SOTorrent

Reconstructing and Analyzing the Evolution of Stack Overflow Posts

Sebastian Baltes

@s_baltes

15th International Conference on Mining Software Repositories (MSR 2018)
May 28-29, Gothenburg, Sweden
Thanks to my co-authors!
Why Reconstruct and Analyze SO Post Evolution?

• The content of **14.3 million posts** has been **edited** after creation (SO data dump 2018-03-01)

• Like other **software artifacts**, SO posts **evolve over time**:
  • Bugs in code snippets are fixed
  • Clarifications are added in text documenting the code
  • Snippets are updated to new language/library versions

• **Copying code** from Stack Overflow (SO) is common, despite licensing, security, and maintainability implications
Why Reconstruct and Analyze SO Post Evolution?

Evolution of code on SO differs from regular software projects:

- **Short** code snippets (12 LOC on average)
- **No bug tracking** system (just comments and new answers)
- **No versioning** for individual snippets (just whole posts)
Example

Question

https://stackoverflow.com/q/309424

Answer

https://stackoverflow.com/a/5445161
Here's a way using only standard Java library (note that the stream is not closed, YMMV).

```java
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\A");
    return s.hasNext() ? s.next() : "";
}
```

I learned this trick from "Stupid Scanner tricks" article. The reason it works is because Scanner iterates over tokens in the stream, and in this case we separate tokens using "beginning of the input boundary" (A) thus giving us only one token for the entire contents of the stream.

Note, if you need to be specific about the input stream's encoding, you can provide the second argument to Scanner constructor that indicates what charset to use (e.g. "UTF-8").

Hat tip goes also to Jacob, who once pointed me to the said article.

EDITED: Thanks to a suggestion from Patrick, made the function more robust when handling an empty input stream. One more edit: mixed try/catch, Patrick's way is more laconic.
Here's a way using only standard Java library (note that the stream is not closed, YMMV).

```java
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s =
        new java.util.Scanner(is).useDelimiter("\A\n");
    return s.hasNext() ? s.next() : "";
}
```

I learned this trick from "Stupid Scanner tricks" article. The reason it works is because Scanner iterates over tokens in the entire contents of the stream, hence we separate tokens using a "boundary" (\A) thus giving back all of them.

Note, if you need to be specific about the input stream's encoding, you can provide the second argument to Scanner constructor that indicates what charset to use (e.g. "UTF-8").

Hat tip goes also to Jacob, who once pointed me to the said article.

**EDITED:** Thanks to a suggestion from Patrick, made the function more robust when handling an empty input stream. **One more edit:** Mixed try/catch, Patrick's way is more laconic.
Comments

Bug report

Alternative solution

Bug report

Bug report

Comment by author

This stuff is clearly a hack.

Sebastian Baltes - SOTorrent (MSR 2018)
Even for such a simple code snippet, the context is quite complex:

- The snippet is based on an external source
- Hidden in the comments, the author acknowledges that his solution is “clearly a hack”
- There are several bug reports pointing to issues
- Has the snippet been edited to fix those issues?
- Is the snippet safe to use?
SO Revisions

Problems:

- Version history is only available on the level of whole posts, thus **individual code snippets hard to trace**
- **Comments and edits** are not linked
- Unclear how **external sources** are related

https://stackoverflow.com/posts/5445161/revisions
SOTorrent
SOTorrent

Open dataset based on the official Stack Overflow data dump

**Problem:**

- Version history only available on the level of whole posts
- Analysis of individual code snippets
- Relation to external resources

**SOTorrent:**

- Reconstructed evolution of individual post blocks
- Possible
- Supports researchers in analyzing links from/to posts
Retrieve all versions of a code snippet:

```
SELECT PostHistoryId, Content, Length, LineCount, PredSimilarity
FROM PostBlockVersion
WHERE PostId=5445161 AND LocalId=2 AND PredEqual=0
ORDER BY PostHistoryId DESC;
```

<table>
<thead>
<tr>
<th>PostHistoryId</th>
<th>Content</th>
<th>Length</th>
<th>LineCount</th>
<th>PredSimilarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>155295527</td>
<td>static String convertStreamToString(java.io.In...</td>
<td>192</td>
<td>4</td>
<td>0.7532467532467533</td>
</tr>
<tr>
<td>154620092</td>
<td>static String convertStreamToString(java.io.In...</td>
<td>352</td>
<td>13</td>
<td>0.7532467532467533</td>
</tr>
<tr>
<td>44935719</td>
<td>static String convertStreamToString(java.io.In...</td>
<td>192</td>
<td>4</td>
<td>0.9846153846153847</td>
</tr>
<tr>
<td>31249705</td>
<td>public static String convertStreamToString(java...</td>
<td>199</td>
<td>4</td>
<td>0.9523809523809523</td>
</tr>
<tr>
<td>30827994</td>
<td>String convertStreamToString(java.io.InputStr...</td>
<td>185</td>
<td>4</td>
<td>0.6875</td>
</tr>
<tr>
<td>25270546</td>
<td>String convertStreamToString(java.io.InputStr...</td>
<td>239</td>
<td>7</td>
<td>0.9714285714285714</td>
</tr>
<tr>
<td>21289331</td>
<td>public String convertStreamToString(java.io.I...</td>
<td>246</td>
<td>7</td>
<td>0.8157894736842105</td>
</tr>
<tr>
<td>21230790</td>
<td>import java.util.Scanner; import java.util.No...</td>
<td>298</td>
<td>10</td>
<td>0.840579710149275</td>
</tr>
</tbody>
</table>
Retrieve line-based difference for latest version:

```
SELECT PostHistoryId, LocalId, PredLocalId, PostBlockDiffOperationId, Text
FROM PostBlockDiff
WHERE PostHistoryId=155295527 AND LocalId=2;
```

<table>
<thead>
<tr>
<th>PostHistoryId</th>
<th>LocalId</th>
<th>PredLocalId</th>
<th>PostBlockDiffOperationId</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>155295527</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Equal</td>
</tr>
<tr>
<td>155295527</td>
<td>2</td>
<td>2</td>
<td>-1</td>
<td>Delete</td>
</tr>
<tr>
<td>155295527</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>Insert</td>
</tr>
<tr>
<td>155295527</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Equal</td>
</tr>
</tbody>
</table>

```
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\A");
    if (is == null) {
        return "";
    }    java.util.Scanner s...
```
Reconstructing Stack Overflow Post Evolution
1. Post Block Extraction
Here's a way using only standard Java library (note that the stream is not closed, YMMV).

```java
2034
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\A");
    return s.hasNext() ? s.next() : "";
}
```

I learned this trick from "Stupid Scanner tricks" article. The reason it works is because Scanner iterates over tokens in the stream, and in this case we separate tokens using "beginning of the input boundary" (\A) thus giving us only one token for the entire contents of the stream.

Note, if you need to be specific about the input stream's encoding, you can provide the second argument to Scanner constructor that indicates what charset to use (e.g. "UTF-8").

Hat tip goes also to Jacob, who once pointed me to the said article.

EDITED: Thanks to a suggestion from Patrick, made the function more robust when handling an empty input stream. One more edit: mixed try/catch, Patrick's way is more laconic.
Here’s a way using only standard Java library (note that the stream is not closed, YMMV).&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;xA;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;xA; static String convertStreamToString(java.io.InputStream is) {&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;xA; java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\A");&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA; return s.hasNext() ? s.next() : "";&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA; I learned this trick from ["Stupid Scanner tricks"]\[1\] article. The reason it works is because [Scanner]\[2\] iterates over tokens in the stream, and in this case we separate tokens using "beginning of the input boundary" (\A) thus giving us only one token for the entire contents of the stream.&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;**Note, if you need to be specific about the input stream’s encoding, you can provide the second argument to \`Scanner\` constructor that indicates what charset to use (e.g. "UTF-8").**&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;Hat tip goes also to [Jacob,]\[3\] who once pointed me to the said article.&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;**EDITED:** Thanks to a suggestion from [Patrick]\[4\], made the function more robust when handling an empty input stream. **One more edit:** nixed try/catch, Patrick’s way is more laconic.&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xD;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;&quot;xA; [1]: https://community.oracle.com/blogs/pat/2004/10/23/stupid-scanner-tricks&amp;xD;&amp;xA; [2]: http://download.oracle.com/javase/8/docs/api/java/util/Scanner.html&amp;xD;&amp;xA; [3]: https://stackoverflow.com/users/68127/jacob-gabrielson&amp;xD;&amp;xA; [4]: https://stackoverflow.com/users/101272/patrick&amp;xD;&amp;xA;&amp;xD;&amp;xA;
Post Block Extraction: Result

**Text block (local id 1)**
Here's a way using only standard Java library (note that the stream is not closed, YMMV).

**Code block (local id 2)**
```java
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\\A");
    return s.hasNext() ? s.next() : "";
}
```

**Text block (local id 3)**
I learned this trick from ["Stupid Scanner tricks"](https://community.oracle.com/blogs/pat/2004/10/23/stupid-scanner-tricks) article. The reason it works is because ...
**Note, if you need to be specific about the input stream's encoding, you can provide the second ...
Hat tip goes also to [Jacob,](https://stackoverflow.com/users/68127/jacob-gabrielson) who once pointed me to the said article.
**EDITED:** Thanks to a suggestion from [Patrick](https://stackoverflow.com/users/101272/patrick), made the function more robust when ...

[2]: http://download.oracle.com/javase/8/docs/api/java/util/Scanner.html
[3]: https://stackoverflow.com/users/68127/jacob-gabrielson
[4]: https://stackoverflow.com/users/101272/patrick
Post Block Extraction: Our Approach

• SO Markdown specs: code blocks **indented by four spaces**
• However, other Markdown conventions and HTML tags are correctly rendered on the SO website
• We used **iterative** approach involving a random **sample of 100,000 SO posts**: 
  
  1. **Check extraction in random subsample**
  2. **Add test case for posts differing from SO website**
  3. **Update extraction method until all test cases pass**
  4. **(Re-)start extraction**
Post Block Extraction: Our Approach

- SO Markdown specs: Code blocks **indented by four spaces**
- However, other Markdown conventions and HTML tags are correctly rendered on the SO website
- We used an iterative approach involving a random sample of 100,000 SO posts:
  - (Re-)run extraction
  - Check extraction for subsample
  - Add test case for posts differing from SO website
  - Update extraction until all test cases pass
Post Block Extraction: Our Approach

• Some observed variants:
  • SO Markdown (indented by 4 spaces)
  • Code-fencing Markdown (enclosed by three backticks)
  • SO Stack Snippets (enclosed by <!--begin/end snippet-->)
  • Stack Snippet language tags (prepended by <!--language: ...-->)
  • HTML code tags (enclosed by <pre><code>)
  • HTML script tags (enclosed by <script>)

https://github.com/sotorrent/posthistory-extractor
2. Post Block Matching
Post Block Matching: Example

https://stackoverflow.com/q/13064858
Post Block Matching: Example

\[ \begin{array}{ccc}
T_1 & \rightarrow & T'_1 \\
C_1 \\
T_2 \\
C_2 \\
T_3 \\
\ldots \\
\end{array} \]

\[ \begin{array}{ccc}
T_1 & \rightarrow & T'_1 \\
C_3 & \rightarrow & \text{new} \\
T_3 & \rightarrow & \text{new} \\
C_1 \\
T_2 \\
C_2 \\
T_3 \\
\ldots \\
\end{array} \]
Post Block Matching: Example

T_1
C_1
T_2
C_2
T_3

T'_1 modified
C_3 new
T_3 new
C_1
T_2
C_2
T_3

...
Post Block Matching: Example

T_1
C_1
T_2
C_2
T_3

T'_1
modified
C_3
new
T_3
new
C_1
T_2
C_2
T_3

...
Post Block Matching: Predecessor Selection Strategy

Algorithm 1 Matching Strategy

for all \( p_{i \leq i \leq n} \) do
  // set predecessors where only one candidate exists
  for all \( b_{(i, j)} \) do
    if \( |\text{Pred}(b_{(i, j)})| = 1 \) then
      Let \( \text{pred} \) be the equal or similar predecessor
      if \( |\text{Succ}(\text{pred})| = 1 \) then
        Set \( \text{pred} \) as predecessor of \( b_{(i, j)} \)
        continue
    end if
  end if
// set predecessors using context
predSet = true
while predSet do
  predSet = setPredContext\( (p_i, \text{BOTH}) \)
end while
while predSet do
  predSet = setPredContext\( (p_i, \text{BELOW}) \)
end while
while predSet do
  predSet = setPredContext\( (p_i, \text{ABOVE}) \)
end while
// set predecessors using position
setPredPosition\( (p_i) \)
end for

- If post block has only one possible successor and this possible successor has only one possible predecessor, no strategy required
- Otherwise, consider the context (successors and predecessors of neighboring post blocks)
- Set remaining post blocks using position (minimal local id distance)
Post Block Matching: Similarity Metrics

- Determine similar post blocks in neighboring post versions
- Must work for all text and code blocks on Stack Overflow
- Implemented 134 variants of generic string similarity metrics

https://github.com/sotorrent/string-similarity

<table>
<thead>
<tr>
<th>Type</th>
<th>Metric</th>
<th>Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>edit</td>
<td>levenshtein</td>
<td>dameraulLevenshtein</td>
</tr>
<tr>
<td></td>
<td>longestCommonSubsequence (LCS)</td>
<td>optimalAlignment (OA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with/without normalization</td>
</tr>
<tr>
<td>set</td>
<td>nGram[Jaccard</td>
<td>Dice</td>
</tr>
<tr>
<td></td>
<td>token[Jaccard</td>
<td>Dice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with/without normalization, padding (nGram)</td>
</tr>
<tr>
<td>profile</td>
<td>cosineNgram[Boolean</td>
<td>TF</td>
</tr>
<tr>
<td></td>
<td>cosineNShingle[Boolean</td>
<td>TF</td>
</tr>
<tr>
<td></td>
<td>cosineToken[Boolean</td>
<td>TF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>nGram : n ∈ {2, 3, 4, 5}, nShingle : n ∈ {2, 3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with normalization (both) and without (cosine)</td>
</tr>
<tr>
<td>fingerprint</td>
<td>winnowingNGram[Jaccard</td>
<td>Dice</td>
</tr>
<tr>
<td>equal</td>
<td>equal</td>
<td>tokenEqual</td>
</tr>
</tbody>
</table>
Post Block Matching: Ground Truth

• To **evaluate the metrics** regarding their suitability for comparing post blocks, we needed a ground truth

• Implemented **tool** to create **manually validated** version histories

• Used tool to create manually validated post version histories for **500 posts**

https://github.com/sotorrent/posthistory-gt
Post Block Matching: Metrics Evaluation Results

**manhattanFourGramNormalized (Text)**

<table>
<thead>
<tr>
<th>False positive rate</th>
<th>True positive rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>0.15</td>
<td>0.2</td>
</tr>
<tr>
<td>0.2</td>
<td>0.25</td>
</tr>
<tr>
<td>0.25</td>
<td>0.3</td>
</tr>
<tr>
<td>0.3</td>
<td>0.35</td>
</tr>
<tr>
<td>0.35</td>
<td>0.65</td>
</tr>
<tr>
<td>0.65</td>
<td>0.7</td>
</tr>
<tr>
<td>0.7</td>
<td>0.75</td>
</tr>
<tr>
<td>0.75</td>
<td>0.8</td>
</tr>
<tr>
<td>0.8</td>
<td>0.85</td>
</tr>
<tr>
<td>0.85</td>
<td>0.9</td>
</tr>
<tr>
<td>0.9</td>
<td>0.95</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
</tbody>
</table>

Threshold: 0.17
TPR: 0.99
FPR: 0.14
Matthews Corr.: 0.86

**winnowingFourGramDiceNormalized (Code)**

<table>
<thead>
<tr>
<th>False positive rate</th>
<th>True positive rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>0.05</td>
<td>0.1</td>
</tr>
<tr>
<td>0.1</td>
<td>0.15</td>
</tr>
<tr>
<td>0.15</td>
<td>0.2</td>
</tr>
<tr>
<td>0.2</td>
<td>0.25</td>
</tr>
<tr>
<td>0.25</td>
<td>0.3</td>
</tr>
<tr>
<td>0.3</td>
<td>0.35</td>
</tr>
<tr>
<td>0.35</td>
<td>0.65</td>
</tr>
<tr>
<td>0.65</td>
<td>0.7</td>
</tr>
<tr>
<td>0.7</td>
<td>0.75</td>
</tr>
<tr>
<td>0.75</td>
<td>0.8</td>
</tr>
<tr>
<td>0.8</td>
<td>0.85</td>
</tr>
<tr>
<td>0.85</td>
<td>0.9</td>
</tr>
<tr>
<td>0.9</td>
<td>0.95</td>
</tr>
<tr>
<td>0.95</td>
<td>1</td>
</tr>
</tbody>
</table>

Threshold: 0.23
TPR: 0.99
FPR: 0.07
Matthews Corr.: 0.92
Link Extraction
Extracting Links From Stack Overflow Posts

• Extracted 31.4m links from 11.6m posts, pointing to 567k different domains using a regular expression (SOTorrent 2018-05-04)

• Extracted 6.0m links from 438k GitHub repos, pointing to 147k posts using Google BigQuery (SOTorrent 2018-05-04)
Retrieve links from a post version:

```
SELECT PostId, PostHistoryId, Domain, Url
FROM PostVersionUrl
WHERE PostHistoryId = 155295527;
```
Retrieve links from GitHub repos to post:

```
SELECT PostId, RepoName, Branch, Path, FileExt, Size, Copies
FROM PostReferenceGH
WHERE PostId=5445161;
```

![Table showing referenced repos]

Referenced in 103 distinct repos
First Results
First Results

• Analysis based on SOTorrent 2018-03-28
• Posts **grow over time** in terms of number of post blocks, but the size of individual blocks is relatively **stable**
• In only 6% of all edits, **code blocks** were changed without also updating **text blocks**
• 78% of all edits made on the **same day** the post was created
• 87% of edits made by **post authors**
• **Comments and edits are closely related**: Edited posts have significantly more comments and commented posts have significantly more edits, conducted qualitative analysis
Next Steps

• Study SO **post editing** in detail, e.g.
  • Analyze communication structures/edit patterns, in particular how **comments** are used for **feedback**
  • Compare **edits** shortly after post **creation** to **later** edits
  • Identify typical **moderator changes**

• Identify **bug-fixing edits** and
  • search for buggy snippet versions in GitHub repos
  • link them to comments reporting the bug

• **Improve the dataset** (e.g., detect non-code code blocks and bot edits, improve post block matching)
Summary

Post Block Extraction
Post Block Matching
Link Extraction
First Results

sotorrent.org
Dataset available on Zenodo and BigQuery