Automatic Code Review
by Learning the Revision of Source Code

Shu-Ting Shi¹, Ming Li¹,², David Lo³, Ferdian Thung³, Xuan Huo¹

¹National Key Laboratory for Novel Software Technology, Nanjing University
²Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing University
³School of Information Systems, Singapore Management University, Singapore

{shist, lim}@lamda.nju.edu.cn
Code keeps on changing

I've revised the code, is that ok?

```java
private static String
listToString(List<String> list) {
if (list == null)
    return "";
StringBuilder sb = new StringBuilder();
for (String item : list) {
    sb.append(item);
    sb.append(',');
}
return sb.toString();
}
```

Key point: Correctly **model the revision** between source code.

How to determine?
Modeling the revision is hard

The state-of-the-art software mining approaches leverage the *global* information by modeling the relationship between two pieces of code.

But apparently, model the revision needs to emphasize the *local* information, i.e., the revised small portion of codes.

**Key challenge:** model the revision of source code and avoid being misled by the overwhelming among of unchanged code in the revision.

This work
Outline

• What is code review?

• The proposed approach DACE

• Experiments

• Conclusion
Code Review

Code review is the process of inspection on two versions of a source code in order to improve the overall software quality and ensure the security of the project.

automating such the code review process will alleviate the burden of code reviewers and speed up the software maintenance process.
Automatic Code Review

Key challenge: how to model the revision.

Exiting models for modeling the relationship between software artifacts

- NPCNN [Huo and Li, 2017]
- CDLH [Wei and Li, 2017]
- LSCNN [Huo and Li, 2018]
- CDPU [Wei and Li, 2018]

But none of them can focus on the local feature to model the "difference".

these approaches are equipped for extract global feature to model the "similarity".
Challenge 1

How to model the revision of source code?

**Key idea:** A particularly designed pairwise autoencoder (PAE) is employed to learn the revision features. We construct an encoder to pact the revised code into a compact feature representations based on which the transformation can be learned.
**Challenge 2**

SAME CHANGE

**Same change, different result.**

Approved

Rejected

Context of the **changed hunks** provides rich information about the revision.
Challenge 2

• The changed hunk is usually short, while the context is usually long.

• Dilemma
  – Utilize all the context? 
    
    overwhelmed by same unchanged code lines. ✗
  – Discard the context?
    the revised code is too short to provide sufficient information for modeling. ✗

How to focus on modeling the revision but also considering the rich information in the context?

Key idea: Enrich the revised lines with the information of context around them.
Outline

• What is code review?

• The proposed approach DACE

• Experiments

• Conclusion
Formalization

- We formalize the code review as a learning task, which is a binary classification problem.

Given a sample of data: \((c^O_i, c^R_i, y_i), \quad c^O_i \in \mathcal{C}^O, c^R_i \in \mathcal{C}^R\)

- Learn a mapping \(f: \mathcal{C}^O \times \mathcal{C}^R \mapsto \mathcal{Y}\) that can be learned by minimizing the objective function:

\[
\min_f \sum_i \mathcal{L}(f(c^O_i, c^R_i), y_i) + \lambda \Omega(f)
\]
Model Structure of DACE

Output layer

Review result $y_i$

Fully connected layer

Pairwise AutoEncoder for model revision

Revision feature extraction layer

LSTM for enrichment

CNN for each statement

Embedding

Context enrichment layer

Input layer

Original code $c_i^O$

New code $c_i^R$
Model Structure

Output layer

Revision feature extraction layer

Context enrichment layer

Input layer

Review result $y_i$

Fully connected layer

Pairwise AutoEncoder for model revision

LSTM for enrichment

CNN for each statement

Embedding

Original code $c_i^O$

New code $c_i^R$

Challenge 1: How to model the revision
Model Structure

Revision feature extraction layer

Context enrichment layer

Input layer

Output layer

Review result $y_i$

Fully connected layer

Pairwise AutoEncoder for model revision

LSTM for enrichment

CNN for each statement

Embedding

Original code $c_i^O$

New code $c_i^R$

Challenge 1: How to model the revision

Challenge 2: How to balance the use of context
Context Enrichment Layer

- Extract the statement level features by convolutional layer
- Enrich the current statement with other statements in the context by exploiting the sequential relationship by LSTM layer

![Diagram of context enrichment layer]

Source code file with changed hunk

- Multiple filters within statements
- Statement feature
- Get interactions between statements

Convolutional layer
Pooling layer
LSTM layer

H context-enriched changed hunk feature for further prediction
Revision Feature Extraction Layer

Decoder

\[ \hat{S}_\text{seq}^O = \{\hat{s}_1^O, \hat{s}_2^O, \hat{s}_3^O, \ldots, \hat{s}_{T_1}^O\} \]

Encoder

\[ S_\text{seq}^O = \{s_1^O, s_2^O, s_3^O, \ldots, s_{T_1}^O\} \]

\[ S_\text{seq}^N = \{s_1^N, s_2^N, s_3^N, \ldots, s_{T_2}^N\} \]

\[ \hat{S}_\text{seq}^N = \{\hat{s}_1^N, \hat{s}_2^N, \hat{s}_3^N, \ldots, \hat{s}_{T_2}^N\} \]

Fusion

Reconstruction loss

\[ \mathcal{L}_{re} = \sum_{i=1}^{m} \left( \frac{1}{T_i^O} \right) \sum_{j=1}^{T_i^O} \left\| s_j^O - \hat{s}_j^O \right\| \]

Prediction loss

\[ \mathcal{L}_{cl} = -\sum_{i=1}^{m} (c_a y_i \log \hat{p}_i + c_r (1 - \hat{p}_i)) \]

HTTP://LAMDA.NJU.EDU.CN

College of Computer Science

Nanjing University

Learning And Mining from Data

http://lamda.nju.edu.cn/shist
Outline

• What is code review?

• The proposed approach DACE

• Experiments

• Conclusion
## Experiment Setting

### Dataset
- The code review data used in the experiments is crawled from Apache Code Review Board.
- For each changed hunk, if it has the highlighted lines that marked by reviewers denoting they have issues, we regard the hunk as rejected.

<table>
<thead>
<tr>
<th>Repository</th>
<th>#hunk</th>
</tr>
</thead>
<tbody>
<tr>
<td>accumulo</td>
<td>5,620</td>
</tr>
<tr>
<td>ambari</td>
<td>6,810</td>
</tr>
<tr>
<td>aurora</td>
<td>6,762</td>
</tr>
<tr>
<td>cloudstack</td>
<td>6,171</td>
</tr>
<tr>
<td>drill-git</td>
<td>3,575</td>
</tr>
<tr>
<td>hbase-git</td>
<td>6,702</td>
</tr>
</tbody>
</table>

### Evaluation measure:
- √ AUC
- √ F1
Experiment

- **TFIDF-LR**: uses TFIDF to represent the original and the revised source code, and LR is used for prediction. [Schütze et al., 2008][Gay et al., 2009]
- **TFIDF-SVM**: like above but SVM is used for prediction. [Schütze et al., 2008][Gay et al., 2009]

- **Deeper**: a state-of-the-art model on software engineering, which uses basic features for changes. [Yang et al., 2015]
- **Deeper-SVM**: like above but SVM is used for prediction. [Yang et al., 2015]

- **LSCNN**: a variant of DACE that without PAE. It is similar to the state-of-art method LSCNN and using Multilayer Perceptron (MLP) for prediction.
- **PAE**: a variant of DACE that only use PAE to extract features from CNN without considering sequential and long-term dependency information between statements.
# Experiment

<table>
<thead>
<tr>
<th>Repository</th>
<th>TFIDF-LR</th>
<th>TFIDF-SVM</th>
<th>Deeper</th>
<th>Deeper-SVM</th>
<th>LSCNN</th>
<th>PAE</th>
<th>DACE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>AUC</td>
<td>F1</td>
<td>AUC</td>
<td>F1</td>
<td>AUC</td>
<td>F1</td>
</tr>
<tr>
<td>accumulo</td>
<td>0.227</td>
<td>0.666</td>
<td>0.239</td>
<td>0.703</td>
<td>0.199</td>
<td>0.705</td>
<td>0.417</td>
</tr>
<tr>
<td>ambari</td>
<td>0.240</td>
<td>0.708</td>
<td>0.278</td>
<td>0.848</td>
<td>0.306</td>
<td>0.680</td>
<td>0.238</td>
</tr>
<tr>
<td>aurora</td>
<td>0.204</td>
<td>0.582</td>
<td>0.220</td>
<td>0.645</td>
<td>0.349</td>
<td>0.682</td>
<td>0.299</td>
</tr>
<tr>
<td>cloudstack</td>
<td>0.250</td>
<td>0.745</td>
<td>0.275</td>
<td>0.827</td>
<td>0.352</td>
<td>0.795</td>
<td>0.265</td>
</tr>
<tr>
<td>drill-git</td>
<td>0.212</td>
<td>0.658</td>
<td>0.236</td>
<td>0.725</td>
<td>0.229</td>
<td>0.593</td>
<td>0.212</td>
</tr>
<tr>
<td>hbase-git</td>
<td>0.232</td>
<td>0.679</td>
<td>0.256</td>
<td>0.759</td>
<td>0.193</td>
<td>0.590</td>
<td>0.154</td>
</tr>
<tr>
<td>average</td>
<td>0.228</td>
<td>0.673</td>
<td>0.251</td>
<td>0.751</td>
<td>0.272</td>
<td>0.671</td>
<td>0.228</td>
</tr>
</tbody>
</table>

DACE outperforms other state-of-the-art competing methods in term of F1 and AUC.
The internal structure of pairwise autoencoder helps DACE to captures the revision by learning the revision pattern by PAE.
## Experiment

<table>
<thead>
<tr>
<th>Repository</th>
<th>TFIDF-LR F1</th>
<th>TFIDF-LR AUC</th>
<th>Deeper F1</th>
<th>Deeper AUC</th>
<th>Deeper-SVM F1</th>
<th>Deeper-SVM AUC</th>
<th>LSCNN F1</th>
<th>LSCNN AUC</th>
<th>PAE F1</th>
<th>PAE AUC</th>
<th>DACE F1</th>
<th>DACE AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>accumulo</td>
<td>0.227</td>
<td>0.666</td>
<td>0.239</td>
<td>0.703</td>
<td>0.202</td>
<td>0.688</td>
<td>0.199</td>
<td>0.705</td>
<td>0.417</td>
<td>0.787</td>
<td>0.373</td>
<td>0.814</td>
</tr>
<tr>
<td>ambari</td>
<td>0.240</td>
<td>0.708</td>
<td>0.278</td>
<td>0.848</td>
<td>0.306</td>
<td>0.680</td>
<td>0.238</td>
<td>0.572</td>
<td>0.444</td>
<td>0.824</td>
<td>0.473</td>
<td>0.861</td>
</tr>
<tr>
<td>aurora</td>
<td>0.204</td>
<td>0.582</td>
<td>0.220</td>
<td>0.645</td>
<td>0.349</td>
<td>0.682</td>
<td>0.299</td>
<td>0.564</td>
<td>0.336</td>
<td>0.750</td>
<td>0.571</td>
<td>0.819</td>
</tr>
<tr>
<td>cloudstack</td>
<td>0.250</td>
<td>0.745</td>
<td>0.275</td>
<td>0.827</td>
<td>0.352</td>
<td>0.795</td>
<td>0.265</td>
<td>0.646</td>
<td>0.360</td>
<td>0.761</td>
<td>0.415</td>
<td>0.820</td>
</tr>
<tr>
<td>drill-git</td>
<td>0.212</td>
<td>0.658</td>
<td>0.236</td>
<td>0.725</td>
<td>0.229</td>
<td>0.593</td>
<td>0.212</td>
<td>0.540</td>
<td>0.318</td>
<td>0.780</td>
<td>0.382</td>
<td>0.806</td>
</tr>
<tr>
<td>hbase-git</td>
<td>0.232</td>
<td>0.679</td>
<td>0.256</td>
<td>0.759</td>
<td>0.193</td>
<td>0.590</td>
<td>0.154</td>
<td>0.524</td>
<td>0.348</td>
<td>0.751</td>
<td>0.411</td>
<td>0.764</td>
</tr>
<tr>
<td>average</td>
<td>0.228</td>
<td>0.673</td>
<td>0.251</td>
<td>0.751</td>
<td>0.272</td>
<td>0.671</td>
<td>0.228</td>
<td>0.592</td>
<td>0.370</td>
<td>0.777</td>
<td>0.438</td>
<td>0.814</td>
</tr>
</tbody>
</table>

PAE only not perform better than DACE because it abandons the context of revised hunk which is indispensable.
Outline

• What is code review?

• The proposed approach DACE

• Experiments

• Conclusion
Conclusion

• Major Contribution
  – Put forward a new software mining problem that aims to model the relationship between two source files by learning the revision features.
  – Propose a novel model DACE, which learns the revision features by modeling the hidden state, based on pairs of context-enriched representation of source code.

• Future work
  • Considering the relationship between hunks.
  • Incorporating additional data.
  • …

Thanks!