Summary Review: SOTorrent: Reconstructing and analyzing the evolution Stack Overflow posts,

Christina Li
University of Waterloo
Waterloo, Canada
christina.li1@uwaterloo.ca

1 What is the Problem Being Solved?

The paper seeks to understand how posts on Stack Overflow (SO) evolve over time, including changes in text and code blocks. The paper explores three research questions: How do Stack Overflow posts evolve? (RQ1), which posts get edited? (RQ2) and what is the temporal relationship between edits and comments? (RQ3). Current SO data dumps provide limited insights into version histories at a granular level, restricts the ability to analyze how individual text or code elements change. This hinders researchers' and practitioners' ability to study the quality and maintainability of knowledge shared on SO, especially the widely used code snippets.

2 What is the New Idea they are proposing?

The authors introduce SOTorrent, an open dataset designed to analyze the version history of SO posts at both the post and block levels. Key features and contributions of SOTorrent include:

- Fine-grained Version History Reconstruction: SOTorrent extracts version histories for individual text and code blocks, allowing detailed analysis of how these components evolve over time. The dataset links different versions of a post and tracks changes using algorithms to evaluate 134 string similarity metrics to determine relationships between blocks in different versions.
- Linking SO Posts to External Resources: SOTorrent connects SO posts to GitHub repositories and other external URLs by analyzing references in text blocks and source code, providing a more comprehensive ground to understand how SO evolve overtime. This enables researchers to study how SO knowledge influences or is influenced by other platforms.
- Insights into SO Post Evolution: The authors analyze the dataset to answer key research questions, such as how posts evolve, which posts are edited, and the temporal relationship between edits and comments. For example, they find that post edits are often triggered by comments and that code changes are typically accompanied by updates to the surrounding text.
- Dataset Availability and Tools: The authors make SOTorrent available on Zenodo, alongside with scripts for data processing and analysis, encouraging further research.

3 Positive Points

- **Comprehensive Dataset:** SOTorrent provides granular data on post and block evolution, which is valuable for understanding how knowledge in SO evolves. The connection with GitHub adds another layer of utility for studying the flow of knowledge between platforms.
- **Robust Methodology:** Rather than relying on a single approach, the authors rigorously test metrics from five categories, including edit-based, set-based, profile-based, fingerprint-based, and equality-based methods. This comprehensive analysis identifies Manhattan Four-Gram Normalized as the best metric for text and Winnowing Four-Gram Dice Normalized for code.
  To ensure accuracy, the study employs a three-stage evaluation process, testing individual metrics, refining similarity thresholds, and optimizing metric combinations for text and code. The approach is validated with 600 manually reviewed posts and assessed using the Matthews Correlation Coefficient, which provides a more balanced measure than standard precision-recall. This rigorous methodology ensures high accuracy and reliability, making SOTorrent a strong foundation for studying post-evolution in developer communities.
- **Actionable Insights:** The paper offers practical insights, such as the importance of comments in driving post edits and the co-evolution of text and code, which are relevant for improving SO's usability and moderation practices.

4 Negative Points

- **Lack of a Clear Thematic Structure** The related work section covers various topics (e.g., SO knowledge reuse, API documentation, user behavior, source code similarity, and software evolution), but the organization could be more structured. It jumps between themes without clear transitions, making it harder to follow how each reference contributes to the context of the research.
- **Minimal Discussion of Limitations in Prior Work** The paper does not fully analyze the limitations of existing approaches. It lists studies on SO post evolution, API documentation, and source code reuse but does not critically assess what those studies lacked and how SOTorrent addresses these gaps. A more explicit discussion of why existing datasets or tools are insufficient would strengthen the justification for SOTorrent.
- **Limited Coverage of Long-Term Evolution Trends:** While the data set provides a snapshot of post-evolution, the article could delve deeper into long-term trends, such as the impact of edits on the quality or relevance of posts.
- **Focused on Technical Details:** The paper emphasizes the technical implementation of SOTorrent and its evaluation, but provides relatively less discussion on how the data set can be applied to practical problems, such as improving the SO user experience or API documentation.

5 Future Works

Future research could use SOTorrent to analyze code snippet quality and maintenance, focusing on how code evolves in terms of security, readability, and maintainability. By tracking changes over time, researchers could identify patterns in which types of edit improve code robustness and whether developers prioritize bug fixes, performance optimizations, or style refinements. This could lead to automated recommendations for improving code quality on Stack Overflow, ensuring that frequently reused snippets adhere to best-practices and security guidelines.

Another possible direction could be cross-platform analysis, particularly in understanding how knowledge flows between SO and GitHub. Many developers copy and modify SO code in open source projects, but the impact of these snippets on project outcomes, bug reports, and long-term maintainability remains underexplored. By studying how frequently SO code is adopted, modified, or abandoned, researchers could gain insights into best practices for knowledge sharing between developer communities. This could also help in designing better linking mechanisms between SO and GitHub, improving traceability and citation practices for open-source contributions.

6 Rating

4/5: The paper introduces a valuable dataset and robust methodology, making significant contributions to the study of SO post-evolution. However, the related work section is not very well written, and a deeper exploration of real-world applications and long-term trends would strengthen its research aim.

7 Discussion

7.1 How can SOTorrent be used to improve SO's moderation and recommendation systems, such as identifying posts most likely to need edits or updates?

One key issue raised was that SO recommendations rely primarily on upvotes and checkmarks, which means older posts that were once highly rated may no longer be relevant due to evolving programming languages or best practices. Students suggested that SO's recommendation algorithm should place greater weight on newer interactions, such as recent comments, edits, and external references. However, there was concern that if search engines prioritize last edit dates, users might inadvertently ignore older but still valid posts. A student brought up programming language C as an example which does not evolve very much over time, whereas Professor Godfrey mentioned C++, a language that undergoes frequent changes, suggesting older solutions could potentially be misleading.

7.2 How can we determine whether post edits actually improve content, rather than just making superficial changes?

It was noted that some edits may be purely superficial and that some comments are unhelpful in terms of improving the quality of the post, making it hard to assess the true impact of an edit. One of the challenges discussed was that testing whether an edit improves the code is difficult, as it requires running the code in various contexts, making large-scale validation impractical. Instead, alternative approaches were suggested, including analyzing comments that appear after an edit to check whether they validate or critique the changes. Another idea was to log whether checkmarks appear after an edit, which could indicate whether an update actually improves the quality of a post.

7.3 Are there ethical concerns in tracking and analyzing developer interactions across different platforms?

Although this discuss point was mentioned, it was not properly discussed which left room for future in-depth discussion when more paper are introduced.

7.4 Other Points Discussed

Beyond these above topics, students also discussed the role of AI and LLMs in the light of SO's future. While SOTorrent offers structured insights into post-evolution, the growing use of LLMs like ChatGPT could change how developers search for and consume programming knowledge. Some students noted that Google searches which lead to SO, remain faster and more efficient than interacting with verbose LLM responses, highlighting the importance of SO and researches around SO.

Additionally, ideas for new tools and platform features were proposed. One suggestion was a VSCode extension that logs the execution of SO snippets within real projects, providing implicit feedback on snippet usability. Another proposal focused on reducing SO's notorious duplicate post problem by grouping related questions under a single "main" post, similar to Quora's model, which would improve the contents' discoverability and avoid fragmentation.