Automatically Generating Commit Messages from Diffs Using Neural Machine Translation

Problem Statement
• Commit messages are important for software change comprehension
• However, developers do not always write good commit messages

Contributions
• This paper adapts a neural machine translation (NMT) algorithm to automatically generate commit messages from program diffs
• The commit messages generated either have very high or very low quality
• Open source the data sets and software

Preparing a Data Set for NMT
• Initially started with 2M commits from the top 1k Java projects
  – As 82% of the commit messages have only one sentence, so this paper aims to generate one-sentence commit message
  – Remove issue ids from the extracted sentences and removed commit ids from diffs
    • Perhaps the commit ids should be also removed from the commit messages?

Preparing a Data Set for NMT (cont’d)
– Removed merge and rollback commits -> 1.8M
– Set the maximum length of diffs as 100 tokens, because the tool does not work well for the settings 50 and 200 -> 75k
– Remove messages not matching the V-DO patterns (verb/direct-object) -> 32k
– Split the 32k messages so that 26k are used for training, 3k for testing, and 3k for validation

The Approach

Part A
Pairs of diffs -> commit messages
Filter for V-DO Message Patterns

Part B
Evaluation of Predicted Messages
NMT Training Procedure

Part C
QA Filter for likely poor predictions
Updated Evaluation with QA Filter
**NMT Training and Testing**

- **Evaluation**
  - RQ1: Compared to the messages generated by a baseline, are the messages created by NMT more or less similar to the oracle?
  - RQ2: With V-DO filter enabled or disabled, how does the NMT model create messages?

- V-DO filter should be used to effectively remove messages with low quality or complex messages

**RQ1 & RQ2**

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>LenRef</th>
<th>LenGen</th>
<th>p1</th>
<th>p2</th>
<th>p3</th>
<th>p4</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOSES</td>
<td>3.63</td>
<td>129899</td>
<td>22872</td>
<td>8.3</td>
<td>5.6</td>
<td>2.7</td>
<td>2.1</td>
</tr>
<tr>
<td>NMT1</td>
<td>31.92</td>
<td>24344</td>
<td>22872</td>
<td>38.1</td>
<td>31.1</td>
<td>29.5</td>
<td>29.7</td>
</tr>
<tr>
<td>NMT2</td>
<td>32.81</td>
<td>21287</td>
<td>22872</td>
<td>40.1</td>
<td>34.0</td>
<td>33.4</td>
<td>34.3</td>
</tr>
</tbody>
</table>
| V-DO filter described in Section IV-B. NMT is a model trained without V-DO filter described in Section IV-D. LenGen is the total length of the generated messages (e in Equation (1)). LenRef is the total length of the reference messages (r in Equation (1)). The modified n-gram precision pn, where n = 1, 2, 3, 4, is defined in Equation (8). * This BLEU score is calculated on a test set that is not V-DO filtered described in Section V-A3. The other BLEU scores are tested on a V-DO filtered test set described in Section V-A4.

**Human Evaluation**

- **BLEU is a widely used metric to compare translation models, but it is not recommended for evaluating individual sentences**
- **Human evaluation can assess how similar a generated message is to the original human-created message**

**An Example**

Example 1 of 3

message 1: "Added X to readme"
message 2: "edit readme"

Recommended score: 6

Explanation: The two messages have only one shared word, "readme". But the two messages are very similar in meaning, because "Added" is a type of "edit".

**Results**

![Graph showing distribution of median scores in human study](image)
• To filter out the generated bad messages with SVM
  - Bad: 0, 1
  - Not bad: 2-7

Fig. 11: The predict results of the cross evaluation of QA filter. QA filter reduced 108 messages that are scored 0, 43 messages that are scored 1, 42 messages that are scored 2, 32 messages that are scored 3, 32 messages that are scored 4, 21 messages that are scored 5, 10 messages that are scored 6, and 6 messages that are scored 7. We note that although we trained the QA filter with binary labels, “bad” and “not bad”, the evaluation result shows that QA filter is able to reduce more messages for lower scores.

Results
• QA filter reduced 44% of the “bad” messages at a cost of 11% of the “good” messages (6-7)

An Exemplar Generated Good Message

<table>
<thead>
<tr>
<th>TABLE III: Example Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE:</td>
</tr>
<tr>
<td>-- a/core/.../CursorTableCursorAdapter.java</td>
</tr>
<tr>
<td>*** b/core/.../CursorTableCursorAdapter.java</td>
</tr>
<tr>
<td>99 -143, 8 -143, 7 # # public final class</td>
</tr>
<tr>
<td>CursorTableCursorAdapter ...</td>
</tr>
<tr>
<td>public void close() {</td>
</tr>
<tr>
<td>ncursor.deactivate();</td>
</tr>
<tr>
<td>ncursor.close();</td>
</tr>
<tr>
<td>public int querySTableObserver observer, ...</td>
</tr>
<tr>
<td>Generated Message:</td>
</tr>
<tr>
<td>Reference Message:</td>
</tr>
<tr>
<td>&quot;Call close () instead of deactivate () in CursorTableCursorAdapter. close () .&quot;</td>
</tr>
<tr>
<td>Scores: 7, 6, 7</td>
</tr>
</tbody>
</table>