Semantic Estimation for Texts in Software Engineering

Reporter: Xiaochen Li
Dalian University of Technology, China
Oscar Lab

- Ph.D. candidate at OSCAR Lab, in Dalian University of Technology, China, under supervision with Prof. He Jiang from 2015. **OSCAR**: Optimizing Software by Computation from Artificial intelligence
Oscar Lab

- Mining software repositories
  - API mining
  - Crowd testing reports
  - Code search
  - Design pattern mining
  - Mobile APP mining
- Program & testing
  - Model checking
  - Complier optimization
- Search based software engineering
  - Next release problem
  - Software Task Allocation

He Jiang
Lab Manager, Ph.D., Professor

- 2 Professors
- 1 Associate Professor
- 1 Lecturer
- 7 PhD. Candidates
- 17 Master Students
Overview

Texts in Software Engineering (SE)

- 4,200,000 test logs / year in industry
- 300,000 projects in GitHub

- 5,000,000 Q&A in Stack Overflow
- 485,000 bug reports in Eclipse Repo.
Overview

**Texts in Software Engineering (SE)**

- Classify test logs
- Search APIs by queries
- Seek codes by asking questions
- Read bug reports

---

Texts mixed of Natural Language (NL) words and APIs or codes in Software Language (SL)
Overview

Semantic estimation for SE texts

- Given texts mixed natural language words and software APIs or codes,
  - how to estimate the similarity between texts?
  - how to find salient sentences in the text?

relatedness

importance
Overview

Semantic estimation work

- **Cosine similarity** + KNN
- Analyze the failure causes of test scripts
- **Word embedding**
- Recommend API sequences
- **Crowdsourcing**
- Summarize bug reports
- **Deep neural network**
- Summarize bug reports

Shallow Bag-of-words → Deep Continuous spaces
Overview

Semantic estimation work

- Cosine similarity+ KNN
- Analyze the failure causes of test scripts
- Word embedding
- Recommend API sequences
- Link API documents to Ques.

Crowdsourcing
- Summarize bug reports

Shallow Bag-of-words

Deep Continuous spaces

Deep neural network
- Summarize bug reports
Analyze failed test scripts

Semantic estimation work

- Cosine similarity + KNN
- Analyze the failure causes of test scripts

1. Why do we analyze failed test scripts?
   - Failure causes are complex
   - Testers manually read logs to analyze
   - Logs are lengthy and complex

2. How do we do that?
   - Cosine similarity
   - KNN

3. What are the results?
Continuous integration increases SIT’s frequency.
- DevOps: faster time to market
- Cloud-based system: run 1,000 test scripts in 25 minutes

Running test scripts in SIT may fail.
- We find 6000+ failures in a single month in one product

Testers need to figure out the failure causes
- Require the stakeholders to fix them
**Background**

**Test alarms in SIT**

- Test scripts may fail for various causes
  - A test alarm is an alarm to warn the test script failure

<table>
<thead>
<tr>
<th>ID</th>
<th>Type of cause</th>
<th>Testers’ solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Obsolete test</td>
<td>update test scripts</td>
</tr>
<tr>
<td>C2</td>
<td>Product code defect</td>
<td>submit bugs to developers</td>
</tr>
<tr>
<td>C3</td>
<td>Configuration error</td>
<td>correct configuration files</td>
</tr>
<tr>
<td>C4</td>
<td>Test script defect</td>
<td>debug test scripts</td>
</tr>
<tr>
<td>C5</td>
<td>Device anomaly</td>
<td>submit bugs to instrument suppliers</td>
</tr>
<tr>
<td>C6</td>
<td>Environment issue</td>
<td>diagnose the environment</td>
</tr>
<tr>
<td>C7</td>
<td>Software problem</td>
<td>ask site reliability engineers to diagnose</td>
</tr>
</tbody>
</table>
The Problem

Test alarm analysis

• Analyze the cause of test alarms (test script failure) by test logs
  ➢ Test logs are easy to get
  ➢ Testers also read test logs to analyze the alarms
The Problem

A test log

- Bilingual documents: English & Chinese
- Long: more than 1000 lines, more than 10GB (14,000 logs)
CAM’s Idea

➢ Search the test logs of historical test alarms that may have the same failure cause with the new test log
Test log preprocess

- Language Detection

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [exception happens continuously for more than 20 times]
[2015-06-28 02:10:52.964] timed out while waiting for more data
Test log preprocess

- Language Detection
- English NLP
  - Tokenization,
  - Stop words removal
  - Stemming

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”

E [exception happens continuously for more than 20 times]
[2015-06-28 02:10:52.964] timed out while waiting for more data

E [] [2015-06-28 02:10:52.964] \ timed \ out \ while \ waiting \ for
\ more \ data
Cause Analysis Model (CAM)

Test log preprocess

- Language Detection
- English NLP
  - Tokenization,
  - Stop words removal (single letters, punctuation marks, and numbers ),
  - Stemming
- Chinese NLP
  - Word segmentation

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [exception happens continuously for more than 20 times]
[2015-06-28 02:10:52.964] timed out while waiting for more data

E [] [2015-06-28 02:10:52.964] \ timed \ out \ while \ waiting \ for \ more \ data

exception \ happens \ continuously \ for more than \ 20 \times
 Cause Analysis Model (CAM)

Test log preprocess

- Language Detection
- English NLP
  - Tokenization,
  - Stop words removal
    - (single letters, punctuation marks, and numbers ),
  - Stemming
- Chinese NLP
  - Word segmentation
- Term Integration
  - bag-of-words

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [exception happens continuously for more than 20 times]
[2015-06-28 02:10:52.964] timed out while waiting for more data
Cause Analysis Model (CAM)

**Cause prediction**

- Log similarity with historical logs
  - 2-shingling terms (successfully applied in information retrieval)
  - TF-IDF based cosine similarity

<table>
<thead>
<tr>
<th>Logs</th>
<th>Func. Point</th>
<th>$\text{Sim}_{\text{log}}$</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>his3</td>
<td>AUS</td>
<td>0.586</td>
<td>C2</td>
</tr>
<tr>
<td>his4</td>
<td>AUS</td>
<td>0.472</td>
<td>C3</td>
</tr>
<tr>
<td>his1</td>
<td>AUS</td>
<td>0.322</td>
<td>C3</td>
</tr>
<tr>
<td>his2</td>
<td>AUS</td>
<td>0.320</td>
<td>C3</td>
</tr>
<tr>
<td>his5</td>
<td>AUS</td>
<td>0.134</td>
<td>C2</td>
</tr>
</tbody>
</table>
Cause Analysis Model (CAM)

Cause prediction

- Predict by k-Nearest Neighbor
  - Case 1: the similarity of top 1 log (his3) exceeds a threshold
  - Case 2: the similarity of top 1 log (his3) is lower than a threshold
    - $C_2 = 0.586 + 0.134; \quad C_3 = 0.472 + 0.311 + 0.320$

<table>
<thead>
<tr>
<th>Logs</th>
<th>Func. Point</th>
<th>$\text{Sim}_{\text{log}}$</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>his3</td>
<td>AUS</td>
<td>0.586</td>
<td>C2</td>
</tr>
<tr>
<td>his4</td>
<td>AUS</td>
<td>0.472</td>
<td>C3</td>
</tr>
<tr>
<td>his1</td>
<td>AUS</td>
<td>0.322</td>
<td>C3</td>
</tr>
<tr>
<td>his2</td>
<td>AUS</td>
<td>0.320</td>
<td>C3</td>
</tr>
<tr>
<td>his5</td>
<td>AUS</td>
<td>0.134</td>
<td>C2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logs</th>
<th>Func. Point</th>
<th>$\text{Sim}_{\text{log}}$</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>his3</td>
<td>AUS</td>
<td>0.586</td>
<td>C2</td>
</tr>
<tr>
<td>his4</td>
<td>AUS</td>
<td>0.472</td>
<td>C3</td>
</tr>
<tr>
<td>his1</td>
<td>AUS</td>
<td>0.322</td>
<td>C3</td>
</tr>
<tr>
<td>his2</td>
<td>AUS</td>
<td>0.320</td>
<td>C3</td>
</tr>
<tr>
<td>his5</td>
<td>AUS</td>
<td>0.134</td>
<td>C2</td>
</tr>
</tbody>
</table>

Case 1 threshold = 0.5
Case 2 threshold = 0.6
Experimental Setup

- Two industrial testing projects at Huawei-Tech Inc.
  - 14,000 test logs of failed test scripts, manually labeled

- Evaluation method
  - Accuracy, Area-Under-Curve
  - Running time, memory consumption
  - Incremental framework (simulate testers’ daily work)

- Baseline Algorithms: bag-of-words
  - Lazy Associative Classifier (LAC)
  - Best First Tree (BFT)
  - Topic Model (TM)
Experimental Results

Overall performance

- How does CAM perform against baseline algorithms?

- Outperform the baseline algorithms ($p<0.05$)
- Superior over the majority of cause types
- Resources saving, take about 0.1s and less than 4GB memory to process a test log.

Fig. 1 Accuracy

Fig. 2 Comparison on computation resources
Experimental Results

Evaluation in real scenario

- How does CAM perform in a real development scenario?
  - 72% accuracy after running for two months.

- Feedback
  - CAM is better than manually building regular expressions.
  - Actually, I will not believe in an automatic tool. However, after presenting the historical test logs, I can quickly decide whether the prediction is correct. CAM accelerates my work.
  - Suggestions: labeling the defect-related snippets, provide suggestions on how to fix defects.
Overview

Semantic estimation work

- **Cosine similarity** + KNN
- Analyze the failure causes of test scripts
- Word embedding
- Recommend API sequences
- Link API documents to Ques.
- Crowdsourcing
- Summarize bug reports
- Deep neural network
- Summarize bug reports

Shallow Bag-of-words

Deep Continuous spaces

Shallow Bag-of-words

Deep Continuous spaces
Word2API

Semantic estimation work

- Word embedding
- Recommend API sequences
- Link API documents to Ques.

1. Why do we need word embedding?
   - Relatedness between words and APIs
   - Better than bag-of-words

2. How do we do that?
   - Collect large documents having words & APIs
   - Word embedding

3. What are the results?
Semantic gaps

- Gaps between natural languages and APIs
  - High-level vs. Low-level
  - For example: *read a file*

```java
File file = new File();
FileReader fr = new FileReader(file);
BufferedReader br = new BufferedReader(fr);
String line = "";
while (null != (line = br.readLine())) {
    System.out.println(line);
}
```

- `java.io.File#new`,
- `java.io.FileReader#new`,
- `java.io.BufferedReader#new`,
- `java.io.BufferedReader#readLine`
Word Embedding

Words into low-dimension vectors

- Easy to implement
  - Prepare a dataset
- Word2Vec Tool
- Run CBOW or Skip-gram
Word Embedding

Continuous Bag-of-Words model CBOW

- Minimize differences between output and $w_x$

$\mathcal{L}_M = \frac{1}{X} \sum_{x=1}^{X} \log p(w_x | W^d_x)$

Example of one-hot vector for word “current”

{begin, xx, ..., current, ..., xx, end}

{0, 0, ..., 1, ..., 0, 0}
Word Embedding

Challenge

• Acquisition challenge
  ➢ how to collect large numbers of documents that contain diversity words and APIs

Interface IPageLayout
Description: A page layout defines the initial layout for a perspective within a page in a workbench window... View placeholders may also have a secondary id. ... For example, the placeholder "someView:" will match any occurrence of the view that has primary id "someView" and that also has some non-null secondary id. Note that this placeholder will not match the view if it has no secondary id ...

➢ org.w3c.dom.views.DocumentView#getDefaultView()
➢ java.x.swing.text.View.ComponentView#new()
Word Embedding

Challenge

• Alignment challenge

➢ how to make semantically related words and APIs co-occur in a fixed window size

Interface IPageLayout
Description: A page layout defines the initial layout for a perspective within a page in a workbench window... View placeholders may also have a secondary id. ... For example, the placeholder "someView:*" will match any occurrence of the view that has primary id "someView" and that also has some non-null secondary id. Note that this placeholder will not match the view if it has no secondary id...
Word Embedding

Word2API

- Collect source codes and APIs from GitHub (**acquisition**)
- Pre-process words & APIs with NLP and Abstract Syntax Trees
- Shuffle words and APIs (**alignment**)
- Run Word Embedding Modeling
Word2API

Data acquisition

- GitHub from 2008-2016
  - 391,690 Java projects
  - 31,211,030 source code files
  - Many words and APIs that developers used
Word2API

Word-API Tuples Construction

- NLP
  - tokenization,
  - Stop word removal,
  - Stemming

open a file and output the contents

**Word sequence**
<open, file, output, content>
Word2API

Word-API Tuples Construction

- AST (Abstract Syntax Trees)
- Finding API fully qualified name in the text

```java
public void readFile(String path) throws IOException {
    File file = new File(path);
    FileReader fr = new FileReader(file);
    BufferedReader br = new BufferedReader(fr);
    String line = "";
    while (null != (line = br.readLine())) {
        System.out.println(line);
    }
}
```

API Sequence
<java.io.File#new,
java.io.FileReader#new,
java.io.BufferedReader,
java.lang.String#new,
java.io.BufferedReader#readLine,
....>

NLP for Comments
```java
/**
 * Open a file and output the contents
 * @param path
 * @return
 * @throws IOException
 */
```

Word Sequence: Word1 → Word2 → …

API Sequence: API1 → API2 → API3 → …

AST for Codes
```
expr
  +
  term
  println("+")

2
println("2")
```

Word-API tuples construction
Word2API

Word-API Tuples Construction

- 13,883,230 tuples

<word1, word2, ..., API1, API2...>
Word2API

Training Set Creation

- 13,883,230 tuples

<word1, word2, word3, word4, word5,..., API1, API2, API3...>
The underlying reason of the above procedure is that if a word and an API are semantically related, they tend to co-occur in the same tuple. After shuffling, the related words and APIs will have higher chances to locate in the same window than unrelated ones when the corpus is a large

- 138,832,300 shuffled results
- >30 GigaByte.
Word2API

Word Embedding Modeling

- 87,270 word vectors
- 37,431 API vectors
- Semantic estimation with these vectors

- Word-API similarity
  \[
  \text{sim}(w, a) = \frac{V_w \cdot V_a}{|V_w||V_a|}.
  \]

- Words-APIs similarity
  \[
  \text{sim}(W, A) = \frac{1}{2} \left( \frac{\sum (\text{maxSim}(w, A) \times \text{idf}(w))}{\sum \text{idf}(w)} \right) + \frac{\sum (\text{maxSim}(a, W) \times \text{idf}(a))}{\sum \text{idf}(a)}.
  \]
Query augmentation

● For API Sequences Recommendation

Query

"read a file"

API Seq.

• java.io.File#new,
• java.io.FileReader#new,
• java.io.BufferedReader#new,
• java.io.BufferedReader#readLine

These sequences are retrieved from source code corpus, e.g. GitHub corpus.
**Application 1**

**Query augmentation algorithms**

- Augment queries into API vectors

**Query** → **API vector** → **API Seq.**

- SWIM: Word Alignment based Augmentation
- CodeHow: API Description based Augmentation
- Word2API based Augmentation
### TABLE III: Performance of query augmentation algorithms over 30 human written queries.

<table>
<thead>
<tr>
<th>ID</th>
<th>Query</th>
<th>SWIM</th>
<th>CodeHow</th>
<th>Word2API</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FR RR5 RR10</td>
<td>FR RR5 RR10</td>
<td>FR RR5 RR10</td>
</tr>
<tr>
<td>Q1</td>
<td>convert int to string</td>
<td>11 1 0</td>
<td>11 0 0</td>
<td>3 0 0.2 0.1</td>
</tr>
<tr>
<td>Q2</td>
<td>convert string to int</td>
<td>1 1 0.5</td>
<td>1 0 0</td>
<td>1 0.8 0.8</td>
</tr>
<tr>
<td>Q3</td>
<td>append string</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q4</td>
<td>get current time</td>
<td>11 0 0</td>
<td>11 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q5</td>
<td>parse datetime from string</td>
<td>10 0 0.1</td>
<td>11 0 0</td>
<td>1 1 0.7</td>
</tr>
<tr>
<td>Q6</td>
<td>test file exists</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 0.8 0.8</td>
</tr>
<tr>
<td>Q7</td>
<td>open a url</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 0.8 0.8</td>
</tr>
<tr>
<td>Q8</td>
<td>open file dialog</td>
<td>11 0 0</td>
<td>1 0.8 0.7</td>
<td>1 0.4 0.7</td>
</tr>
<tr>
<td>Q9</td>
<td>get files in folder</td>
<td>11 0 0</td>
<td>1 0.8 0.9</td>
<td>1 1 0.9</td>
</tr>
<tr>
<td>Q10</td>
<td>match regular expressions</td>
<td>1 1 0.8</td>
<td>1 0.6 0.7</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q11</td>
<td>generate md5 hash code</td>
<td>11 0 0</td>
<td>11 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q12</td>
<td>generate random number</td>
<td>1 0.4 0.2</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q13</td>
<td>round a decimal value</td>
<td>11 0 0</td>
<td>3 0.2 0.1</td>
<td>1 0.8 0.8</td>
</tr>
<tr>
<td>Q14</td>
<td>execute file</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q15</td>
<td>copy file</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q16</td>
<td>create file</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q17</td>
<td>copy a file and save it to your destination path</td>
<td>1 1 1</td>
<td>3 0.6 0.4</td>
<td>1 0.8 0.9</td>
</tr>
<tr>
<td>Q18</td>
<td>delete files and folders in a directory</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q19</td>
<td>reverse a string</td>
<td>11 0 0</td>
<td>11 0 0</td>
<td>11 0 0</td>
</tr>
<tr>
<td>Q20</td>
<td>create socket</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q21</td>
<td>rename a file</td>
<td>11 0 0</td>
<td>11 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q22</td>
<td>download file from url</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q23</td>
<td>serialize an object</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q24</td>
<td>read binary file</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q25</td>
<td>save an image to a file</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q26</td>
<td>write an image to a file</td>
<td>1 1 1</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q27</td>
<td>parse xml</td>
<td>11 0 0</td>
<td>11 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Q28</td>
<td>play audio</td>
<td>11 0 0</td>
<td>1 0.8 0.9</td>
<td>1 0.4 0.5</td>
</tr>
<tr>
<td>Q29</td>
<td>play the audio clip at the specified absolute URL</td>
<td>11 0 0</td>
<td>1 1 1</td>
<td>1 0.6 0.4</td>
</tr>
<tr>
<td>Q30</td>
<td></td>
<td></td>
<td></td>
<td>1.933 0.680 0.677</td>
</tr>
</tbody>
</table>

Position of first correct API seq.: lower is better
Ratio of correct API seq.: higher is better

Average scores over 30 queries

<table>
<thead>
<tr>
<th></th>
<th>SWIM</th>
<th>CodeHow</th>
<th>Word2API</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR</td>
<td>5.633</td>
<td>4.467</td>
<td>1.933</td>
</tr>
<tr>
<td>RR5</td>
<td>0.513</td>
<td>0.547</td>
<td>0.680</td>
</tr>
<tr>
<td>RR10</td>
<td>0.463</td>
<td>0.533</td>
<td>0.677</td>
</tr>
</tbody>
</table>
Application 2

API documents linking

- Link API documents with Stack Overflow questions

  ➢ Question: "Are there any good CachedRowSet implementations other than the proprietary Sun one?"

  You shouldn't be directly instantiating implementation of CachedRowSet -- use its Provider to obtain an instance: see

  http://docs.oracle.com/javase/7/docs/api/javax/sql/rowset/RowSetProvider.html (available since JDK7)

  In fact, CachedRowSet's interface and related
Application 2

Word2API for API Doc. Linking

- Collect words in the question
  - *are there any good CachedRowSet implementations other than the proprietary Sun one*

- Collect APIs in API documents
  - `javax.sql.rowset.RowSetProvider#newFactory`
  - `javax.sql.rowset.RowSetProvider#createCachedRowSet`
  - ......
Application 2

Results

- MAP: Mean Average Precision
- MRR: Mean Reciprocal Rank

Algorithms

- VSM: bag-of-words
- Embedding: previous work
- VSM+XXX: combined

1. Word2API can bridge gaps betw. NL and SL
2. Word Embedding is better that bag-of-words here
3. We can combine different techniques for better results

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSM</td>
<td>0.232</td>
<td>0.259</td>
</tr>
<tr>
<td>Embedding</td>
<td>0.313</td>
<td>0.354</td>
</tr>
<tr>
<td>Word2API</td>
<td>0.402</td>
<td>0.433</td>
</tr>
<tr>
<td>VSM+Embedding</td>
<td>0.340</td>
<td>0.380</td>
</tr>
<tr>
<td>VSM+Word2API</td>
<td>0.436</td>
<td>0.469</td>
</tr>
</tbody>
</table>
Conclusion

Semantic estimation work

- **Cosine similarity** + **KNN**
- Analyze the failure causes of test scripts
- **Word embedding**
- Recommend API sequences
- Link API documents to Ques.
- **Crowdsourcing**
- Summarize bug reports
- **Deep neural network**
- Summarize bug reports
- **Shallow Bag-of-words**
- **Deep Continuous spaces**
Thanks

Reporter:  Xiaochen Li
Dalian University of Technology, China