Unsupervised Deep Bug Report Summarization

Authors: Xiaochen Li$^1$, He Jiang$^1$, Dong Liu$^1$, Zhilei Ren$^1$, Ge Li$^2$

$^1$Dalian University of Technology, $^2$Peking University
BACKGROUND

Summarize software artifacts

The increasing software artifacts motivate a large body of work in software artifacts summarization.

Source code summarization
- Haiduc et al. 2010
- Moreno et al. 2013
- Sridhara et al. 2010
- McBurney and McMillan 2011

Source code comment summarization
- Rastkar et al. 2011
- Ying and Robillard 2013

Bug report summarization
- Rastkarude et al. 2014
- Czarnecki et al. 2012
- Mani et al. 2012
- Jiang et al. 2017

Development activity summarization
- Treude et al. 2015

Over 80 million projects
Over 3 million applications
Over 5 million posts
A single bug repository, e.g., **Eclipse** Bugzilla repository, has already collected over **485,000** historical bug reports.
Stakeholders refer to historical bug reports

**Developers**
- Fix newly reported bugs by referring to similar historical bug reports for possible solutions

**Softw. Users**
- Wade through related bug reports before submitting a new one to avoid duplications
Stakeholders refer to historical bug reports

Developers

Fix newly reported bugs by referring to similar historical bug reports

Softw. Users

Wade through related bug reports before submitting a new one to avoid duplications

Need to read 600 sentences (avg.), if a user refers to only 10 historical bug reports
Summarize bug reports

Extracting and Highlighting informative sentences (summary) from description and comments

1. Open a large grayscale image of your choice (e.g. ....
2. Use “Tools/Color Tools/Threshold” to apply some threshold choosen.
3. Now you have a 8bit grayscale image, which actually consists only of color values “0” and color values “255”. ....

Xuan Baldauf 2005-03-18 14:46:31 UTC

Description

Xuan Baldauf 2005-03-18 14:46:31 UTC

1. Open a large grayscale image of your choice (e.g. ....
2. Use “Tools/Color Tools/Threshold” to apply some threshold choosen.
3. Now you have a 8bit grayscale image, which actually consists only of color values “0” and color values “255”. ....

Xuan Baldauf 2005-03-18 14:46:31 UTC

Description

Xuan Baldauf 2005-03-18 14:46:31 UTC

1. Open a large grayscale image of your choice (e.g. ....
2. Use “Tools/Color Tools/Threshold” to apply some threshold choosen.
3. Now you have a 8bit grayscale image, which actually consists only of color values “0” and color values “255”. ....
Bug reports are special

Bug report summarization
- Rastkarude et al. 2014
- Czarnecki et al. 2012
- Mani et al. 2012
- Jiang et al. 2017

1. Conversation-based text with frequent evaluation behaviors
2. Consist of different sentence types
3. Combined with many predefined fields
MOTIVATION

1. Conversation-based text with frequent evaluation behaviors

The evaluated sentences are frequently discussed and important

Adam D. Moss 2005-03-20 12:26:09 UTC  Comment 10
The 'mono' palette option doesn't even bother to star......

Xuân Baldauf 2005-03-20 13:06:07 UTC  Comment 11
<i>quote> The 'mono' palette option doesn't ......
I don't think that this operation is so rare, ...
</i>

Be Evaluated
2. Consist of different sentence types

- Natural language sentences by the **reporter**;
- Natural language sentences by the **participators**;
- Software language sentences (code snippets and system messages).
3. Combined with many predefined fields

Sentences contain words in the predefined fields may be informative

**Product**: GIMP

**Component**: General

**Version**: 2.2.x

**Hardware**: Other All
Bug reports are special

1. Conversation-based text with **frequent** evaluation behaviors

2. Consist of **different** sentence types

3. Combined with many predefined fields

Summarize bug reports by considering the special characteristics
The framework

A. Use the new bug reports and similar ones to train a machine learning model

B. Use the model to calculate the weight of words in the new bug report

C. Calculate each sentence score by the word weights

D. Generate summary according to the sentence scores
We select AutoEncoder as the machine learning model. A typical architecture of AutoEncoder consists of an input layer, one or more hidden layers, and an output layer. The output layer is defined as a pattern to reconstruct the input layer. The hidden state provides a compressed representation of the input layer. The weights of words can be measured by calculating how much information of a word is reserved in the hidden states. Why AutoEncoder?
### FRAMEWORK

The inputs are word vectors of bug reports.

#### Description

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xuan Baldauf 2005-03-20 13:06:07 UTC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Open a large grayscale or color image of your choice and choose.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Use “Tools → Layers” to apply the needed changes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Now you have a grayscale or color image, which consists only of color values “0” and color values “255”.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>This slow speed is not acceptable for interactive image processing, and this slowness is not necessary at all.</td>
</tr>
</tbody>
</table>

#### Comment

<table>
<thead>
<tr>
<th>Manish Singh 2005-03-19 17:48:16 UTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision 1.1.56.….</td>
</tr>
<tr>
<td>if (palette_type == GIMP_WEB_PALETTE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adam D. Moss 2005-03-20 12:26:09 UTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>The ‘mono’ palette option doesn’t even bother to start this pre-pass because it could only possibly pay off the extra effort if the entire image is pure black and pure white, which is expected to be a comparatively rare occurrence.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xuan Baldauf 2005-03-20 13:06:07 UTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;quote&gt; The ‘mono’ palette option doesn’t …..</td>
</tr>
<tr>
<td>I don’t think that this operation is so rare, and then a “convert to 1bit” operation to actually adjust the internal memory requirements.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adam D. Moss 2005-03-20 14:01:48 UTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; and then a &quot;convert to 1bit&quot; operation to actually adjust</td>
</tr>
<tr>
<td>&gt; the internal memory requirements.</td>
</tr>
<tr>
<td>&gt; If you mean GIMP’s internal memory requirements ….</td>
</tr>
</tbody>
</table>

#### Transform each sentence into a vector

<table>
<thead>
<tr>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>S8</th>
<th>S9</th>
<th>S10</th>
<th>S11</th>
<th>S12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**FRAMEWORK**

The inputs are word vectors of bug reports

<table>
<thead>
<tr>
<th>Description</th>
<th>New Bug Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xuan Baldauf</td>
<td>2005-03-20 13:06:07 UTC</td>
</tr>
<tr>
<td>S1</td>
<td>0 0 … 0</td>
</tr>
<tr>
<td>S2</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>This slow speed is not acceptable for interactive image processing, and this slowness is not necessary at all.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comment</th>
<th>Similar Bug Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manish Singh 2005-03-19 17:48:16 UTC</td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>Revision 1.156. …</td>
</tr>
<tr>
<td>S7</td>
<td>if (palette_type == GIMP_WEB_PALETTE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Evaluation Enhancement</th>
</tr>
</thead>
<tbody>
<tr>
<td>S8 0 0 … 1</td>
</tr>
<tr>
<td>S9 3 4 … 3</td>
</tr>
<tr>
<td>S10 0 3 … 2</td>
</tr>
<tr>
<td>S11 0 2 … 1</td>
</tr>
<tr>
<td>S12 0 1 … 2</td>
</tr>
</tbody>
</table>

Conversation-based text with frequent evaluation behaviors

S9 evaluates S8, if cosine similarity (S8, S9) > threshold (0.9).
**FRAMEWORK**

The inputs are word vectors of bug reports

### Evaluation Enhancement

- **S8** evaluates **S9** if cosine similarity $(S_8, S_9) >$ threshold (0.9).

### Conversation-based text with frequent evaluation behaviors

**New Bug Report**

**Similar Bug Reports**
Encode vectors according to their importance

**Inputs**

- \( S_6 + S_7 \)
  - Software Vector
  - \[ 1 \ 0 \ \ldots \ 2 \]

- \( S_1 + S_9 + S_{10} \)
  - Participator Vector
  - \[ 0 \ 6 \ \ldots \ 5 \]

- \( S_8 + S_{11} + S_{12} \)
  - Reporter Vector
  - \[ 6 \ 8 \ \ldots \ 6 \]

Consist of different sentence types

- reporter’s sentences > participators’ sentences > software sentences

Encode less important sentences (software vector) three times to reduce their influence.

\[
\text{Hidden states} = E_3(E_2(E_1(\text{Software Vector})))
\]
Initialize the network by predefined fields

Randomly initialize the network parameters, e.g., $E_1$, $E_2$, $E_3$, and then maximize some parameters if it connects a word in the predefined fields (the red element).

Combined with many predefined fields
Revisit the framework (DeepSum)

**A. train a machine learning model**

**B. calculate the weight of words** by the trained parameters, e.g., $E_1$, $E_2$, $E_3$, $E_4$, $E_5$.

**Weight of word** $i$ =
$E_3(E_2(E_1(\text{word}_i \text{ in software vector})))$
$+ E_3(E_4(\text{word}_i \text{ in participator vector}))$
$+ E_5(\text{word}_i \text{ in reporter vector})$
Revisit the framework (DeepSum)

C. calculate each sentence score. sentence score = \[ \sum \text{word weight} \times \text{word frequency} \]

D. generate summary, select a set of sentences \( s_{select} \) by:

1. maximizing the total sentence score of \( s_{select} \);

2. total length of \( s_{select} \) < a length limitation (25% length of the bug report)

We generate a summary of the new bug report by dynamic programing. The summary is about \( \frac{1}{4} \) length of the new bug report and has a high total sentence score.
Evaluate with six metrics over two data sets.

- Evaluation metrics include precision, recall, F-score, pyramid, R1, R2

\[
\text{Precision} = \frac{\text{Num}_{\text{hit}}}{\text{Num}_{\text{selected}}}, \\
\text{Recall} = \frac{\text{Num}_{\text{hit}}}{\text{Num}_{\text{ExfRef}}}, \\
\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \\
\text{Pyramid} = \frac{\text{Num}_{\text{TotalLinks}}}{\text{Num}_{\text{MaxLinks}}}, \\
\text{Rouge-n} = \frac{\sum_{s \in \text{AbsRef}} \sum_{\text{gram}_n \in s} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{s \in \text{AbsRef}} \sum_{\text{gram}_n \in s} \text{Count}(\text{gram}_n)},
\]

- Data sets SDS and ADS with 36+96 manually annotated bug reports
**EVALUATION**

**Influence on bug report characteristics**

- A: evaluation enhancement: conversation-based text with frequent evaluation behaviors
- B: predefined fields enhancement: combined with many predefined fields

Both the characteristics have positive influence on summarizing bug reports. DeepSum successfully integrates these characteristics to summarize bug reports.
Influence on calculating word weights with stepped AutoEncoder

- TF Strategy: the word weights are the same as the Term Frequency in the new bug report
- AE Strategy: calculating word weights with standard AutoEncoder, not the stepped one.

DeepSum’s word weighting strategy (consider different sentence types) outperforms the alternatives, i.e., TF strategy and AE strategy.
EVALUATION

We compare DeepSum against algorithms in previous studies.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Pyramid</th>
<th>R-1</th>
<th>R-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRC_{LOO}</td>
<td>0.570</td>
<td>0.350</td>
<td>0.400</td>
<td>0.630</td>
<td>0.521</td>
<td>0.140</td>
</tr>
<tr>
<td>BRC_{FCV}</td>
<td>0.524</td>
<td>0.321</td>
<td>0.362</td>
<td>0.580</td>
<td>0.493</td>
<td>0.130</td>
</tr>
<tr>
<td>ACS_{LOO}</td>
<td>0.595</td>
<td>0.337</td>
<td>0.400</td>
<td>0.604</td>
<td>0.516</td>
<td>0.135</td>
</tr>
<tr>
<td>ACS_{FCV}</td>
<td>0.562</td>
<td>0.310</td>
<td>0.359</td>
<td>0.572</td>
<td>0.488</td>
<td>0.126</td>
</tr>
<tr>
<td>Centroid</td>
<td>0.536</td>
<td>0.269</td>
<td>0.343</td>
<td>0.460</td>
<td>0.471</td>
<td>0.126</td>
</tr>
<tr>
<td>MMR</td>
<td>0.617</td>
<td>0.353</td>
<td>0.429</td>
<td>0.551</td>
<td>0.498</td>
<td>0.145</td>
</tr>
<tr>
<td>Grasshopper</td>
<td>0.525</td>
<td>0.300</td>
<td>0.368</td>
<td>0.521</td>
<td>0.505</td>
<td>0.135</td>
</tr>
<tr>
<td>DivRank</td>
<td>0.591</td>
<td>0.301</td>
<td>0.378</td>
<td>0.546</td>
<td>0.500</td>
<td>0.139</td>
</tr>
<tr>
<td>Hurried</td>
<td>0.712</td>
<td>0.390</td>
<td>0.419</td>
<td>0.719</td>
<td>0.595</td>
<td>0.153</td>
</tr>
<tr>
<td><strong>DeepSum</strong></td>
<td><strong>0.621</strong></td>
<td><strong>0.388</strong></td>
<td><strong>0.462</strong></td>
<td><strong>0.624</strong></td>
<td><strong>0.563</strong></td>
<td><strong>0.177</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Pyramid</th>
<th>R-1</th>
<th>R-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRC_{LOO}</td>
<td>0.568</td>
<td>0.350</td>
<td>0.412</td>
<td>0.659</td>
<td>0.517</td>
<td>0.201</td>
</tr>
<tr>
<td>BRC_{FCV}</td>
<td>0.528</td>
<td>0.314</td>
<td>0.388</td>
<td>0.620</td>
<td>0.492</td>
<td>0.180</td>
</tr>
<tr>
<td>ACS_{LOO}</td>
<td>0.605</td>
<td>0.391</td>
<td>0.452</td>
<td>0.671</td>
<td>0.546</td>
<td>0.235</td>
</tr>
<tr>
<td>ACS_{FCV}</td>
<td>0.556</td>
<td>0.343</td>
<td>0.400</td>
<td>0.625</td>
<td>0.520</td>
<td>0.211</td>
</tr>
<tr>
<td>Centroid</td>
<td>0.488</td>
<td>0.280</td>
<td>0.337</td>
<td>0.561</td>
<td>0.473</td>
<td>0.183</td>
</tr>
<tr>
<td>MMR</td>
<td>0.505</td>
<td>0.356</td>
<td>0.395</td>
<td>0.585</td>
<td>0.503</td>
<td>0.206</td>
</tr>
<tr>
<td>Grasshopper</td>
<td>0.446</td>
<td>0.337</td>
<td>0.362</td>
<td>0.548</td>
<td>0.504</td>
<td>0.201</td>
</tr>
<tr>
<td>DivRank</td>
<td>0.445</td>
<td>0.282</td>
<td>0.325</td>
<td>0.545</td>
<td>0.498</td>
<td>0.202</td>
</tr>
<tr>
<td>Hurried</td>
<td>0.580</td>
<td>0.349</td>
<td>0.418</td>
<td>0.637</td>
<td>0.544</td>
<td>0.241</td>
</tr>
<tr>
<td><strong>DeepSum</strong></td>
<td><strong>0.606</strong></td>
<td><strong>0.394</strong></td>
<td><strong>0.457</strong></td>
<td><strong>0.681</strong></td>
<td><strong>0.553</strong></td>
<td><strong>0.249</strong></td>
</tr>
</tbody>
</table>

DeepSum shows promising performance for summarizing bug reports over distinct evaluation metrics.

**Bug report summarization**
- Rastkarude et al. 2014
- Czarnecki et al. 2012
- Mani et al. 2012
- Jiang et al. 2017
CONCLUSION AND FUTURE WORK

Conclusion
● We propose an **unsupervised deep learning algorithm** for bug report summarization.
● Our model fully **leverages the characteristics of bug reports**.
● Experiments over two public bug report datasets show that our model **outperforms the comparative algorithms** by adopting domain-specific characteristics.

Future Work
● Investigate whether automatic bug report summarization is useful in **a real developing scenario**.
● Construct **large bug report data sets** to evaluate different models.
Thanks

Unsupervised Deep Bug Report Summarization

Reporter: Xiaochen Li
Dalian University of Technology, China

Authors: Xiaochen Li¹, He Jiang¹, Dong Liu¹, Zhilei Ren¹, Ge Li²
¹Dalian University of Technology, ²Peking University