Analyzing Web Logs to Detect User-Visible Failures

Wanchun Li  Georgia Institute of Technology
Ian Gorton  Pacific Northwest National Laboratory
Road Map

I. Introduction
II. Technique
III. Model Training
IV. Evaluation
V. Discussion
VI. Conclusion
INTRODUCTION

• Web applications suffer from poor reliability
  ▪ Top 40 Web sites about 10 days of downtime per year
  ▪ 32% of shoppers experienced online shopping problems during the 2006 holiday season
  ▪ 89% of all online customers experienced errors

Practitioners rely on fast failure detection and recovery to reduce the effects of failures on other users.
INTRODUCTION

• Early failure detection can mitigate about 65% of failures

• Failure detection is challenging
  ▪ Requires up to 75% of failure recovery time

• User feedback has limited help for detecting failures
  ▪ User survey of www.clinicalguard.com in 2008
    • 200 users
    • 9 responses
    • 1 specified the failure
Existing Detection Techniques

- Resource usages analysis
  - Constructing statistics using data of resources usage
    - Focusing on performance failures
    - Not on failures related to software bugs

- Runtime components interaction analysis
  - Detecting runtime execution path anomalies
  - Not always effective to software bugs

- User-behavior-based analysis
  - Analyzing request bursts to a URL/resource
    - Assume users refreshing browsers for failures
    - Users have different behavior than refreshing
I. Introduction

II. **Technique**

III. Model Training

IV. Evaluation

V. Discussion

VI. Conclusion
Overview

The Goal: Detecting failures caused by software bugs

Assumptions

HCI Rational Principle
Users must respond if the result of a sequence of interactions is not satisfactory

Navigation Patterns
- Web users follow certain navigation patterns
- Users’ response to failures may break these patterns

The Idea: Detecting anomalous navigation paths as indications that users encountered failures
The Model

- A directed graph representing a Web site
  - Nodes are Web pages
  - Edges are users’ navigation

\[ S = \{A, B, C, C, D, A, D\} \]

- A Markov model in the 1\textsuperscript{st} order for estimating the probability of a navigation path
  - The transition probability to the next state is conditionally dependent on only the current state

\[
\]
Two types of transition probability

- **Outgoing Transition Probability (OTP)**
  The probability that users go from page A to page B

- **Incoming Transition Probability (ITP)**
  The probability that users at page B coming from page A

OTP usually is different from ITP

- A user can navigate to the Home page from any page
- But not vice versa
Occurrence Probability for Failure Detection

- Given a sequence of user requests
  - Compute the occurrence probability
  - Using 1\textsuperscript{st}-order Markov model

- Outgoing Occurrence Probability (OOP)
  The occurrence probability computed using OTP

- Incoming Occurrence Probability (IOP)
  The occurrence probability computed using ITP

If \( \min (\text{OOP}, \text{IOP}) < \text{threshold} \)
Raise a failure alarm
Road Map

I. Introduction

II. Technique

III. Model Training

IV. Evaluation

V. Discussion

VI. Conclusion
Bayesian Learning

• Assume
  ▪ The parameter to estimate is a random variable
• Estimate
  ▪ The distribution of the parameter as a random variable
  ▪ A statistic as the estimator
• Process
  ▪ Assume a distribution of the parameter
  ▪ Find a *conjugate prior distribution*
  ▪ Compute the *posterior distribution*
    • Update the prior distribution using the training data
  ▪ Decide an estimator
    • *posterior mean*: the mean of *the posterior distribution*
Bayesian Learning to train a First-order Markov Model
  - A Multinomial distribution
  - A Dirichlet distribution as the conjugate prior

Learn Outgoing/Incoming Transition Probability

The learning process
  - A small amount of training data for setting prior
  - The rest training data for updating prior
  - The posterior mean as the estimator
Estimated Transition Probability

\[ \hat{\theta}_{i,j} = \frac{n_{i,j} + \alpha q_j}{n_i + \alpha} \]

- \( \hat{\theta}_{i,j} \): Estimated OTP from state \( i \) to state \( j \)
- \( n_i \): All hits on state \( i \) in data for setting the prior
- \( n_{i,j} \): Transitions from \( i \) to \( j \) in data for setting the prior
- \( \alpha \): All hits on state \( i \) in the rest training data
- \( q_j \): Transition frequency from \( i \) to \( j \) in the rest training data
I. Introduction

II. Technique

III. Model Training

**IV. Evaluation**

V. Discussion

VI. Conclusion
• NASA Web site

• Construct user-sessions using one month access log
  ▪ 1,891,714 HTTP requests from real users

• Training data

  \[ \hat{\theta}_{ij} = \frac{n_{ij} + \alpha q_j}{n_i + \alpha} \]

  ▪ \( n_i n_{ij} \) Prior: 572 user-sessions on 1\(^{st}\) day
  ▪ \( \alpha q_j \) Learning: 2404 user-sessions on 2\(^{nd}\) to 10\(^{th}\) day

• Testing data
  ▪ 7941 non-error sessions for detection
  ▪ 500 error sessions for false positive
Equal Error Rate (i.e., EER): the decision boundary when detection and false-positive have the same loss function. Our model’s EER=0.71/0.26
I. Introduction

II. Technique

III. Model Training

IV. Evaluation

V. Discussion

VI. Conclusion
Discussion

- Improving the detection power
  - Semi-Markov model (e.g., time)
  - Hidden state

- The “ground truth”
  - Error sessions as user-visible failures

- More case studies
  - Controlled environments
    - Recruit users
    - Instrument real-world Web sites
Road Map

I. Introduction

II. Technique

III. Model Training

IV. Evaluation

V. Discussion

VI. Conclusion
Conclusion

• Detecting User-visible failures
  ▪ Improving both reliability and user’s satisfaction

• User’s behavior changes when encounter failures
  ▪ Breaking navigation patterns

• Our technique detects anomaly user navigation paths

• The experiment results demonstrate our technique can detect failures with reasonable cost

• Future work aims at model improvements and case studies
Thank You!