October 2023

User-centric Test Sequence Generation for Fault Detection

Kun-Da Wu

Berlin Hsin

User-centric Test Sequence Generation for Fault Detection

ABSTRACT

Test engineering, including development of new test cases from large volumes of log data, can be expensive. User reports often do not include clear description of the context of poor user experience. The effectiveness of a test case derived from user reports is hard to validate. This disclosure describes the use of machine learning techniques such as attention based neural networks, language models, etc. to generate from user reports realistic, multi-faceted test sequences designed to trigger failures. A pattern extractor extracts critical events relevant to particular failures from session logs. A test sequence generator generates user-like action sequences or event sequences based on patterns extracted by the pattern extractor. A test sequence validator determines if the generated test sequences are close to user-like behavior, and likely to trigger the expected failures.

KEYWORDS

- Test case
- Software testing
- Test sequence
- Attention based neural network
- Language model
- Fault detection
- User report
- Critical user journey (CUJ)
- Recurrent neural network
- Attention model

BACKGROUND

To ensure the quality of consumer devices such as smartphones, engineers investigate user-reported issues by creating test cases designed to trap the particular issue. Designing a test case often requires manually exploring large volumes of log data to identify symptoms of failures which is a time-consuming task, especially when the user's report lacks clarity.

Test engineering generally requires substantial investments in hardware and personnel. New test cases can be particularly expensive. Moreover, device users often find it difficult to clearly describe the context of their poor user experience or device failures. The effectiveness of a test case derived from user reports is hard to a priori validate.

Codeless test automation can reduce the effort of writing test-case code but still requires manual effort to describe the test sequence. Also, the generated test sequences may be ones that neither cause failures nor are used naturally by users.

DESCRIPTION

This disclosure describes techniques to generate realistic multi-faceted test sequences designed to trigger failures given the hardware device model, software system version, particular types of failures, etc. Components of the techniques include:

Pattern extractor: The pattern extractor extracts critical events relevant to particular failures from large volumes of session logs. The pattern extractor can be, for example, an attention-based recurrent neural network model.

Test sequence generator: The test sequence generator generates user-like action sequences or event sequences based on patterns extracted by the pattern extractor. The test sequence generator can be, for example, a language model. Given the expected sequence length sequence and the order of critical events, the test sequence generator iterates to generate realistic sequences that lead to expected failures with high confidence.

Test sequence validator: The test sequence validator determines if the generated test sequences are close to user-like behavior, and whether they are likely to trigger the expected failures. The test sequence validator can be an attention-based recurrent neural network. The input to the test

sequence validator is a sequence of multi-faceted data. The output is a score that indicates a degree of realisticness of the sequence, and also, a likelihood of the sequence triggering the expected failures.

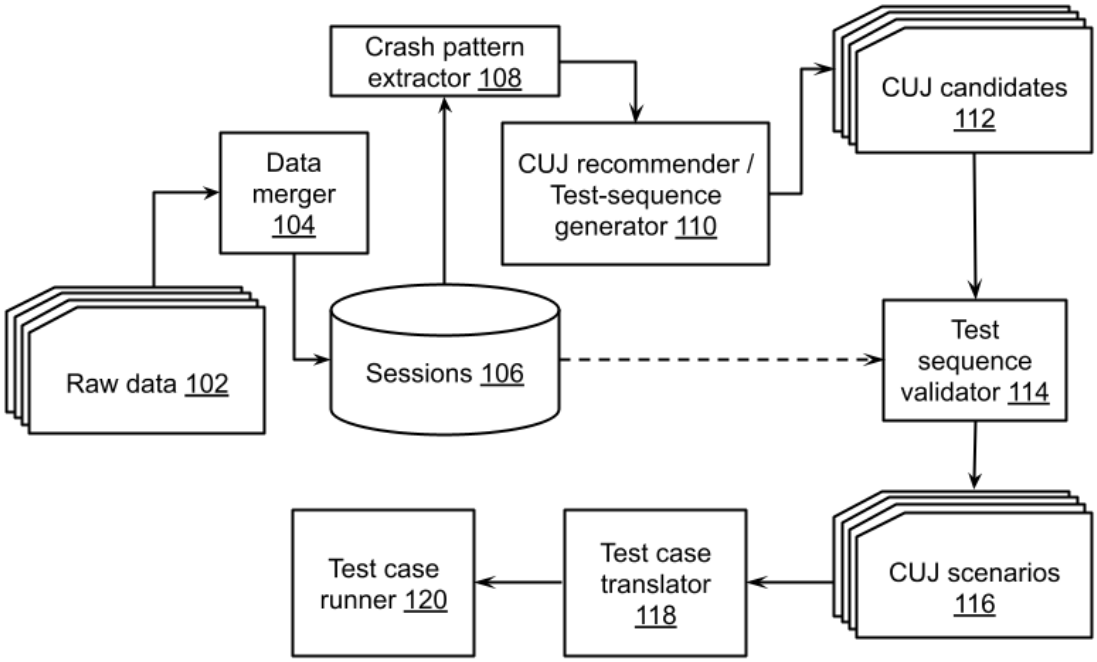

Fig. 1: User-centric test sequence generation for fault detection

Fig. 1 illustrates user-centric test sequence generation for fault detection, per techniques of this disclosure. Raw log data (102) that originates from various sources and can be in differing formats, is merged in sessions (106) by a data merger (104). A crash pattern extractor (108) extracts crash patterns from the merged data. The pattern extractor can be, for example, an attention-based recurrent neural network model.

A test-sequence generator (110), also referred to as a critical user journey (CUJ) recommender, generates simulated candidate CUJs (112) based on user reports. For example, the test-sequence generator / CUJ recommender can be a language model. The expected sequence length sequence and the order of critical events as input to the language model. The test-

sequence generator/ CUJ recommender iterates to generate realistic CUJ candidates that lead to the expected failures with high confidence. Of the generated CUJ candidates, a test sequence validator (114) determines CUJ scenarios (116) that mimic user behavior and that are likely to trigger the expected failures. For example, the test sequence validator can be an attention-based recurrent neural network. The CUJ scenarios are translated to test scenario API calls by a test case translator (118). A test case runner (120) runs the test on test infrastructure that hosts the devices-under-test.

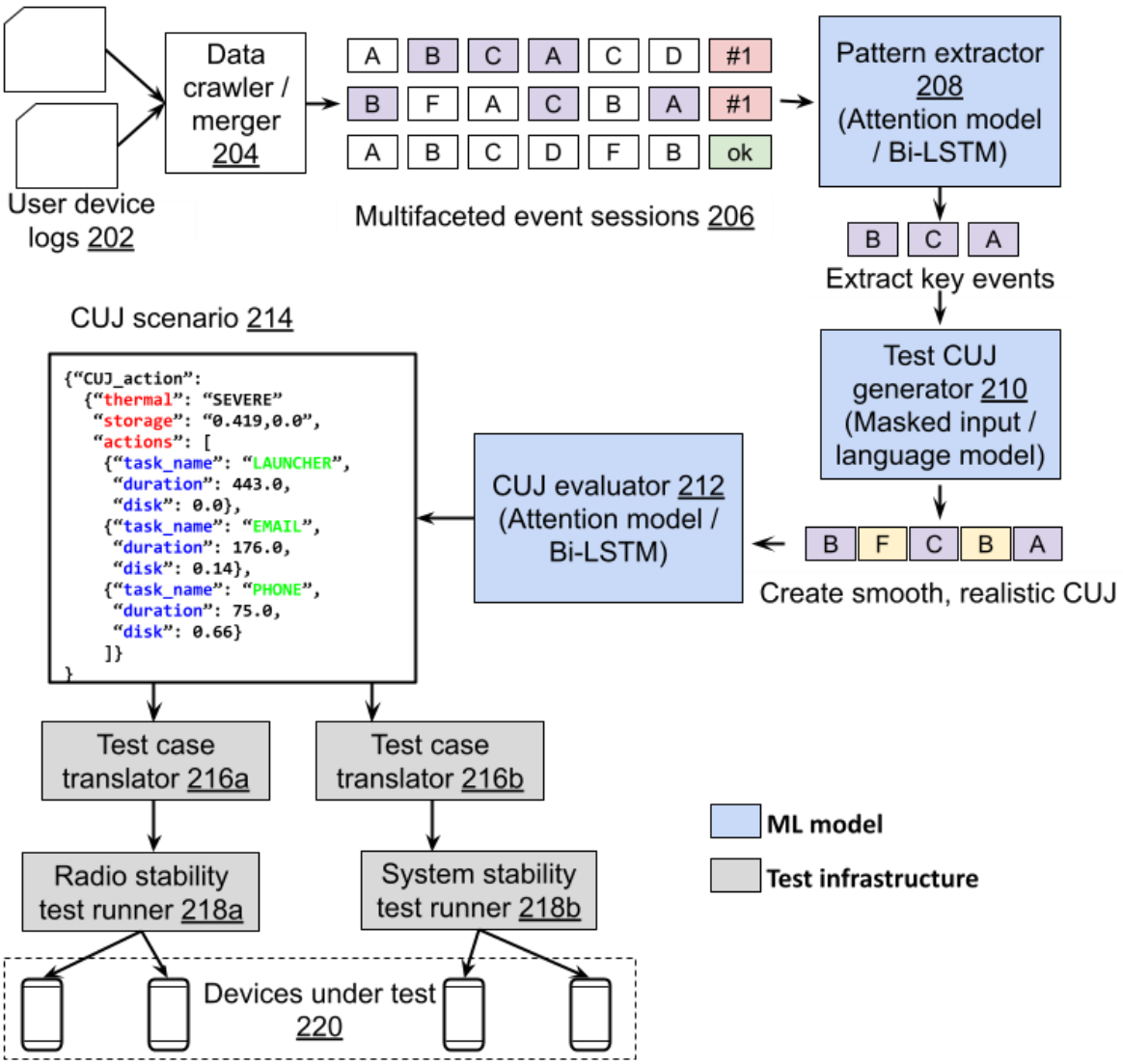

Fig. 2: User-centric test sequence generation for fault detection

Fig. 2 illustrates in greater detail user-centric test sequence generation for fault detection, highlighting in particular the creation of smooth, realistic CUJs from key events extracted from multifaceted event sessions. A data crawler/merger (204) merges logs (202) (e.g., obtained with permission user devices) and produces multifaceted event sessions (206). Some event sessions result in failures of a particular type (#1, in red), whereas others result in faultless operation ('ok,' in green).

A pattern extractor (208), which can be an attention model, e.g., a bidirectional long short-term memory (LSTM) recurrent neural network, extracts key events ('B-C-A') from the multifaceted event sessions. A test CUJ generator or recommender (210), which can be a masked-input language model, creates smooth, realistic CUJs ('B-F-C-B-A') from the key events. A CUJ evaluator or validator (212), which can be an attention model, e.g., a bidirectional long short-term memory (LSTM) recurrent neural network, generates CUJ scenarios (214). The scenarios are translated by test case translators (216a-b) into test runners (218a-b) that test various aspects, e.g., radio stability, system stability, etc., of devices under test (220).

CONCLUSION

This disclosure describes the use of machine learning techniques such as attention based neural networks, language models, etc. to generate from user reports realistic, multi-faceted test sequences designed to trigger failures. A pattern extractor extracts critical events relevant to particular failures from session logs. A test sequence generator generates user-like action sequences or event sequences based on patterns extracted by the pattern extractor. A test sequence validator determines if the generated test sequences are close to user-like behavior, and likely to trigger the expected failures.

REFERENCES

1. "Deliver quality software at devops speed." available online at https://www.tricentis.com accessed Oct. 3, 2023.
2. "State of AI applied to quality engineering 2021-22. From logs to tests: usage-driven regression testing." available online at https://www.sogeti.com/ai-for-qe/section-8-operate/chapter-3/ accessed Oct. 3, 2023.
3. Khaliq, Zubair, Sheikh Umar Farooq, and Dawood Ashraf Khan. "Artificial intelligence in software testing: Impact, problems, challenges and prospects." *arXiv preprint arXiv:2201.05371* (2022).