A Semantic-aware Representation Framework for Online Log Analysis

Weibin Meng, Ying Liu, Yuheng Huang, Shenglin Zhang
Federico Zaiter, Bingjin Chen, Dan Pei
Outline

1 Background
2 Design
3 Evaluation
4 Summary
Background
Internet Services

Growing rapidly

Various types of services

Stability are important
Logs

- Monitoring data:
  - logs, traffic, PV.
- Logs are one of the most valuable data for service management

Logs record a vast range of runtime information (7*24)

Every service generates logs
Logs

- Logs are unstructured text
- designed by developers
- printed by logging statements (e.g., printf())

L₁. Interface ae3, changed state to down
L₂. Interface ae3, changed state to up
L₃. Interface ae1, changed status to down
L₄. Interface ae1, changed status to up
L₅. Vlan-interface vlan22, changed state to down
L₆. Vlan-interface vlan22, changed state to up

Logs are similar to nature language
Manual inspection of logs

- Manual inspection of logs is impossible
- A large-scale service is often implemented/maintained by hundreds of developers/operators.
- The volume of logs is growing rapidly.
- Traditional way: labor-intensive and time consuming

Automatic log analysis
Automatic log analysis

Automatic log analysis approaches, which are employed for services management, have been widely studied.
Log representation

- Most of automatic log analysis require **structured input**
  - Logs are unstructured text
- Log representation serves as **the first step** of automatic log analysis
  - Template index
  - Template count vector → **Lost semantic information**

( Semantic-aware log representation approach )
Challenges

1. Domain-specific semantic information
   - Logs contain logs of domain-specific words

2. Out-of-vocabulary (OOV) words
   - The vocabulary is growing continuously because the service can be upgraded to add new features and fix bugs
Original goal of logs: “logs are for users to read”

Logs are designed by developers and “printf”-ed by services

The intuition and methods in NLP can be applied for log representation

Log2Vec
Design
Overview of Log2Vec

1. Log-specific word embedding
2. Out-of-vocabulary word processor
3. Log vector generation

Open source toolkit: https://github.com/WeibinMeng/Log2Vec
Log-specific semantics

When embed words of logs, we should consider many information:

- Antonyms
- Synonyms
- Relation triples
- Others (future work)

Traditional word embedding methods (e.g., word2vec) assumes that words with a similar context tend to have a similar meaning fail to capture the log-specific meaning.
Prepare log-specific information

- Automatically extract
  - Antonyms & Synonyms
    - Search from WordNet\(^1\), a lexical database for English
  - Triples
  - Dependency tree\(^2\)

- Manually modify

---

**Relations**

<table>
<thead>
<tr>
<th></th>
<th>Word pairs</th>
<th>Adding methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synonyms</td>
<td>Interface</td>
<td>port</td>
</tr>
<tr>
<td></td>
<td>DOWN</td>
<td>UP</td>
</tr>
<tr>
<td></td>
<td>powerDown</td>
<td>powerUp</td>
</tr>
<tr>
<td>Relations</td>
<td>(interface, changed, state)</td>
<td></td>
</tr>
</tbody>
</table>

---


Log-specific word embedding combines two existing methods:

- **Lexical Information word embedding (LWE)** \(^{[1]}\) -> ants & syns
- **Semantic Word embedding (SWE)** \(^{[2]}\) -> relation triples

---


- **We adopt MIMICK**\(^3\) **to handle OOV words at runtime.**
- **Learn a function from spelling to distributional embeddings.**

---

**Historical logs:**
- \(L_1\): Interface ae3, changed state to **down**
- \(L_2\): Interface ae3, changed state to **up**
- \(L_3\): Interface ae1, changed status to **down**
- \(L_4\): Interface ae1, changed status to **up**

**Real-time logs:**
- \(L_5\): **Vlan-interface** vlan22, changed state to **down**
- \(L_6\): **Vlan-interface** vlan22, changed state to **up**

**Out-of-vocabulary**
- **Relation triples:** (interface, changed, status)
- **Antonym pairs:** (down, up)
- **Synonym pairs:** (state, status)

---

Log vector generation (Online stage)

1. Determine whether each word in logs is in vocabulary
2. Convert existing words to word vectors
3. Assign a new embedding vector to the OOV word
4. Calculate the log vector by averaging of its word vectors.
Evaluation
Experimental setting

Datasets:

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Description</th>
<th># of logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC</td>
<td>High performance cluster</td>
<td>433,489</td>
</tr>
<tr>
<td>HDFS</td>
<td>Hadoop distributed file system</td>
<td>11,175,629</td>
</tr>
<tr>
<td>ZooKeeper</td>
<td>ZooKeeper service</td>
<td>74,380</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Hadoop MapReduce job</td>
<td>394,308</td>
</tr>
</tbody>
</table>

Experimental setup:

- Linux server with Intel Xeon 2.40 GHz CPU
To highlight the challenge in processing OOV words

Generate training sets with the percentage of original logs ranging from 10% to 90% and regard the remaining logs as the testing set.

OOV words has a big percentage when trained on a smaller sample.

Always more than 90% logs contain OOV words in Spark/Windows.

It’s important to handle OOV words.
Evaluation of OOV processor

- Randomly select a word in each log
- Changed one of the letters to make the word as an OOV
- Test the similarity between the changed log and the original log

### Distribution of Logs' Similarity

![Graph showing the distribution of logs' similarity](image)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Spark</th>
<th>HDFS</th>
<th>Windows</th>
<th>Hadoop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Similarity</td>
<td>0.964</td>
<td>0.984</td>
<td>0.993</td>
<td>0.996</td>
</tr>
</tbody>
</table>

**Average similarity** when Log2Vec processes logs with OOV words
Log-based service management task

- **Online log classification**
  - Baselines: LogSig, FT-tree, Spell, template2Vec
  - Divide: 50% training set and 50% testing set

Average Fscore of Log2Vec is 0.944
Average Fscore of baselines 0.745

Log2Vec is stable

Comparison of log classification when use 50% training logs
Summary
Summary

Semantic-aware representation framework for online log analysis

We have open-sourced Log2Vec,

A mechanism for generating OOV word embeddings when new types of logs appear

The results are excellent
Thanks

mwbp16@mails.tsinghua.edu.cn

Open source toolkit: https://github.com/WeibinMeng/Log2Vec