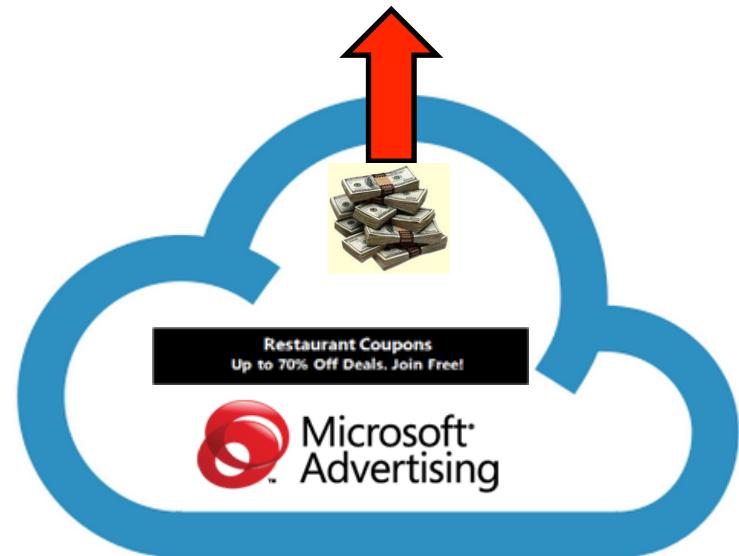
See/Click Ads



App User

Phone/Tablet App

Ad Plugin



Ad Network

Introduction



DECAF



Evaluation



Characterization



Conclusion

Mobile Ad Fraud

App developers have incentive to commit fraud
by inflating clicks and impressions



Ad Fraud: a Big Business

Very large mobile marketplaces

1 billion dollars lost due to ad fraud in 2013

Google Play Will Beat Apple App Store To 1,000,000 Apps

If growth remains the same for each mobile operating system, Android will have one million apps in its app store months before Apple's iOS.

By Dan Riedlko on January 08, 2013



Apple and Google have been playing the numbers game about iOS and Android for years. Apple had the headstart, of course, but Android first caught on the number of smartphones that it ships and activates every day. Apple maintained its lead when it comes to sheer volume of apps in its App Store vs Android's Google Play until October 2011.

Apple announces 1 million apps in the App Store; more than 1 billion songs played on iTunes

By Nathan Ingraham on October 08, 2011



Apple has just kicked off its annual iPad event, and as usual Tim Cook is on stage taking members, just as he did at the start of last month's iPhone 5S and 5C launch. Cook says that more than 1 million apps are in the App Store, up from 900,000 at WWDC in June. More than 60 billion total apps have been downloaded, up from the big 50 billion.

Report: 40 percent of mobile ad clicks are fraud or accidents

15 Comments [Twitter](#) [Facebook](#) [LinkedIn](#) +1 [Email](#)

A▼ A▲

SUMMARY: A study by mobile app marketing platform Trademob found that 40 percent of mobile clicks on ads are essentially useless, the result of accidental presses or fraud. That's one of the challenges facing mobile advertising, which is still lagging online advertising.



Al Urbanski, Senior Editor
Follow @AlUrbanski

Bots Mobilize

Mobile marketing networks are getting good at geo-fencing for advertisers, but they need to get better at fencing out bogus ad traffic perpetrated by robotic programs.

Computer hackers are intensifying their infiltration of mobile ad networks, according to a Bot Traffic Market Advisory released this week by Solve Media, whose Type-In advertising platform is designed to repel non-human visitors. The ad platform's tracking of some 7 million mobile transactions in Q1 2013 identified 29% of them as suspicious and confirmed that 14% were confirmed as bots.



Bucks to battle the bots.

"It's easier for bad guys to infiltrate mobile," says Solve Media CEO Ari Jacoby. "It's a much newer technology and the industry lacks the sophistication to stop it. We're seeing an alarming increase in mobile traffic comprised of bots and not humans." Advertisers will waste close to \$1 billion on bogus mobile publishers and leads in 2013, the study estimates.

Introduction

DECAF

Evaluation

Characterization

Conclusion

Placement Ad Fraud

We explore a sub-class of ad fraud, called placement ad fraud

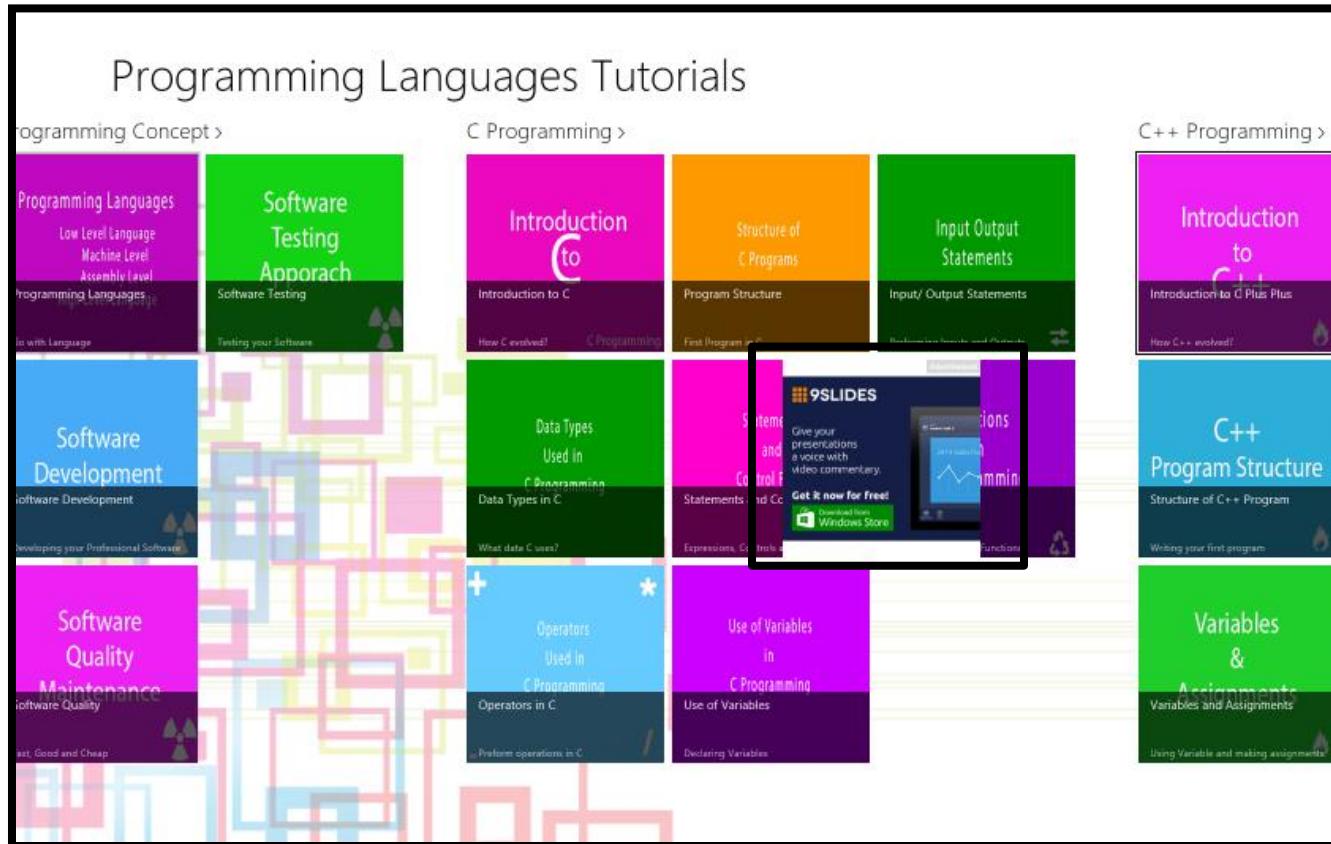
Developers manipulate visual layouts to trigger invisible impressions or unintentional clicks

Microsoft Advertising Prohibits Placement Ad Fraud

“A developer must not edit, resize, modify, filter, obscure, hide, make transparent, or reorder any advertising”

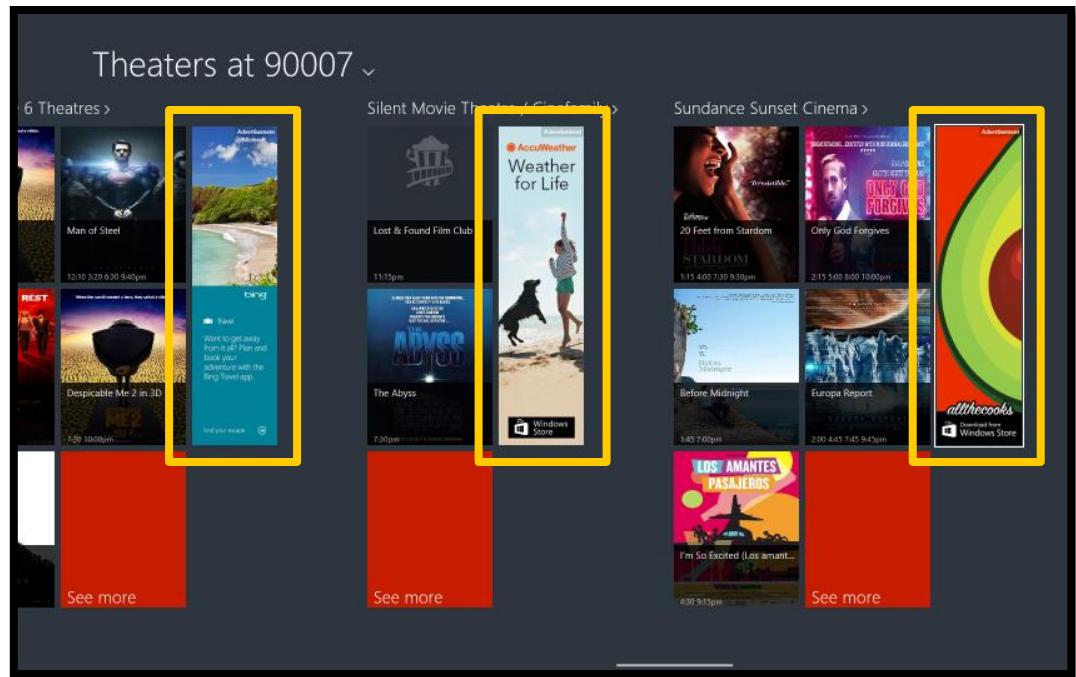
Placement Ad Fraud Examples

Intrusive ads



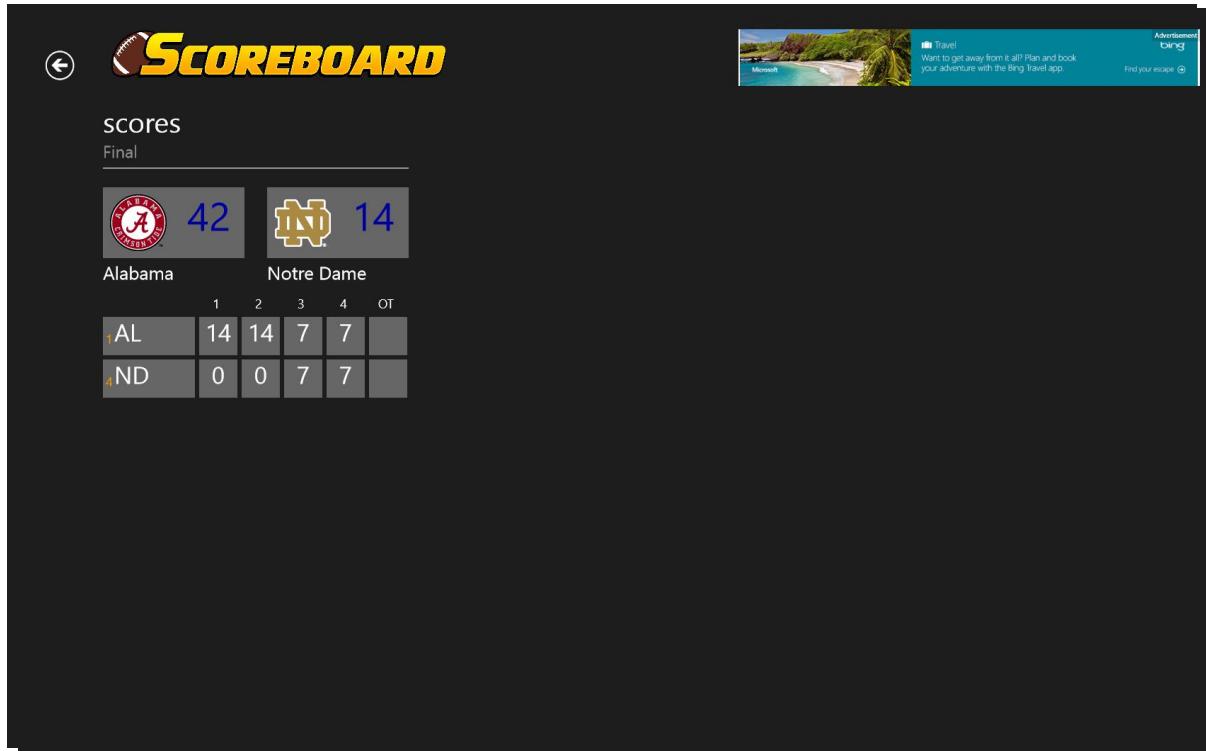
Placement Ad Fraud Examples

Many ads



Placement Ad Fraud Examples

Hidden ads



Current Approach

Manual inspection, which is labor-intensive and error-prone

Several tens of minutes to manually scan one app

Cannot detect some placement ad fraud, like hidden ads



Goal

To design an automated system for detecting placement fraud



Challenges

Challenge 1: Scaling to thousands of visually complex apps

Reddit >

Food Pairings (1st Category)

Visit [WINETABLET](#)

OUTDOORLIFE

THE ULTIMATE RESOURCE FOR THE OUTDOOR ENTHUSIAST!

Download from Windows Store

Featured Wine Lists

TOS546 | 16 - 2023

Login to WineTablet

Food Pairings Cheese

Aged Cheddar Gouda Soft Pungent Cheese

Blue Cheese Mozzarella Swiss

Brie Parmesan Pepper Cheese

Cream Cheese Provolone

Dry Jack Cheese Ricotta

Pepper Cheese Sharp Cheddar

Smoked Gouda

Reddit To Go! Suggested features, report

1058 points 71 comments

We We to f AQ

The Seeing Eye Flying Lemur to TheLastAirbender house_your_reddit_on 10 hours ago [quequerido.deviantart.com]

421 3 comments

The landing page has tiles for the front page, subreddits, etc.

Chateau St. Michelle & Dr. Loosen Riesling 2008 United States, Washington, 2008

Chateau Montelena Riesling 2008 United States, California, Napa Valley, 2008

Carl Schenkt-Wagner Riesling 2009 Germany, Mosel-Saar-Ruwer, 2009

Dr. Loosen Riesling 2009 Germany, Mosel-Saar-Ruwer, 2009

Domaine Weinbach Reserve Pinot Noir 2009 France, Alsace, 2009

Joh. Jos. Prüm Riesling Auslese 2009 Germany, Mosel-Saar-Ruwer, 2009

Kung Fu Girl Riesling 2009 Germany, Mosel-Saar-Ruwer, 2009

Trimbach Riesling 2009 Willamette Valley, Oregon, 2009

Popular SCIENCE

THE ULTIMATE GUIDE TO WHAT'S NEW AND WHAT'S NEXT!

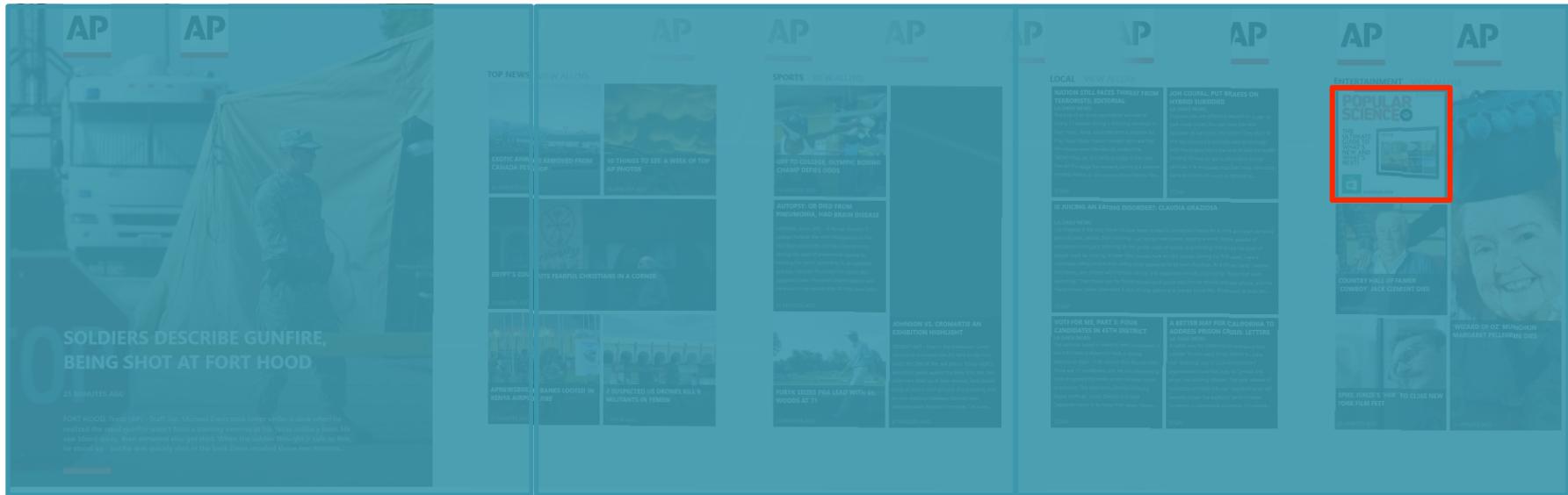
Download from Windows Store

(accuracy) and processing more apps in a given time (scalability)

Ads can appear on any app page. Apps can have many pages.

Challenges

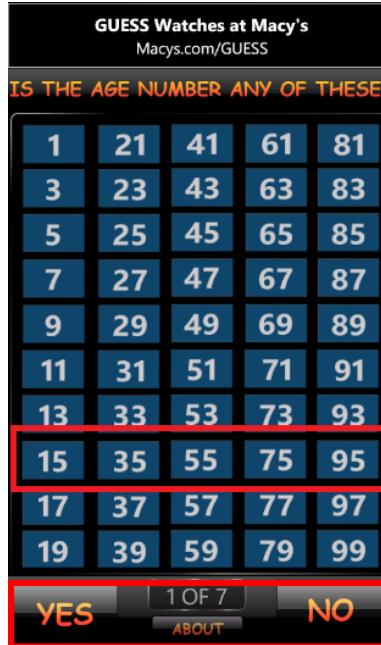
Challenge 2: accurately and quickly identify fraud



**“Sliding Screen” Problem
(in a Panoramic Page)**

Challenges

Challenge 2: accurately and quickly identify fraud



Hidden Ads
(Z-index not Available)

Our Approach – Dynamic Analysis

Use UI-automation based dynamic analysis to detect placement fraud



Dealing with Visual Complexity

Develop automated scalable navigation of app pages through dynamic execution

Accurate Fraud Detection

Design several efficient fraud detectors, one for each fraud type

Contributions

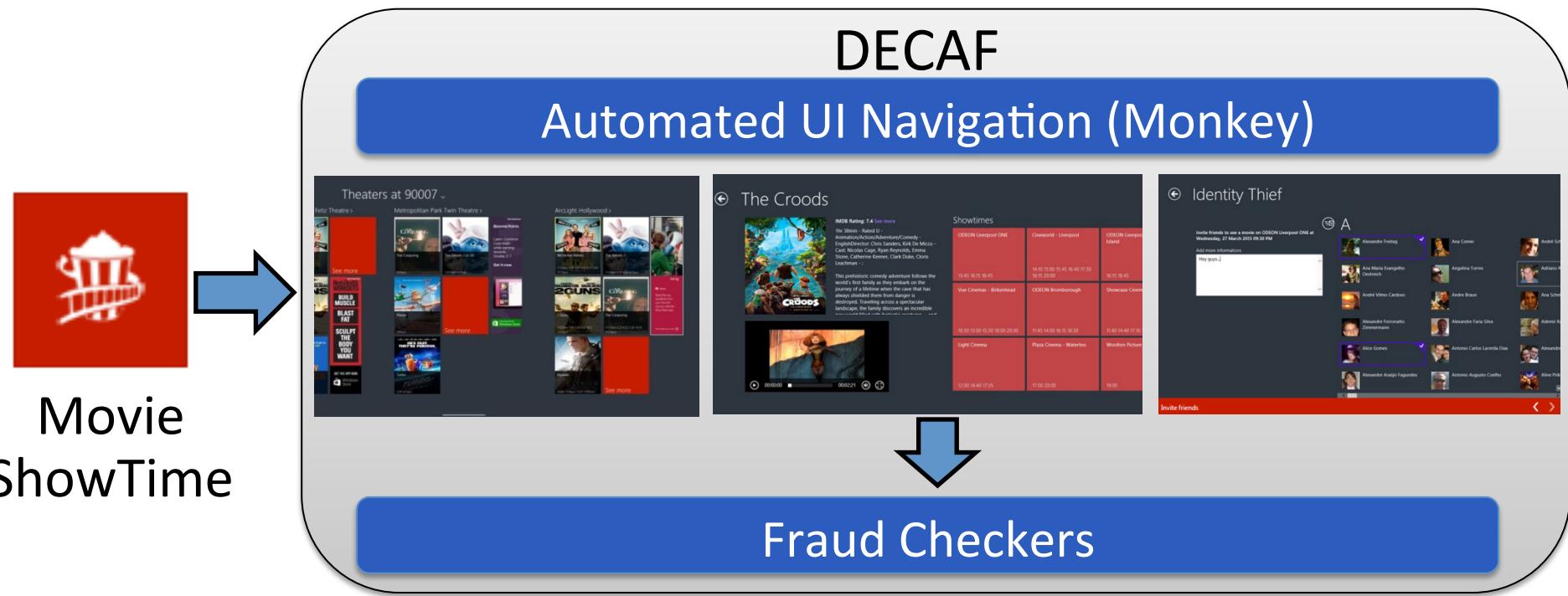
Design and implementation of the DECAF system to detect placement fraud

Characterization of placement fraud by analyzing 50,000 Windows Phone apps and 1,150 tablet apps using DECAF

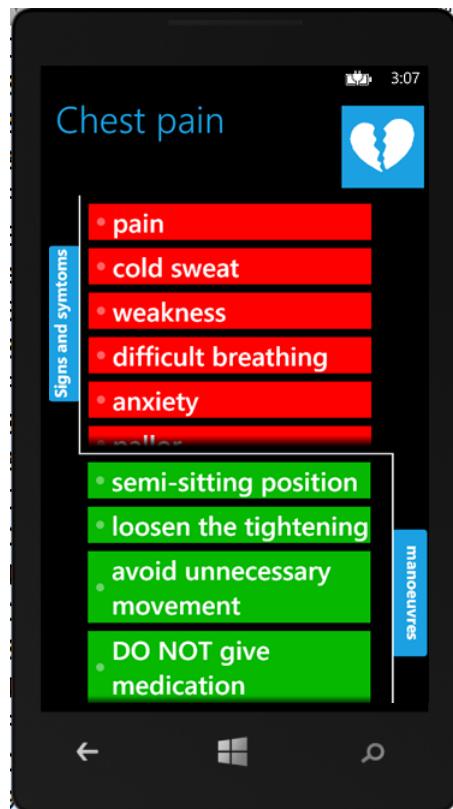
Deployment of DECAF in the ad fraud team at Microsoft, which has helped detect many instances of fraud



DECAF Overview



Automated UI Navigation



UI Extraction

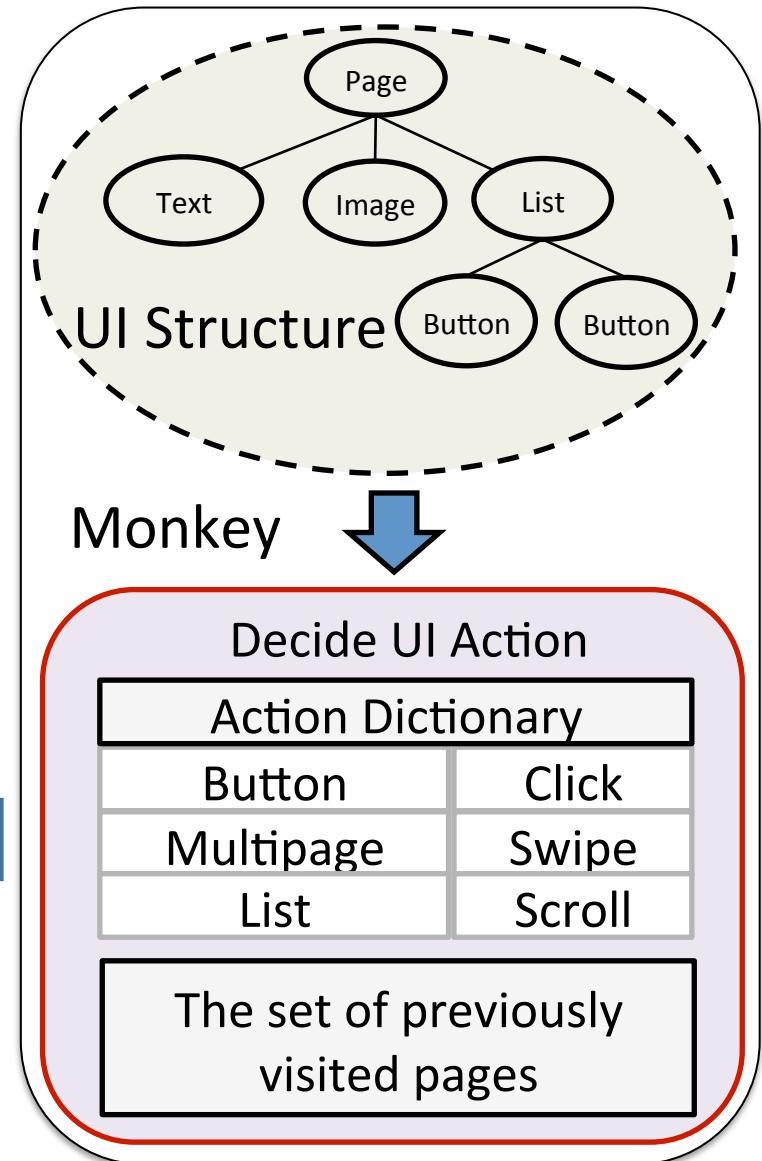
UI Extraction Channel

UI Action Channel

Click “Continue” Button

UI Action

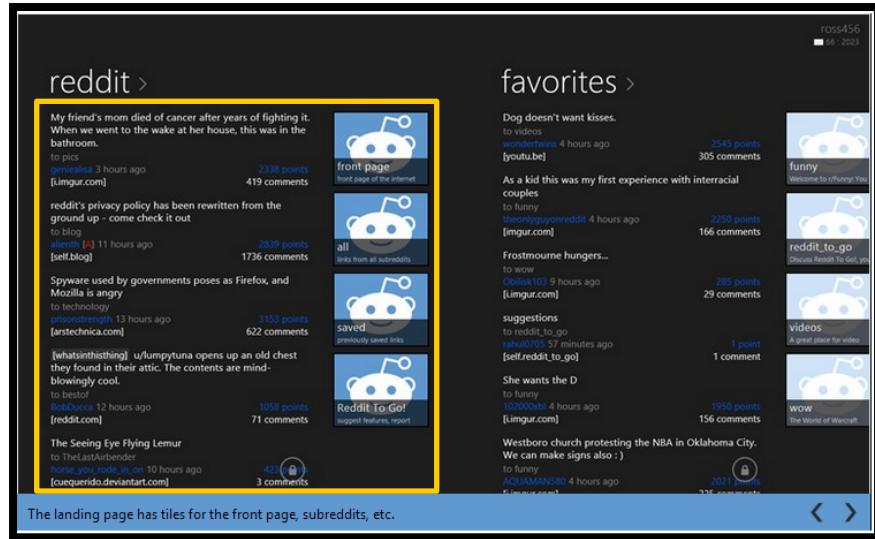
Next App Page



Problem: Reducing the Search Space

Avoid clicking UI elements on previously visited pages?

UI page space can be practically infinite



The post list is updated every several minutes

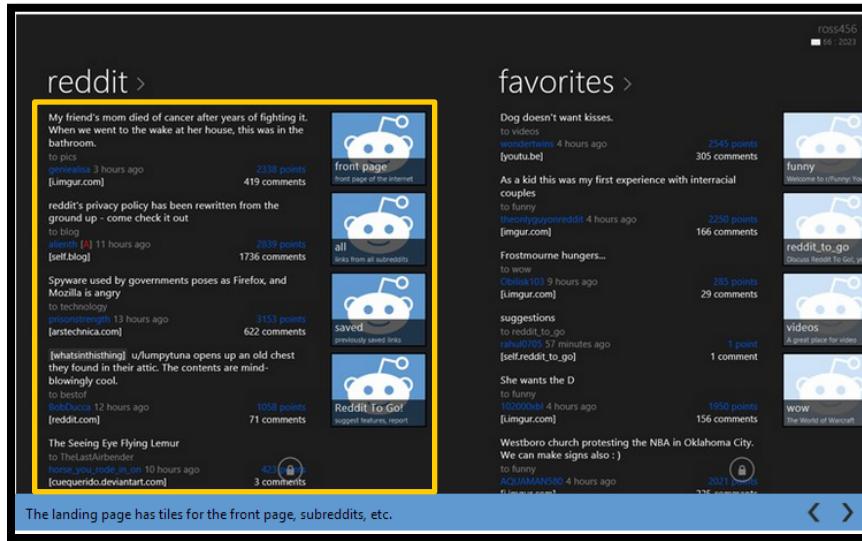
One Reddit App Page

Reducing the Search Space

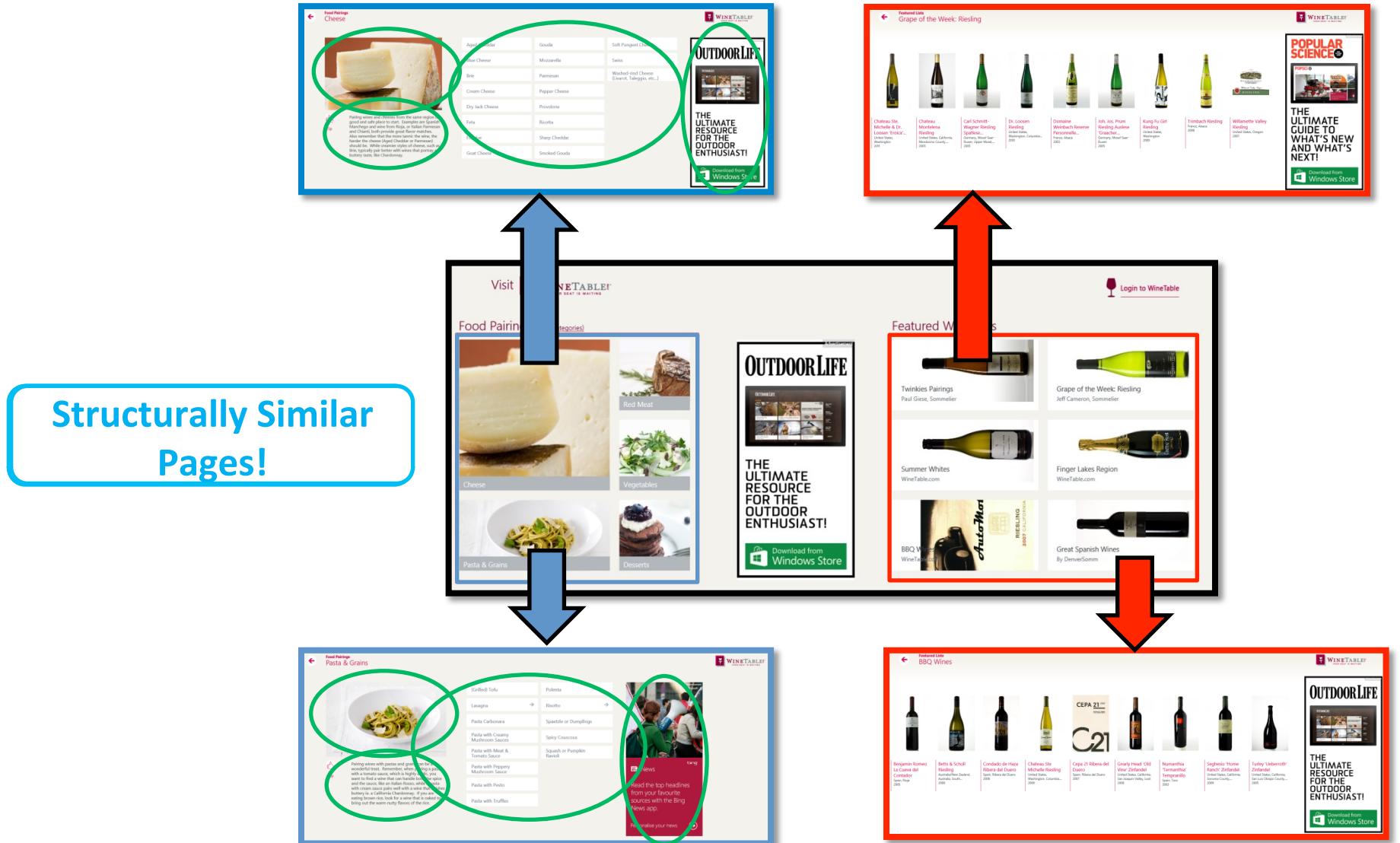
Key observation

For placement fraud, it is sufficient to visit structurally dissimilar pages

Two pages can be structurally similar even if their content differs



Structurally Similar Pages



Introduction

DECAF

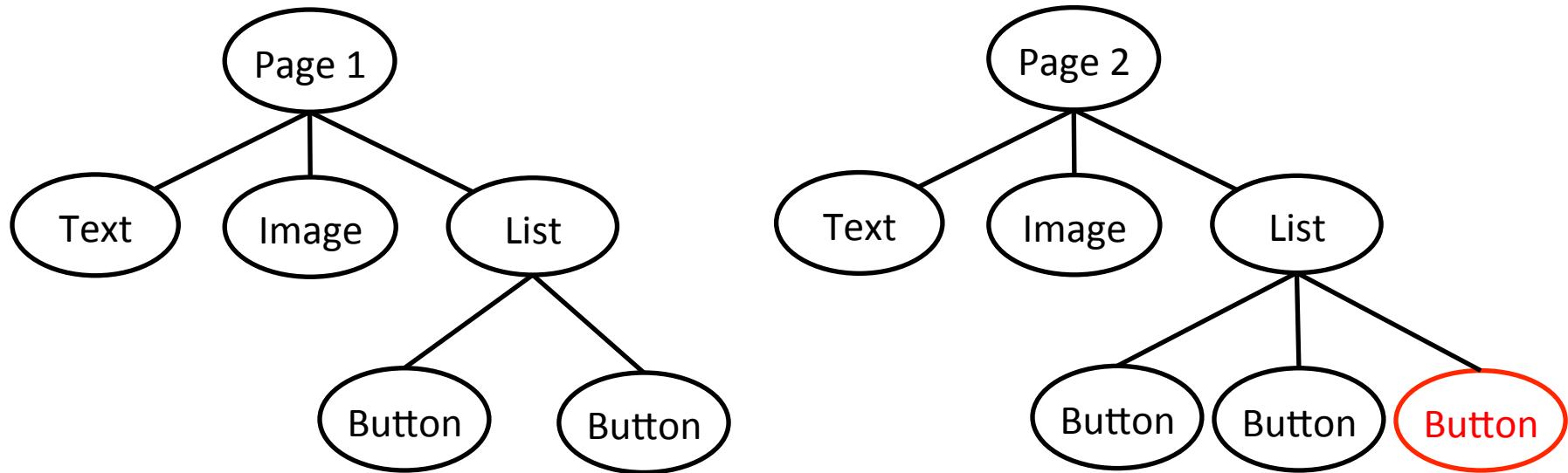
Evaluation

Characterization

Conclusion

Determining Structural Similarity

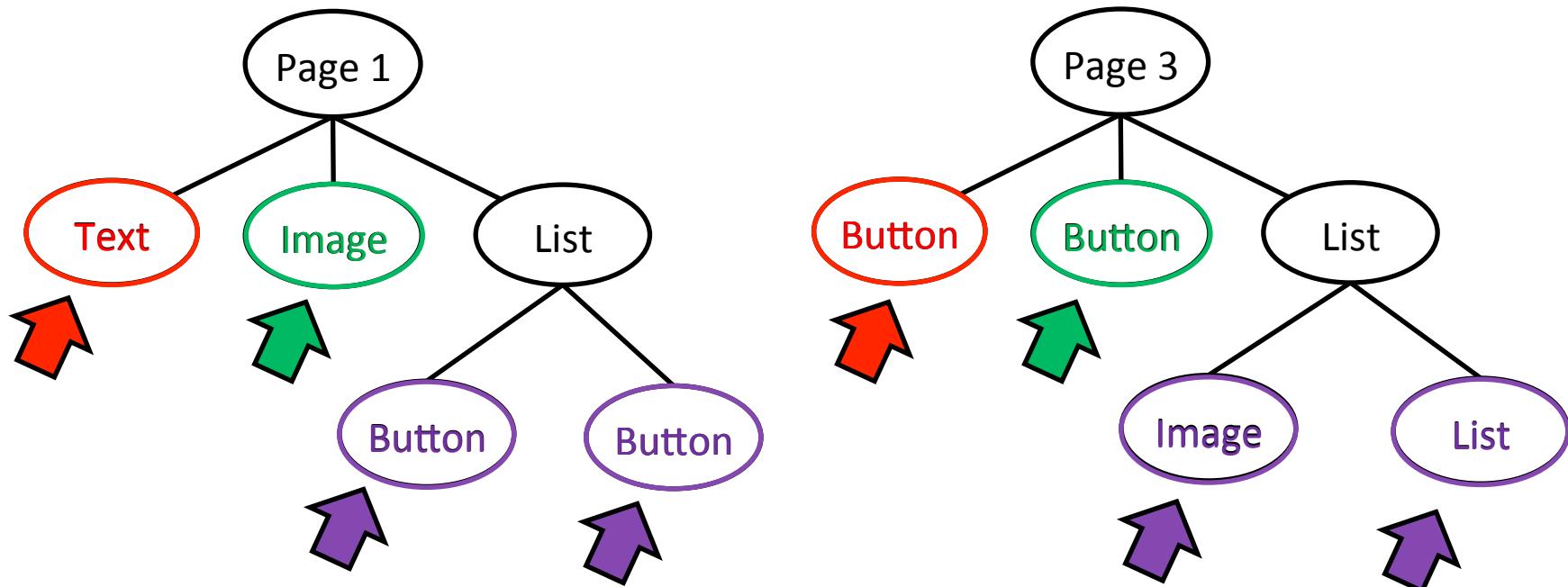
Two pages are structurally similar if they have “similar” UI hierarchies



Structurally Similar

Determining Structural Similarity

Two pages are structurally similar if they have “similar” UI hierarchies

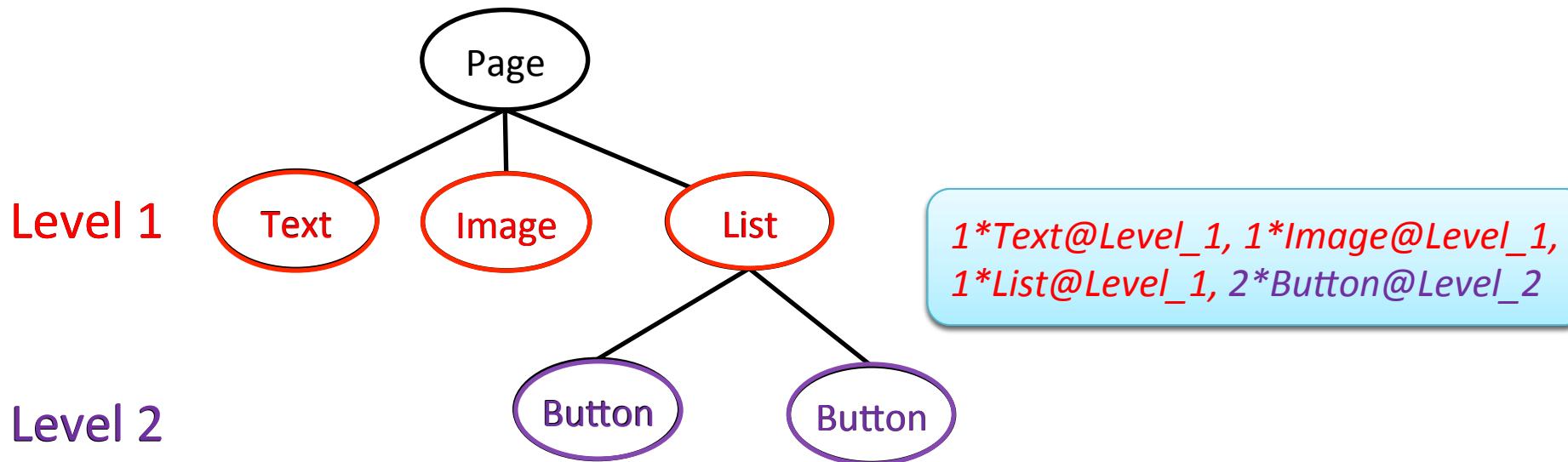


Structurally Dissimilar

Defining Structural Similarity

Feature vector defined on UI elements

Encodes type of UI element and position in hierarchy

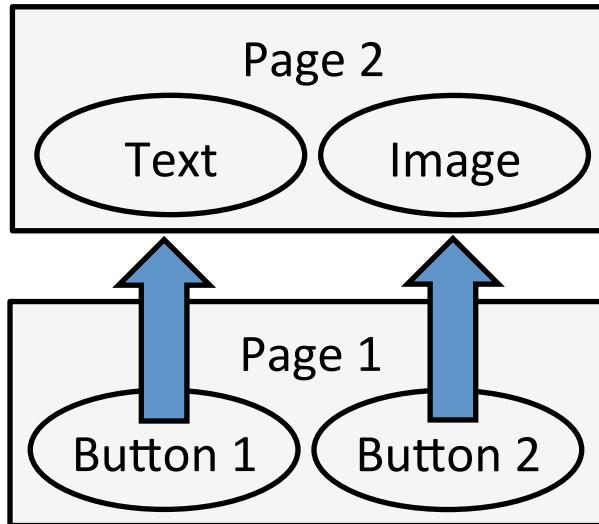


Use cosine similarity metric

Users specify a *similarity threshold*

Problem: Avoid previously visited states

Monkey can waste time by visiting previously visited pages



Not necessary to click **Button 2**. However, to detect this...

Extra time (up to several minutes) for backtracking is needed

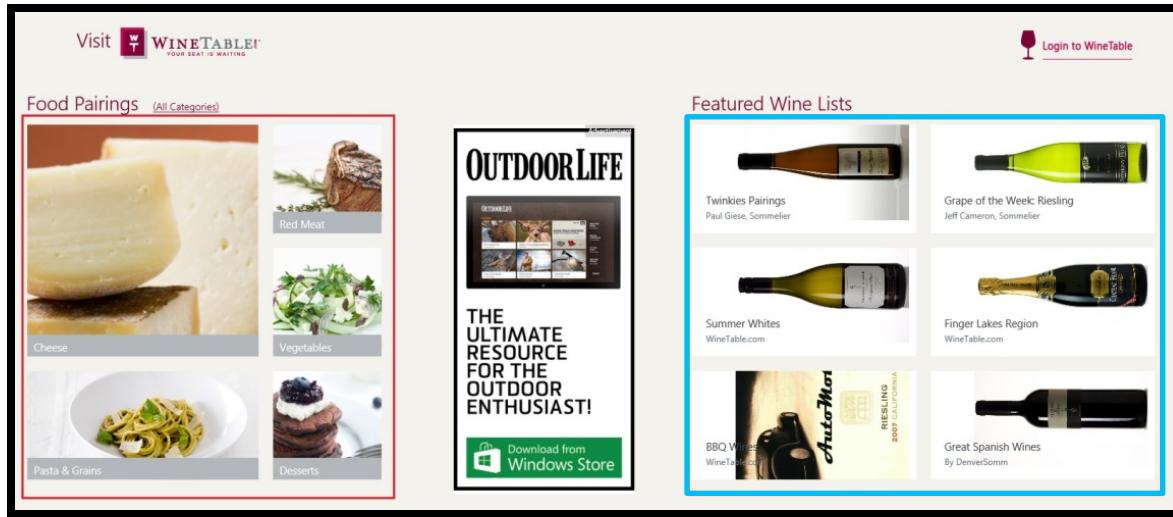
The Monkey needs to anyway go back to page 1 again and click button 2

Avoiding Previously Visited States



To avoid backtracking costs, can we predict if two buttons on a page lead to structurally similar pages?

Avoiding Revisiting Similar Pages



Our method is to use machine learning classifiers

Two buttons that have a similar neighborhoods in UI hierarchy
likely to lead to structurally similar pages

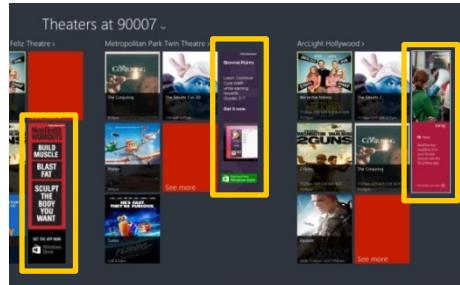
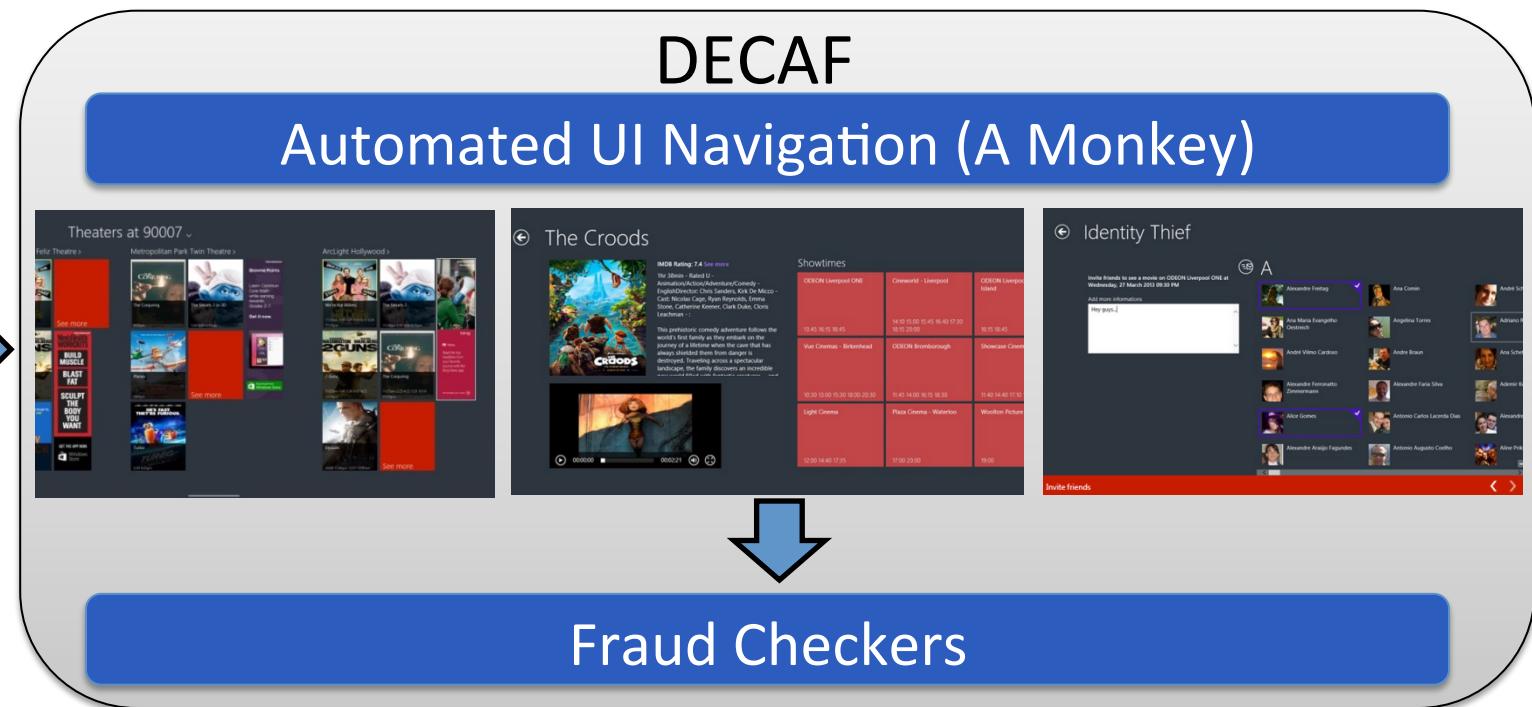
DECAF Overview

DECAF

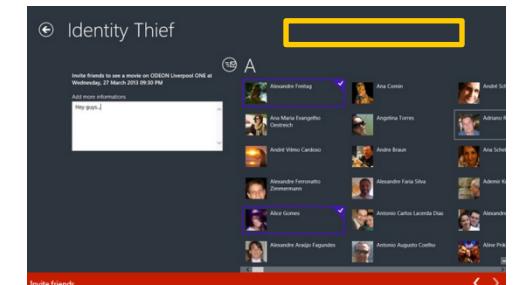
Automated UI Navigation (A Monkey)



Movie
ShowTime



Many Ads



Hidden Ads

Introduction

DECAF

Evaluation

Characterization

Conclusion

Fraud Checkers

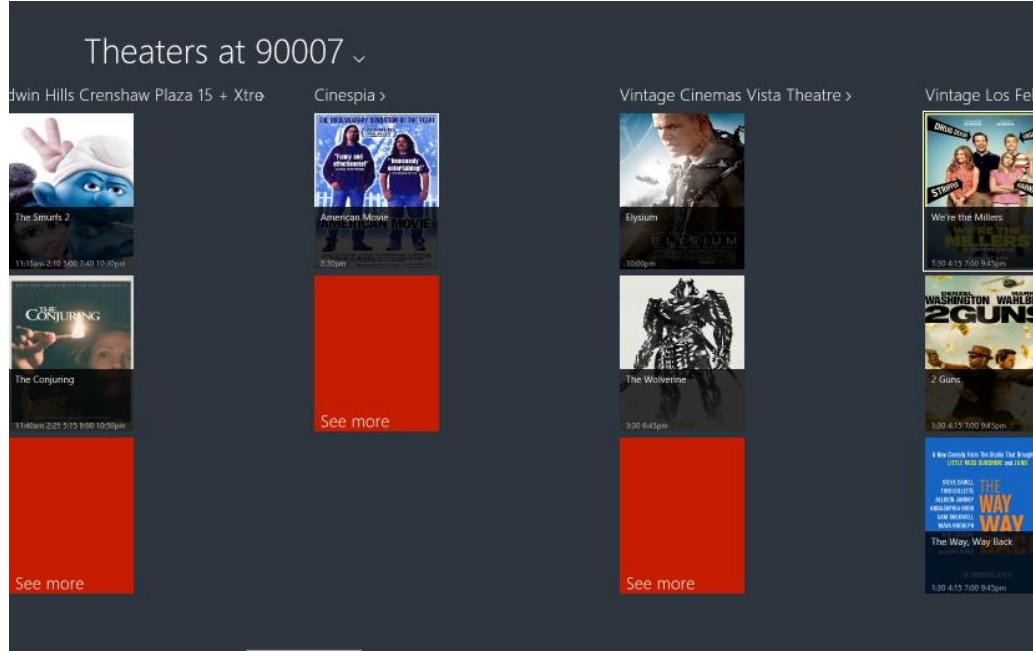
Input to checkers: structural data of ad and non-ad elements

Fraud Type	Checker Summary
Invisible/Hidden Ads	Whether visual elements are overlapped with ads
Smaller Ads	Compare the actual display size of the ad with the minimal valid size
Intrusive Ads	Compare the distance between an ad and clickable non-ad elements
Many Ads	Whether the number of viewable ads is more than the maximum allowed

Efficient Many-Ads Checkers

Many ads in one display screen

Challenge: the “sliding screen” problem

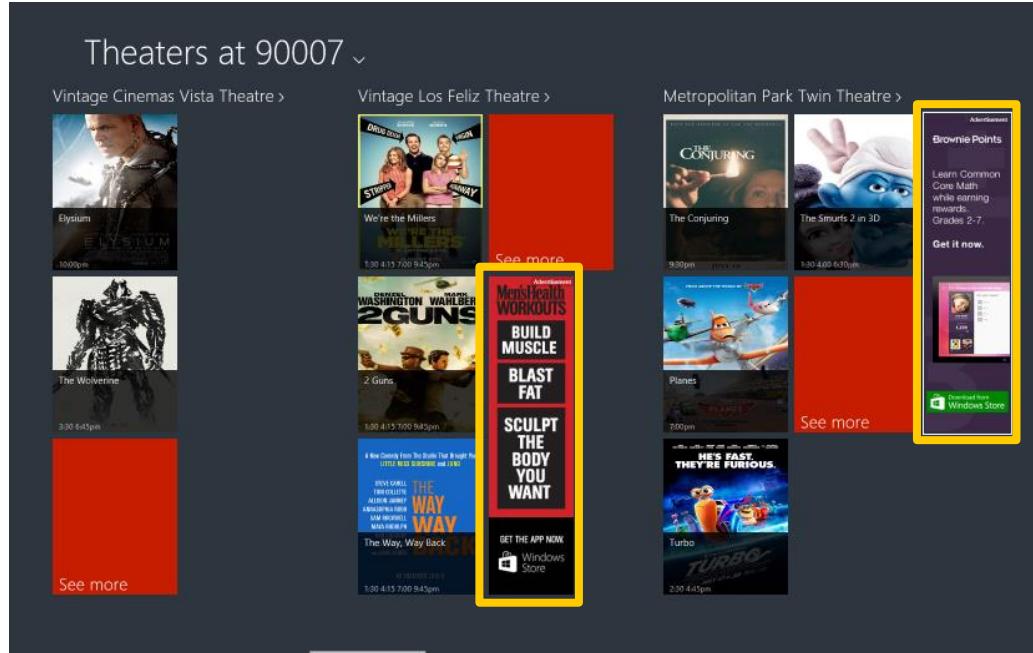


No Ad

Efficient Many-Ads Checkers

Many ads in one display screen

Challenge: the “sliding screen” problem



Two Ads

Introduction

DECAF

Evaluation

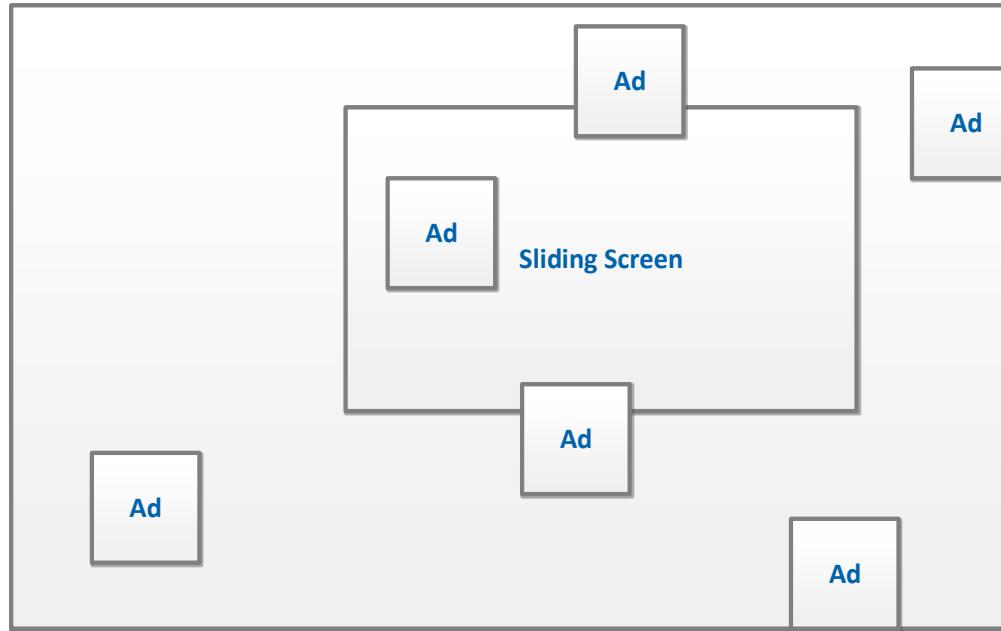
Characterization

Conclusion

Efficient Many-Ad Checker

Many ads in one display screen

Challenge: the “sliding screen” problem



We have designed an efficient algorithm to detect many-ad fraud

Other Optimizations

State Importance Assessment to further reduce the number of app pages that the Monkey needs to explore

Rendering Order Inference and Proxy-Assisted Screen Analysis to efficiently detect hidden ads



Evaluation and Characterization

Basic Setup

Run DECAF on each app for 20 minutes

Collect information from structurally different pages



Structural Page Coverage

Manually find ground truth number of page structure

Limited to 100 top free apps from the tablet app store

Evaluation and Characterization

Basic Setup

Run DECAF on each app for 20 minutes

Collect information from structurally different pages

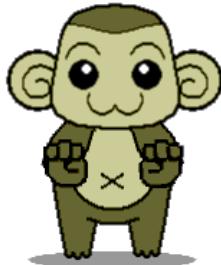


Characterization of Placement Ad Fraud

Run DECAF on 50,000 phone apps and 1,150 tablet apps

Characterize fraud by fraud type, rating and publisher

Structural Page Coverage



A Basic Monkey



A Classifier-Enhanced Monkey

71 apps finish with the Classifier-Enhanced Monkey, but only 30 finish with the Basic monkey

29 apps cannot finish in 20 minutes

The Monkey fails to recognize some clickable elements

Some scenarios require app-specific text input that Monkey cannot handle

Some apps simply have a very large set of structural pages

Characterizing Fraud by Types

Fraud Type	Phone Apps (1000+)	Tablet Apps (50+)
Too Many Ads	11%	4%
Smaller Ads	33%	48%
Hidden Ads	47%	32%
Intrusive Ads	9%	16%

1,000+ phone apps (out of 50,000) and 50+ tablet apps (out of 1,150)
commit at least one fraud

Characterizing Fraud by Types

Fraud Type	Phone Apps (1000+)	Tablet Apps (50+)
Too Many Ads	11%	4%
Smaller Ads	33%	48%
Hidden Ads	47%	32%
Intrusive Ads	9%	16%

“Hidden Ads” violations are more prevalent on the phone, which has a smaller screen for displaying content

Characterizing Fraud by Types

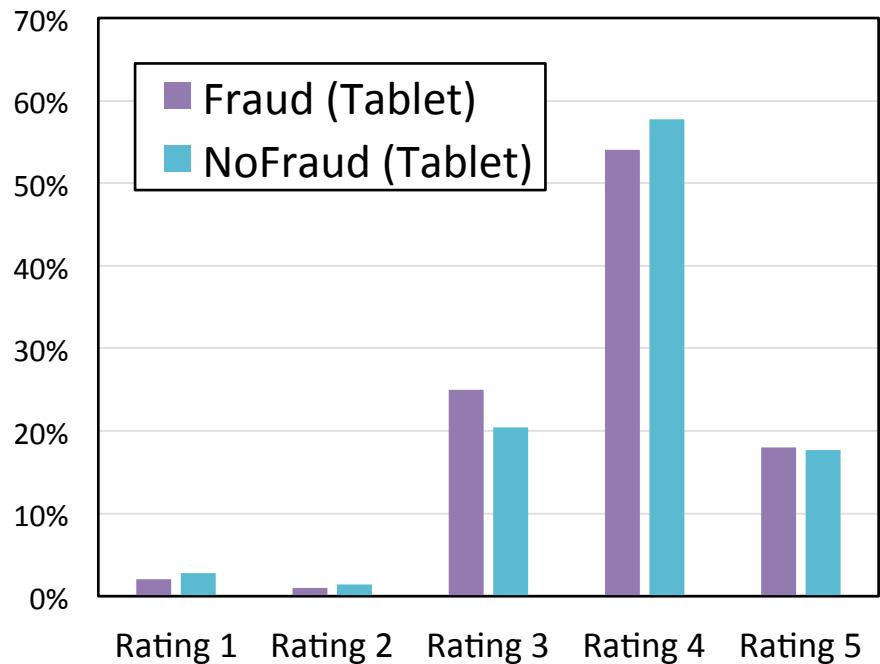
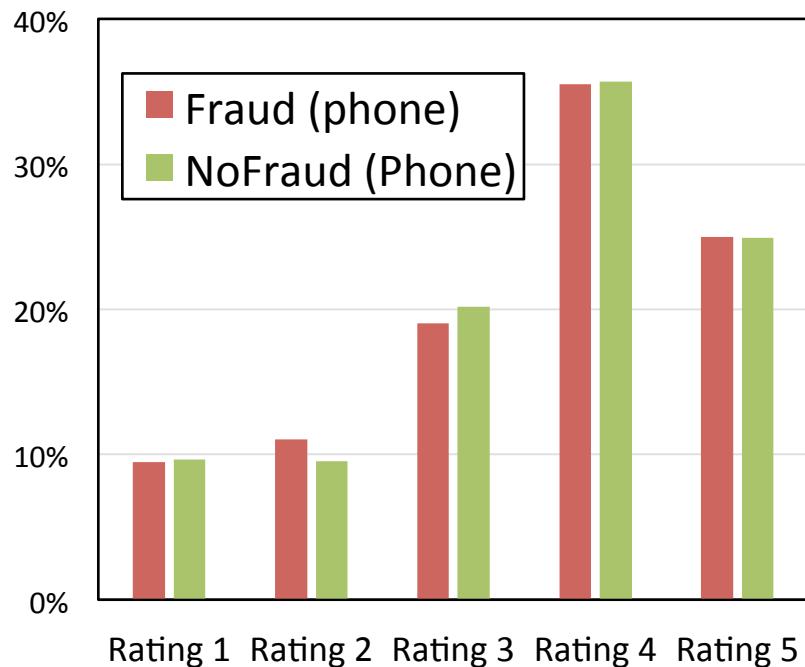
Fraud Type	Phone Apps (1000+)	Tablet Apps (50+)
Too Many Ads	11%	4%
Smaller Ads	33%	48%
Hidden Ads	47%	32%
Intrusive Ads	9%	16%

“Intrusive Ads” violations are more prevalent on the tablet, which has richer controls to be used to trigger accidental clicks

Characterizing Fraud by Rating

Rating values are rounded to a number from 1-5

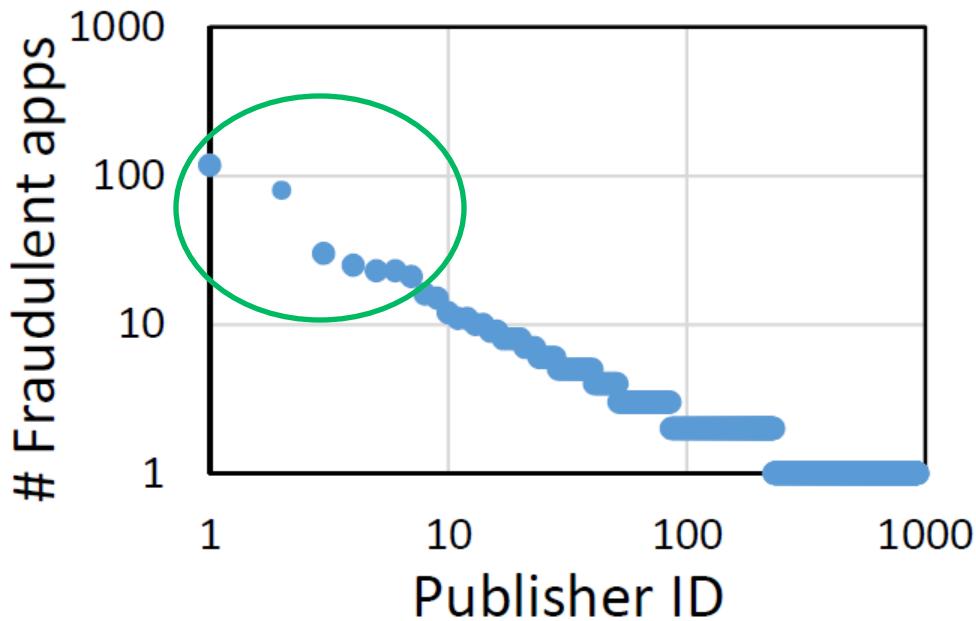
Fraud level does not seem to depend on rating



Fraudulent App Count per Publisher

Each app is developed by a publisher

The distribution of the number of fraud across publishers who commit fraud exhibits a heavy tail



A small number of publishers are responsible for most of the fraud

Conclusion

Mobile ad fraud is a 1 billion dollar business, and ad networks need effective tools to detect fraud

Conclusion

DECAF: a system for detecting placement ad fraud in mobile apps



Efficiently explore structurally different pages of mobile apps

Accurately detect placement ad fraud in a fast and scalable way

Case study of 51,150 apps reveals interesting variability in the prevalence of fraud by type, rating, publisher and etc.

