ICPE 2018
Berlin
April 12, 2018

AI Techniques in Software Engineering Paradigm

Rung-Tsong Michael LYU

Computer Science & Engineering Department
The Chinese University of Hong Kong
A Brief History of the IT World

- **Industrial Revolution**
  - ENIAC
  - The MITS Altair
  - Apple II

- **Information Age**
  - Birth of Internet
  - IBM Desktop PC
  - Apple Macintosh

- **Internet Age**
  - Birth of WWW
  - Birth of XML
  - Birth of Web 2.0
  - Birth of iPhone

- **WWW Age**
  - Time Magazine Person of the Year

- **Attention Age**
  - Time Magazine Person of the Year

Timeline:
- 1750
- 1945
- 1969
- 1975
- 1981
- 1983
- 1984
- 1989
- 1996
- 2004
- 2006
- 2007
# A Brief History of AI Development

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
<th>Year</th>
<th>Event</th>
<th>Year</th>
<th>Event</th>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1965</td>
<td>Basic Structure</td>
<td>1980s</td>
<td>Neural Network</td>
<td>2010</td>
<td>IBM Watson wins on Jeopardy</td>
<td>2011</td>
<td>Alpha Go beats Lee Sedol</td>
</tr>
<tr>
<td>1990-2000</td>
<td>Breakthrough</td>
<td>2016</td>
<td>Now</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**The Age of AI**

**What is “Intelligence”?**

**Symbolics Machine**

**IBM’s Watson**
What is Artificial Intelligence (AI)

Human Intelligence
- learning
- reasoning
- feeling
- perceiving
- understanding

Artificial Intelligence

Artificial Intelligence is the science and engineering of making intelligent machines

Turing Test (1950)

Boston Dynamics: Atlas
What Impact Has AlphaGo Achieved?

- Search space is huge: $\approx 10^{360}$
Reborn of Artificial Intelligence

Face recognition

Speech recognition

Natural language processing

Intelligent systems (e.g., self-driving)

AI will be everywhere
Software Engineering with Artificial Intelligence:

Employing Machine Learning (ML) techniques to assist in labor-intensive and error-prone tasks.
Software Engineering with Intelligence

Development

Operation

Analysis

Apply ML Techniques

Source codes
Logging statements

User behavior data
User reviews

QoS values
System logs

Source codes
Logging statements

User behavior data
User reviews

QoS values
System logs
Introduction

AI Techniques

Development Phase

Operational Phase

Analysis Phase

Conclusion
Artificial Intelligence for Software Engineering
Artificial Intelligence for Software Engineering

Tasks

Development

Code completion
Learning to log

Operational

App issues prioritizing
Emerging issues detection

Analysis

Service reliability prediction
Log Management

Techniques

Machine Learning

RNN with Attention
Classification

Classification,
Topic modeling,

Matrix factorization,
Classification,
Parallel computing platform
Machine Learning Framework

- **General framework:**
  \[
  \arg \min_{\hat{\theta}} \Gamma(\{x_i, y_i\}_{i=1}^{N}; \hat{\theta}) + \Psi(\hat{\theta})
  \]

- **Iterative Update:**
  \[
  \hat{\theta}^{t+1} = \hat{\theta}^t + \Delta_{f}\theta(D)
  \]

- Unchanged Data
- Frequently updated Parameter
- Very Big: cannot be handled with single PC
- Gradient Descent: Computed in Distributed Environment
Matrix Factorization

\[ R \approx U^T V \]

\[
R = \begin{bmatrix}
5 & 2 & 3 & 4 \\
4 & 3 & 5 & \\
4 & 2 & 2 & 4 \\
5 & 1 & 2 & 4 & 3 \\
4 & 3 & 2 & 4 & 3 & 5
\end{bmatrix}
\]

\[
U = \begin{bmatrix}
1.55 & 1.22 & 0.37 & 0.81 & 0.62 & -0.01 \\
0.36 & 0.91 & 1.21 & 0.39 & 1.10 & 0.25 \\
0.59 & 0.20 & 0.14 & 0.83 & 0.27 & 1.51 \\
0.39 & 1.33 & -0.43 & 0.70 & -0.90 & 0.68 \\
1.05 & 0.11 & 0.17 & 1.18 & 1.81 & 0.40
\end{bmatrix}
\]

\[
V = \begin{bmatrix}
1.00 & -0.05 & -0.24 & 0.26 & 1.28 & 0.54 & -0.31 & 0.52 \\
0.19 & -0.86 & -0.72 & 0.05 & 0.68 & 0.02 & -0.61 & 0.70 \\
0.49 & 0.09 & -0.05 & -0.62 & 0.12 & 0.08 & 0.02 & 1.60 \\
-0.40 & 0.70 & 0.27 & -0.27 & 0.99 & 0.44 & 0.39 & 0.74 \\
1.49 & -1.00 & 0.06 & 0.05 & 0.23 & 0.01 & -0.36 & 0.80
\end{bmatrix}
\]
Low-Rank Matrix Factorization

- **Objective function**

\[
\min_{U,V} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij} (R_{ij} - U_i^T V_j)^2 + \frac{\lambda_1}{2} \|U\|_F^2 + \frac{\lambda_2}{2} \|V\|_F^2
\]

- **Main Objective**
- **Regularization terms**

\[I_{ij}\] is the indicator function that is equal to 1 if user \(u_i\) rated item \(v_j\) and equal to 0 otherwise.

\(U_i, V_j\): low dimension column vectors to represent user/item preferences.
The Growing of Deep Learning

Growing Use of Deep Learning at Google

- Unique project directories
- # of directories containing model description files

Across many products/areas:
- Android
- Apps
- drug discovery
- Gmail
- Image understanding
- Maps
- Natural language understanding
- Photos
- Robotics research
- Speech
- Translation
- YouTube
- ... many others ...

Deep learning trends at Google. Source: SIGMOD/Jeff Dean
Deep Learning Is Neural Networks

Accuracy

Scale (data size, model size)

1980s and 1990s

more compute

neural networks

other ML approaches
Deep Learning Is Neural Networks

Now

Accuracy

Scale (data size, model size)

more compute

neural networks

other ML approaches
Deep Learning

Feedforward Neural Networks (FFN)

\[ y = w \cdot x + b \]
Deep Learning

Convolutional Neural Networks (CNN)

layer m+1

layer m

layer m-1

Local Filter

Image

Convolved Feature
What is Deep Learning?

- A powerful class of machine learning model
- Modern reincarnation of artificial neural networks
- Collection of simple, trainable mathematical functions
- Compatible with many variants of machine learning

“cat”

Diagram showing the process of deep learning with layers transforming input data into the output "cat".
What is Deep Learning?

- Loosely based on (what little) we know about the brain
How Do Neural Networks Work?
How Do Neural Networks Work?

Anything humans can do in 0.1 sec, the right big 10-layer network can do too

0.1 sec: neurons fire only 10 times!
Combining Vision with Robotics


“Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection”, Sergey Levine, Peter Pastor, Alex Krizhevsky, & Deirdre Quillen, Arxiv, arxiv.org/abs/1603.02199
What Can Neural Networks Compute?

Human perception is very fast (0.1 second)

- Recognize objects (“see”) 🌟
- Recognize speech (“hear”) 🦊
- Recognize emotion 😊
- Instantly see how to solve some problems 🌟
- And many more!

These can all be computed by Neural Networks.
Deep Learning

Recurrent Neural Networks (RNN)

A standard RNN

An LSTM network
Deep Learning

Sequence-to-Sequence Models

ENCODER

Are you free tomorrow?

DECODER

thought vector

Reply

Yes, what’s up? <END>

Incoming Email

<START>
Deep Learning Platforms

- TensorFlow
- PyTorch
- CNTK
- K
- Caffe2
- Theano
- dmlc
- mxnet
- Chainer
- Caffe
Introduction

AI Techniques

Development Phase

Operational Phase

Analysis Phase

Conclusion

Code completion

Learning to log
Development: Code Completion

- Code completion

```java
Aliases aliases = template.getClass().getAnnotation(Aliases.class);
if (aliases != null) {
    aliases.value().
    for (String reg)
    } else {
        for (int i = expr.length-1; i >= 0; i--)
        } findOne
    instanceof expr instanceof SomeType ? (SomeType) expr). : null
    var
    notnull
    par
    for (int i = 0; i < expr.length; i++)
    null
    field
    return
    for
    synchronized
```
Development: Code Completion

- Abstract syntax tree (AST)
- Locally repeated terms

A Python program and its corresponding abstract syntax tree
Development: Code Completion

- **Pointer mixture network**
  - Global RNN component
  - Local pointer component
  - Controller

- **Contributions:**
  - Pointer mixture network for better predicting OoV words
  - Effectiveness of attention mechanism
  - Significant improvements in code completion task
Development: Code Completion

• Case study

```python
class Operator(Employee):
    def __init__(self, name, employee_id):
        super(Operator, self).__init__(name, Rank.OPERATOR)
        self.employee_id = employee_id

def _dispatch_call(self, call):
    for employee in employees:
        employee.take_call(call)

def record_path(self, base_name):
    return os.path.join(base_name, str(self.__?__))
```

- Pointer Mixture Network successfully point to `employee_id`, which is an OoV word
Development: Code Completion

- Dataset
  - JavaScript (JS) and Python (PY)

<table>
<thead>
<tr>
<th></th>
<th>JS</th>
<th>PY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Queries</td>
<td>$10.7 \times 10^7$</td>
<td>$6.2 \times 10^7$</td>
</tr>
<tr>
<td>Test Queries</td>
<td>$5.3 \times 10^7$</td>
<td>$3.0 \times 10^7$</td>
</tr>
<tr>
<td>Type Vocabulary</td>
<td>95</td>
<td>329</td>
</tr>
<tr>
<td>Value Vocabulary</td>
<td>$2.6 \times 10^6$</td>
<td>$3.4 \times 10^6$</td>
</tr>
</tbody>
</table>

- Accuracies on next value prediction with different vocabulary sizes

<table>
<thead>
<tr>
<th>Vocabulary Size (OoV Rate)</th>
<th>JS</th>
<th>PY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1k (20%)</td>
<td>10k (11%)</td>
</tr>
<tr>
<td>Vanilla LSTM</td>
<td>69.9%</td>
<td>75.8%</td>
</tr>
<tr>
<td>Attention-enhanced LSTM</td>
<td>71.7%</td>
<td>78.1%</td>
</tr>
<tr>
<td>Pointer Mixture Network</td>
<td>73.2%</td>
<td>78.9%</td>
</tr>
<tr>
<td></td>
<td>1k (24%)</td>
<td>10k (16%)</td>
</tr>
<tr>
<td>Vanilla LSTM</td>
<td>63.6%</td>
<td>66.3%</td>
</tr>
<tr>
<td>Attention-enhanced LSTM</td>
<td>64.9%</td>
<td>68.4%</td>
</tr>
<tr>
<td>Pointer Mixture Network</td>
<td>66.4%</td>
<td>68.9%</td>
</tr>
<tr>
<td></td>
<td>10k (16%)</td>
<td>50k (11%)</td>
</tr>
<tr>
<td>Vanilla LSTM</td>
<td>67.3%</td>
<td>69.8%</td>
</tr>
<tr>
<td>Attention-enhanced LSTM</td>
<td>68.4%</td>
<td>70.1%</td>
</tr>
<tr>
<td>Pointer Mixture Network</td>
<td>68.9%</td>
<td>70.1%</td>
</tr>
</tbody>
</table>
Development: Code Completion

• Comparisons against the state-of-the-arts
  – Note that Pointer Mixture Network can be only used for predicting VALUE node (TYPE node has small size of vocabulary)

<table>
<thead>
<tr>
<th></th>
<th>JS TYPE</th>
<th>JS VALUE</th>
<th>PY TYPE</th>
<th>PY VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla LSTM</td>
<td>87.1%</td>
<td>78.6%</td>
<td>79.3%</td>
<td>67.3%</td>
</tr>
<tr>
<td>Attention-enhanced LSTM (ours)</td>
<td>88.6%</td>
<td>80.6%</td>
<td>80.6%</td>
<td>69.8%</td>
</tr>
<tr>
<td>Pointer Mixture Network (ours)</td>
<td>-</td>
<td>81.0%</td>
<td>-</td>
<td>70.1%</td>
</tr>
<tr>
<td>LSTM (Liu et al. 2016)</td>
<td>84.8%</td>
<td>76.6%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Probabilistic Model (Raychev et al. 2016)</td>
<td>83.9%</td>
<td>82.9%</td>
<td>76.3%</td>
<td>69.2%</td>
</tr>
</tbody>
</table>

• Observations: our models outperform the state-of-the-art in almost all cases
Development: Learning to Log

• Challenges of logging
  – Logging too little
    • Miss valuable runtime information
    • Increase the difficulty for problem diagnosis
  – Logging too much
    • Additional cost of code dev. & maintenance
    • Runtime overhead
    • Producing a lot of trivial logs
    • Storage overhead

User:
“Apache httpd cannot start.
No log message printed.”
Development: Learning to Log

- What is logging?
  
  ```
  Log (level, "logging message %s", variable);
  ```

  - A common programming practice to record runtime system information
  - Logging functions: e.g., printf, cout, writeline, etc.

- Logs are crucial for system management
  
  - Various tasks of log analysis
    - Anomaly detection, failure diagnosis, etc.
  - The only data available for diagnosing production failures

Logging is important!
Development: Learning to Log

- Focused snippets: potential error sites
  - Exception snippets: try-catch blocks
  - Return-value-check snippets: function-return errors

Example 1
```java
try {
    method(...);
}
catch (IOException) {
    log(...);
    ...
}
```

Example 2
```javascript
var res = method(...);
if (res == null) {
    log(...);
    ...
}
```
Development: Learning to Log

- Framework of learning to log
  - Similar to other machine learning applications (e.g., defect prediction)
Development: Learning to Log

• Structural features: structural info of code

```csharp
private int LoadRulesFromAssembly (string assembly, ...){
    //Code in Setting
    try {
        AssemblyName aname = AssemblyName.
        GetAssemblyName(Path.GetFullPath (assembly));
        Assembly a = Assembly.Load (aname);
    } catch (FileNotFoundException) {
        Console.Error.WriteLine ("Could not load rules
        From assembly '{0}'.", assembly); return 0; }
    ... }
```

Exception Type:
0.39 (System.IO.FileNotFoundException)

Containing method:
Gendarme.Settings.LoadRulesFromAssembly

Invoked methods:
System.IO.Path.GetFullPath,
System.Reflection.AssemblyName.GetAssemblyName,
System.Reflection.Assembly.Load

/* A code example taken from MonoDevelop (v.4.3.3), at file: *
* main\external\mono-tools\gendarme\console\Settings.cs,
* line: 116. Some lines are omitted for ease of presentation. */
Development: Learning to Log

- Within-project evaluation
  - Random: randomly logging (as a new developer)
  - ErrLog [Yuan et al., OSDI’12]: conservatively logging all focused snippets
  - LogAdvisor: 0.846 ~ 0.934 accuracy achieved
Introduction

AI Techniques

Development Phase

Operational Phase
  - App issues prioritizing
  - Emerging issues detection

Analysis Phase

Conclusion
Operation: App Issues Prioritizing

- User reviews are valuable source for pinpointing emerging issues for app development.
- Capturing user-concerned issues and tracking their trends

2,800,000 apps  2,200,000 apps  669,000 apps

Changelog

What's New

5.9.8

1. Optimized Photo Manager - Easier than ever to manage photos and save space.
2. Added anti-intruder feature in AppLock to protect your privacy - Someone who tries to break into your phone will have their picture snapped.
3. Various other improvements.

Getting stupid Not sure what you’ve done but now I get a folder called commandir and it wants to clean it and says its safe to delete, its 12gb and that 12 gb is over half my internal storage so that must be my apps and its saying safe to delete?

Love it Great app. But only one problem. I had Some problem with my screen and did a wrong pattern and it has taken my photos. Plz let me know how do I delete those "intruder selfie’s".
Operation: App Issues Prioritizing

Framework of PAID

A. Data Extraction

① Data Crawling  →  ② Preprocessing  →  ③ Filtering

B. App Issue Generation


C. Visualization and Issue Analysis

⑦ Visualization  →  ⑧ Issue Analysis
Operation: App Issues Prioritizing

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Review</th>
<th>Date</th>
<th>Stars</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Crash</td>
<td>Like it cause it doesn’t crash on androids</td>
<td>2014-11-09T08:55:47</td>
<td>5</td>
<td>15.0.0.15.13</td>
</tr>
<tr>
<td>2</td>
<td>Rubbish</td>
<td>When I try to connect with Mobile Network Package, this don't work and giving &quot;Connecting.. Problem&quot;.</td>
<td>2014-11-12T18:32:25</td>
<td>1</td>
<td>15.0.0.15.10</td>
</tr>
</tbody>
</table>
Based on topic modeling, each topic is labeled with one phrase.

**Topic Labeling Process:**

Rank phrases for each topic by:

- **Semantic aspect:**
  KL-Divergence
  \[
  Sem(\beta_i, l) = Sim(\beta_i, l) - \frac{\mu}{k-1} \sum_{j \neq i} Sim(\beta_j, l)
  \]

- **Sentiment aspect:**
  \[
  Sen(l) = e^{\mu(l/h)}
  \]

- **Total score:**
  \[
  S(\beta_i, l) = Sem(\beta_i, l) * Sen(l)
  \]
Operation: App Issues Prioritizing

The Themeriver of Viber.
### Operation: App Issues Prioritizing

#### Rank top reviews for each topic:

The Top Three Reviews Related to “**Activation Code**”

<table>
<thead>
<tr>
<th>Rank</th>
<th>User Review</th>
<th>Importance Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Upload viber!</strong> I went. Enter a phone number. I enter. Asks for sure your phone? It will be sent an activation code. Ok. <strong>Messages are not present</strong>. He writes to activate viber here, install it to your phone first. But I have it pumped? What to do? Help!</td>
<td>0.836</td>
</tr>
<tr>
<td>2</td>
<td>I hard reset my tab 3. Installed viber for activation code when i write my phone number and press okay a <strong>white popup written only</strong>. ERROR <strong>no description given and an okay button on it please help me viber</strong>s my only way to contact my son abroad.</td>
<td>0.834</td>
</tr>
<tr>
<td>3</td>
<td>I don’t know what’s wrong with Viber. Just downloaded it and it <strong>keeps on saying activation code sent to your device.</strong> For almost a month, no any activation code and it’s really pissing me off. Pls fix.</td>
<td>0.828</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Operation: Emerging Issues Detection

Number of Privacy Related Reviews in Facebook during July 2013 to April 2015

Facebook Messenger users gripe and grumble in online reviews

Around 92 percent of more than 64,000 Facebook users have given the Messenger app a one-star rating on App Annie over the past month.
IDEA

**Identifying Emerging Issues from App Reviews**

- Automatic tool for app review analysis
- Discovering emerging issues dynamically
- Comprehensive issue interpretation
- Visualizing issue progression over versions
Overall Framework

A. Preprocessing

B. Emerging Topic Detection

C. Topic Interpretation

D. Visualization

Emerging Issues

Version $t$

Review Stream

AOLDA

Anomaly Discovery

Candidate Extraction

Topic Labeling
Overview of AOLDA (Adaptively Online Latent Dirichlet Allocation). The red rectangle with dashed dots highlights the adaptive integration of the topics of the w previous versions.
Emerging Issue Detection

Anomaly Detection - Jensen-Shannon Divergence

\[ D_{JS}(\phi^t_k || \phi^{t-1}_k) = \frac{1}{2} D_{KL}(\phi^t_k || M) + \frac{1}{2} D_{KL}(\phi^{t-1}_k || M) \]

\[ D_{KL}(P || Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}. \]
## Experimental Result

<table>
<thead>
<tr>
<th>App Name (#avg. reviews)</th>
<th>Method</th>
<th>Phrase</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Precision$_E$</td>
<td>Recall$_L$</td>
</tr>
<tr>
<td>NOAA Radar (523)</td>
<td>OLDA</td>
<td>0.468</td>
<td>0.528</td>
</tr>
<tr>
<td></td>
<td>IDEA-R</td>
<td>0.606</td>
<td>0.461</td>
</tr>
<tr>
<td></td>
<td>IDEA-S</td>
<td>0.250</td>
<td>0.530</td>
</tr>
<tr>
<td></td>
<td>IDEA$^+$</td>
<td>0.571</td>
<td>0.497</td>
</tr>
<tr>
<td>Youtube (1,143)</td>
<td>OLDA</td>
<td>0.441</td>
<td>0.462</td>
</tr>
<tr>
<td></td>
<td>IDEA-R</td>
<td>0.506</td>
<td>0.429</td>
</tr>
<tr>
<td></td>
<td>IDEA-S</td>
<td>0.548</td>
<td>0.466</td>
</tr>
<tr>
<td></td>
<td>IDEA$^+$</td>
<td>0.592</td>
<td>0.472</td>
</tr>
<tr>
<td>Viber (2,141)</td>
<td>OLDA</td>
<td>0.157</td>
<td>0.305</td>
</tr>
<tr>
<td></td>
<td>IDEA-R</td>
<td>0.542</td>
<td>0.326</td>
</tr>
<tr>
<td></td>
<td>IDEA-S</td>
<td>0.500</td>
<td>0.342</td>
</tr>
<tr>
<td></td>
<td>IDEA$^+$</td>
<td>0.625</td>
<td>0.340</td>
</tr>
<tr>
<td>Clean Master (6,332)</td>
<td>OLDA</td>
<td>0.300</td>
<td>0.269</td>
</tr>
<tr>
<td></td>
<td>IDEA-R</td>
<td>0.500</td>
<td>0.216</td>
</tr>
<tr>
<td></td>
<td>IDEA-S</td>
<td>0.067</td>
<td>0.289</td>
</tr>
<tr>
<td></td>
<td>IDEA$^+$</td>
<td>0.667</td>
<td>0.318</td>
</tr>
<tr>
<td>Ebay (3,943)</td>
<td>OLDA</td>
<td>0.167</td>
<td>0.238</td>
</tr>
<tr>
<td></td>
<td>IDEA-R</td>
<td>0.229</td>
<td>0.243</td>
</tr>
<tr>
<td></td>
<td>IDEA-S</td>
<td>0.125</td>
<td>0.285</td>
</tr>
<tr>
<td></td>
<td>IDEA$^+$</td>
<td>0.229</td>
<td>0.251</td>
</tr>
<tr>
<td>SwiftKey (1,313)</td>
<td>OLDA</td>
<td>0.100</td>
<td>0.567</td>
</tr>
<tr>
<td></td>
<td>IDEA-R</td>
<td>0.333</td>
<td>0.611</td>
</tr>
<tr>
<td></td>
<td>IDEA-S</td>
<td>0.333</td>
<td>0.622</td>
</tr>
<tr>
<td></td>
<td>IDEA$^+$</td>
<td>0.517</td>
<td>0.653</td>
</tr>
</tbody>
</table>

[https://remine-lab.github.io/](https://remine-lab.github.io/)
Software Reliability Prediction: Small Data Modeling

**Reliability** \( R(t) = e^{- \int_0^t \lambda(x)dx} \)

- **Present**
- **Objective**
- **Execution Time** \( t \)
- **Total Failures**

**Failure Intensity** \( \lambda(t) \)

**Additional Time**
Analysis: Service Reliability Prediction

- Approach 1: Neighborhood-based

Reliability is extended to Quality-of-Service (QoS)

- Key idea: Using past usage experiences of similar users.
- Issue: How to calculate user similarity?

Unreliable Web service in US

Service user 1 in Asia

Reliable Web service in US

Service user 2 in US
Analysis: Service Reliability Prediction

- **Similarity Computation**

- **User-item matrix:** $\mathbf{M} \times \mathbf{N}$, each entry is the failure probability of a Web service

- **Pearson Correlation Coefficient (PCC)**

  $$\text{Sim}(a, u) = \frac{\sum_{i \in I_a \cap I_u} (p_{a,i} - \bar{p}_a)(p_{u,i} - \bar{p}_u)}{\sqrt{\sum_{i \in I_a \cap I_u} (p_{a,i} - \bar{p}_a)^2} \sqrt{\sum_{i \in I_a \cap I_u} (p_{u,i} - \bar{p}_u)^2}}$$

\[\begin{array}{cccccc}
  & \text{ws}_1 & \text{ws}_2 & \text{ws}_3 & \text{ws}_4 & \text{ws}_5 & \text{ws}_6 \\
\hline
\text{u}_1 & 0.1 & 0.1 & 0.2 & \textcolor{red}{0.5} & 0.3 \\
\text{u}_2 & 0.1 & 0.1 & 0.2 & 0.5 & 0.3 \\
\text{u}_3 & 0.4 & 0.3 & 0.2 & 0.3 & 0.4 \\
\text{u}_4 & 0.5 & 0.3 & 0.4 & 0.3 & 0.3 \\
\end{array}\]
Analysis: Service Reliability Prediction

WSRec: Hybrid Prediction Approach

- Similar users + Similar Web services

\[ p_{u,i} = \lambda \times \left( \overline{p_u} + \sum_{a \in S(u)} w_a \times (p_{a,i} - \overline{p_a}) \right) + (1 - \lambda) \times \left( \overline{p_i} + \sum_{k \in S(i)} w_k \times (p_{u,k} - \overline{p_k}) \right) \]
### Analysis: Service Reliability Prediction

#### Performance Comparison

<table>
<thead>
<tr>
<th>Metric</th>
<th>Density</th>
<th>Methods</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>UMEAN</td>
<td>1623</td>
<td>3539</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMEAN</td>
<td>903</td>
<td>901</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPCC</td>
<td>1148</td>
<td>877</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPCC</td>
<td>758</td>
<td>736</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSRec</td>
<td>758</td>
<td>700</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>UMEAN</td>
<td>1883</td>
<td>1848</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMEAN</td>
<td>868</td>
<td>859</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPCC</td>
<td>904</td>
<td>722</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPCC</td>
<td>606</td>
<td>610</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSRec</td>
<td>566</td>
<td>551</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>UMEAN</td>
<td>1603</td>
<td>1543</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMEAN</td>
<td>856</td>
<td>854</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPCC</td>
<td>915</td>
<td>671</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPCC</td>
<td>563</td>
<td>566</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSRec</td>
<td>564</td>
<td>504</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UMEAN</td>
<td>3339</td>
<td>3250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMEAN</td>
<td>1441</td>
<td>1436</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPCC</td>
<td>2036</td>
<td>1455</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPCC</td>
<td>1259</td>
<td>1288</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSRec</td>
<td>1329</td>
<td>1247</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>UMEAN</td>
<td>3332</td>
<td>3240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMEAN</td>
<td>1269</td>
<td>1252</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPCC</td>
<td>1356</td>
<td>1128</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPCC</td>
<td>1020</td>
<td>1116</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSRec</td>
<td>997</td>
<td>946</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>UMEAN</td>
<td>3336</td>
<td>3246</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IMEAN</td>
<td>1207</td>
<td>1209</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPCC</td>
<td>1267</td>
<td>1035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IPCC</td>
<td>950</td>
<td>987</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WSRec</td>
<td>921</td>
<td>884</td>
</tr>
</tbody>
</table>
Analysis: Service Reliability Prediction

- Drawbacks of Neighborhood-based Approach

  - Computational complexity $O(mn + n^2)$
  - Matrix sparsity problem
    - Not easy to find similar users (or similar items)

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>$i_3$</th>
<th>$i_4$</th>
<th>$i_5$</th>
<th>$i_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$u_2$</td>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9</td>
</tr>
<tr>
<td>$u_4$</td>
<td></td>
<td></td>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_5$</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analysis: Service Reliability Prediction

- **Approach 2: Model-based Approach**

  - Each row of $U^T$ is a set of feature factors, and each column of $V$ is a set of linear predictors $\Rightarrow$ **Matrix Factorization (MF)**

\[
\begin{align*}
U & = \begin{bmatrix}
0.98 & 0.23 & 0.22 \\
0.13 & 0.27 & 0.25 \\
0.37 & 0.36 & \\
0.69 & 0.22 & 0.34 \\
\end{bmatrix} \\
V & = \begin{bmatrix}
0.73 & 0.35 & 0.31 & 0.26 & 0.32 & 0.42 \\
0.60 & 0.31 & 0.27 & 0.22 & 0.28 & 0.36 \\
0.69 & 0.37 & 0.32 & 0.27 & 0.33 & 0.45 \\
0.95 & 0.46 & 0.42 & 0.35 & 0.41 & 0.54 \\
\end{bmatrix}
\end{align*}
\]

The error between the actual Value and the prediction:

\[
\begin{align*}
\min_{U,V} L(R, U, V) &= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} R_{ij}^2 (R_{ij} - U_i^T V_j)^2 \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} (R_{ij} - U_i^T V_j)^2 \\
&+ \frac{\lambda_U}{2} \| U \|_F^2 + \frac{\lambda_V}{2} \| V \|_F^2,
\end{align*}
\]

Regularization terms
Analysis: Service Reliability Prediction

NIMF: Neighborhood–Integrated Matrix Factorization

\[ \mathcal{L}(R, S, U, V) \]

\[ = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^R (R_{ij} - (\alpha U_i^T V_j + (1 - \alpha) \sum_{k \in T(i)} S_{ik} U_k^T V_j))^2 \]

\[ + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2, \]

\[ S_{ik} = \frac{PCC(i, k)}{\sum_{k \in T(i)} PCC(i, k)} \]

(a) User-Item Matrix

<table>
<thead>
<tr>
<th></th>
<th>( i_1 )</th>
<th>( i_2 )</th>
<th>( i_3 )</th>
<th>( i_4 )</th>
<th>( i_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>0.5</td>
<td>1.2</td>
<td></td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>( u_2 )</td>
<td></td>
<td>0.8</td>
<td>0.6</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>( u_3 )</td>
<td>0.4</td>
<td></td>
<td>0.3</td>
<td></td>
<td>0.9</td>
</tr>
<tr>
<td>( u_4 )</td>
<td></td>
<td>0.6</td>
<td></td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>( u_5 )</td>
<td>0.5</td>
<td></td>
<td>0.7</td>
<td></td>
<td>0.3</td>
</tr>
</tbody>
</table>

User’s own rating  Rating due to similar users
# Analysis: Service Reliability Prediction

## Performance Comparison

### Table 2: Performance Comparison (A Smaller MAE or RMSE Value Means a Better Performance)

<table>
<thead>
<tr>
<th>QoS</th>
<th>Methods</th>
<th>Matrix Density=5%</th>
<th></th>
<th>Matrix Density=10%</th>
<th></th>
<th>Matrix Density=15%</th>
<th></th>
<th>Matrix Density=20%</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Response-time (0-20 s)</td>
<td>UMEAN</td>
<td>0.8785</td>
<td>1.8591</td>
<td>0.8783</td>
<td>1.8555</td>
<td>0.8768</td>
<td>1.8548</td>
<td>0.8747</td>
<td>1.8557</td>
</tr>
<tr>
<td></td>
<td>IMEAN</td>
<td>0.7015</td>
<td>1.5813</td>
<td>0.6918</td>
<td>1.5440</td>
<td>0.6867</td>
<td>1.5342</td>
<td>0.6818</td>
<td>1.5311</td>
</tr>
<tr>
<td></td>
<td>UPCC</td>
<td>0.6261</td>
<td>1.4078</td>
<td>0.5517</td>
<td>1.3151</td>
<td>0.5159</td>
<td>1.2680</td>
<td>0.4884</td>
<td>1.2334</td>
</tr>
<tr>
<td></td>
<td>IPCC</td>
<td>0.6897</td>
<td>1.4296</td>
<td>0.5917</td>
<td>1.3268</td>
<td>0.5037</td>
<td>1.2552</td>
<td>0.4459</td>
<td>1.2095</td>
</tr>
<tr>
<td></td>
<td>WSRec</td>
<td>0.6234</td>
<td>1.4078</td>
<td>0.5365</td>
<td>1.3043</td>
<td>0.4965</td>
<td>1.2467</td>
<td>0.4407</td>
<td>1.2012</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td>0.6182</td>
<td>1.5746</td>
<td>0.6040</td>
<td>1.5494</td>
<td>0.5990</td>
<td>1.5345</td>
<td>0.5982</td>
<td>1.5331</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>0.5678</td>
<td>1.4735</td>
<td>0.4996</td>
<td>1.2866</td>
<td>0.4720</td>
<td>1.2163</td>
<td>0.4492</td>
<td>1.1828</td>
</tr>
<tr>
<td></td>
<td><strong>NIMF</strong></td>
<td><strong>0.5514</strong></td>
<td><strong>1.4075</strong></td>
<td><strong>0.4854</strong></td>
<td><strong>1.2745</strong></td>
<td><strong>0.4534</strong></td>
<td><strong>1.1980</strong></td>
<td><strong>0.4357</strong></td>
<td><strong>1.1678</strong></td>
</tr>
<tr>
<td>Throughput (0-1000 kbps)</td>
<td>UMEAN</td>
<td>54.0084</td>
<td>110.2821</td>
<td>53.6700</td>
<td>110.2977</td>
<td>53.8792</td>
<td>110.1751</td>
<td>53.7114</td>
<td>110.1708</td>
</tr>
<tr>
<td></td>
<td>IMEAN</td>
<td>27.3558</td>
<td>66.6344</td>
<td>26.8318</td>
<td>64.7674</td>
<td>26.6239</td>
<td>64.3986</td>
<td>26.6364</td>
<td>64.1082</td>
</tr>
<tr>
<td></td>
<td>IPCC</td>
<td>29.2651</td>
<td>64.2285</td>
<td>27.3993</td>
<td>60.0825</td>
<td>26.4319</td>
<td>57.8593</td>
<td>25.0273</td>
<td>55.4970</td>
</tr>
<tr>
<td></td>
<td>WSRec</td>
<td>25.8755</td>
<td>60.8685</td>
<td>19.9754</td>
<td>54.8761</td>
<td>17.5543</td>
<td>47.8235</td>
<td>16.0762</td>
<td>47.8749</td>
</tr>
<tr>
<td></td>
<td>NMF</td>
<td>25.7529</td>
<td>65.8517</td>
<td>17.8411</td>
<td>53.9896</td>
<td>15.8939</td>
<td>51.7322</td>
<td>15.2516</td>
<td>48.6330</td>
</tr>
<tr>
<td></td>
<td>PMF</td>
<td>19.9034</td>
<td>54.0508</td>
<td>16.1755</td>
<td>46.4439</td>
<td>15.0956</td>
<td>43.7957</td>
<td>14.6694</td>
<td>42.4855</td>
</tr>
<tr>
<td></td>
<td><strong>NIMF</strong></td>
<td><strong>17.9297</strong></td>
<td><strong>51.6573</strong></td>
<td><strong>16.0542</strong></td>
<td><strong>45.9409</strong></td>
<td><strong>14.4363</strong></td>
<td><strong>43.1596</strong></td>
<td><strong>13.7099</strong></td>
<td><strong>41.1689</strong></td>
</tr>
</tbody>
</table>
Reliability Prediction of Web Services

- Approach 1: Neighborhood-based approach – to consider users
- Approach 2: Model-based approach – to consider data sparsity
- Approach 3: Time-aware approach – to consider temporal factor
- Approach 4: Network coordinate based approach – to consider spatial factor
- Approach 5: Ranking-based approach – to consider ranking
Analysis: Log Management

Log Analysis Framework
Analysis: Log Management

Raw Log Messages

1. 2008-11-11 03:40:58 BLOCK* NameSystem.allocateBlock: /user/root/randtxt4/_temporary/_task_200811101024_0010_m_000011_0/part-00011.blk 964791815409399662
4. 2008-11-11 03:41:48 PacketResponder 0 for block blk_904791815409399662 terminating
5. 2008-11-11 03:41:48 Received block blk_904791815409399662 of size 67108864 from /10.250.18.114
7. 2008-11-11 03:41:48 Received block blk_904791815409399662 of size 67108864 from /10.250.18.210
8. 2008-11-11 03:41:48 BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.18.114:50010 is added to blk_904791815409399662 size 67108864
10. 2008-11-11 08:30:54 Verification succeeded for blk_904791815409399662

Log Parsing

Log Events

- Event1: BLOCK* NameSystem.allocateBlock: *
- Event2: Receiving block * src: * dest: *
- Event3: PacketResponder * for block * terminating
- Event4: Received block * of size * from *
- Event5: BLOCK* NameSystem.addStoredBlock: blockMap updated: * is added to * size *
- Event6: Verification succeeded for *

Structured Logs

- 2008-11-11 03:40:58 Event1
- 2008-11-11 03:40:59 Event2
- 2008-11-11 03:41:01 Event2
- 2008-11-11 03:41:48 Event3
- 2008-11-11 03:41:48 Event4
- 2008-11-11 03:41:48 Event5
- 2008-11-11 03:41:48 Event5
- 2008-11-11 03:41:48 Event5
- 2008-11-11 08:30:54 Event6
We design and implement a parallel log parser (namely POP) on top of Spark.

It can process 200 million lines of raw log messages within 7 min while keeping high accuracy.
Analysis: Log Management

Existing anomaly detection methods: SVM (left) and PCA (right)
Our method: Deep Log Embedding based Anomaly Detection (D-Lead)
Analysis: Log Management

LogPAI
(Log Powered by AI)

loghub
A collection of system log datasets for massive log analysis

loghub
log-analysis logs console-log log-parsing unstructured-logs

⭐ 16 ⭐ 3 Updated 23 days ago

LogAdvisor
Learning to Log: A framework for determining optimal logging points

LogAdvisor
machine-learning logging code-analysis

● C# ⭐ 1 ⭐ 2 Updated on May 1

logparser
logparser: A toolkit for automated log parsing

logparser
log-analysis log log-parser log-mining log-parsing

● Python ⭐ 25 ⭐ 14 MIT Updated on Jul 27

loglizer
loglizer: A log analysis toolkit for automated anomaly detection

loglizer
log-analysis log-management anomaly-detection unstructured-logs

● Python ⭐ 33 ⭐ 17 MIT Updated on Sep 21

https://github.com/logpai
Defect Prediction

- Software defect prediction: build classifiers to predict code areas that potentially contain defects, based on code features.

- More effective feature extraction
  -- Deep Learning
Defect Prediction

- The overall workflow of proposed DP-CNN

1. Parsing source code and extracting token vectors
2. Encoding token vectors
3. Generating features via CNN
4. Predicting defects
## Defect Prediction

- **Performance on 8 open source projects**

<table>
<thead>
<tr>
<th>Project</th>
<th>Traditional</th>
<th>DBN</th>
<th>DBN+</th>
<th>CNN</th>
<th>DP-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>camel</td>
<td>0.329</td>
<td>0.335</td>
<td>0.375</td>
<td>0.505</td>
<td>0.508</td>
</tr>
<tr>
<td>jEdit</td>
<td>0.573</td>
<td>0.480</td>
<td>0.549</td>
<td>0.631</td>
<td>0.580</td>
</tr>
<tr>
<td>lucene</td>
<td>0.618</td>
<td>0.758</td>
<td>0.761</td>
<td>0.761</td>
<td>0.761</td>
</tr>
<tr>
<td>xalan</td>
<td>0.627</td>
<td>0.681</td>
<td>0.681</td>
<td>0.676</td>
<td>0.696</td>
</tr>
<tr>
<td>xerces</td>
<td>0.273</td>
<td>0.261</td>
<td>0.276</td>
<td>0.311</td>
<td>0.374</td>
</tr>
<tr>
<td>synapse</td>
<td>0.500</td>
<td>0.503</td>
<td>0.486</td>
<td>0.512</td>
<td>0.556</td>
</tr>
<tr>
<td>poi</td>
<td>0.748</td>
<td>0.780</td>
<td>0.782</td>
<td>0.778</td>
<td>0.784</td>
</tr>
<tr>
<td>eclipse</td>
<td>0.273</td>
<td>0.290</td>
<td>0.349</td>
<td>0.337</td>
<td>0.367</td>
</tr>
<tr>
<td>Average</td>
<td>0.493</td>
<td>0.511</td>
<td>0.532</td>
<td>0.564</td>
<td>0.578</td>
</tr>
</tbody>
</table>
Introduction

AI Techniques

Development Phase

Operational Phase

Analysis Phase

Conclusion
Conclusion

- Before AI becomes conscientious, its intelligence is still artificial.
- Software is eating the world, and AI is eating the software.

---Nvidia CEO Jensen Huang

- AI may replace many people’s job, but it will certainly enhance software engineers to do a better job.
- Our goal is to employ AI to provide more efficient and effective software development, operation, and analysis.
- The current achievement is just a small step ahead in a largely unexplored area in existing software engineering research paradigms.
Thank You!