NLP Selection of Tests - ...

Marius Bosler

Seminar: Software Quality
Advisor: Roland Würsching
Technical University of Munich
marius.bosler@tum.de

Abstract. Even though there have been many advancements in recent years, a significant amount of testing still needs to be conducted manually. This means that with each software release requiring numerous manual tests, a considerable amount of time is spent executing these tests. Meanwhile, many of these tests tend to overlap in bug detection, which could be avoided if a method were found to execute only the necessary number of tests, as multiple tests may detect the same bug. One way to achieve this is by clustering them through an analysis of the language used in their description. Then, by carefully selecting only a subset of all tests based on said clustering, a lot of time can be saved, especially if the selected tests cover a significant portion of the detected bugs.

Keywords: Test Selection, Natural Language Processing, Clustering

1 Introduction

. . .
The entire process was implemented in Python, and the code for this project is available on GitHub. (link hier noch einfügen)

2 Related Work

Li et al. [1] has proposed a method, [2] another and [3] yet another

3 Methodology

3.1 Data Collection and Description

We used data from XWiki, an open-source wiki platform. From their website, we scraped a dataset of 1,600 manual test cases and 52,860 test executions. Each test case is structured with a description, title, steps to reproduce, and expected results. For instance, a typical test case might look like this:

```
"https://test.xwiki.org/xwiki/bin/view/File%20Manager%20Tests/Delete%20a%20file": {
    "title": "Delete a file",
    "steps_to_reproduce": [
        "Click on File Manager from Applications Panel",
        "Click on All Files",
        "Select a file",
        "Click on Delete",
        "Click on OK"
    ],
    "expected_result": [
        "The file is deleted."
    ]
}
```

The dataset chosen for this study is especially fitting for us, because it contains detailed textual descriptions in the form of test steps, which can tell us how similar the given tests are, which we later want to use to cluster them together.

3.2 Natural language processing (NLP) Techniques and Test Selection Process

First, we converted all words to lowercase and used word stemming, with the python library *nltk*. For example, this converts the sentence *Click on File Manager from Applications Panel* to important word stems: *click file manag applic panel.*
This is done, because it makes text analysis more efficient later, especially if we only want to extract the meaning and similarity of test steps.

We then converted the resulting strings to their embeddings with the Python library *SentenceTransformer* and the model *all-mpnet-base-v2.* Embeddings can be thought of as representations of textual information in a numerical format. This is really important, because they enable us to compare similarities between different elements efficiently in a mathematical way. Every sentence is now represented as a vector of numbers with length 768:

```
[-8.30066670e-03, -4.04462852e-02, -2.05343179e-02,  3.46409194e-02,

⋮

1.46980640e-02,  3.09091341e-02,  3.18837278e-02,  3.14172916e-02]
```

To then find similarities between test cases, we needed to find a way to find out which tests have similar test steps. This is done in the following way: First, we are clustering the embeddings of all test steps, so every test case has a number of clusters its test steps are in. As a simplified example, we might sort them into 10 clusters, then our example test case steps might be in the clusters with ids *4, 2, 2, 6, 7* (the results are better if we use much more than 10 clusters). For the actual clustering of test cases, we've employed three distinct algorithms: KMeans, DBScan, and Optics. After some evaluation, it became clear that KMeans was the most fitting choice for our specific needs, as we later want to choose the n dissimmilar test cases and KMeans lets us define how many test clusters we want. Then we need to find similarities between actual test cases: For every test case, we create a empty null-vector with the length of the number of different clusters. Then we fill it at the positions of said cluster ids with how many each cluster appears:

```
[0, 0, 2, 0, 1, 0, 1, 1, 0, 0]
```

These Vectors now represent our tests in a mathematical way, which, as mentioned earlier, we want to compare the tests with other tests. To achieve this, we cluster again after these vectors and as we want to select n dissimmilar test cases, we cluster with KMeans to cluster the tests into n clusters.

After that, we have n clusters of test cases from which we then select one test each.

4 Evaluation

4.1 Research Questions

To test our approach, we want to use our dataset of XWiki to answer the following Research Questions (RQs):

- RQ1: *How many bug tickets are found when selecting only a subset of tests using NLP-Clustering by Test Specification* The goal of this paper is to demonstrate wether clustering tests in said method can produce helpful results. But is there actually a substantial amount of time/tests that can be saved when only selecting a portion of all tests?
- RQ2: *How does the performance of NLP-based test selection compare with simpler, more naive methods of test selection in terms of bug detection and resource utilization?* This is done to show wether it is even worth it to implement something comparatively complex over something simple.

4.2 Evaluation Metrics

To evaluate how well our algorithm works, we want to find out how many bugs are covered by a certain percentage of run tests. Our source also includes bug tickets and we can link each test to its bug ticket. That way, we can go through all n selected tests and make a set of bug tickets covered and divide them by the number of all bug tickets and thereby know the percentage of bug tickets covered. As we do not yet know how many tests should be selected, we will go through all the percentage numbers from 10 to 70 in steps of ten and see how many bug tickets have been covered by each number of tests.

4.3 Comparison of Approaches

To then find out how our algorithm compares to naive approaches we first have to choose a naive approach. The first proposition that comes to mind is just choosing n random tests from our test cases. Our alternative approach is only possible with data that has some sort of categorization. As an example, the test at

`https://test.xwiki.org/xwiki/bin/view/File%20Manager%20Tests/Delete%20a%20file`

we mentioned earlier, is in the category *P1.Extensions - File Manager Tests*. In total there are TODO categories like this in our data set. As we can assume similar bugs belong to similar test categories, this implementation might lead to a well enough grouping of test cases.

4.4 Results

We compared the three methods of test selection to evaluate their effectiveness in covering bug tickets. These methods are represented in Table 1 as:

- Method 1, selecting test cases randomly from the entire dataset
- Method 2, selecting test cases randomly from each test category
- Method 3, being our presented approach with selecting tests by clustering test steps.

Percentage of Tests Run	Method 1	Method 2	Method 3
10%	14.22%	15.14%	18.81%
20%	26.15%	27.75%	36.24%
30%	37.84%	38.53%	51.61%
40%	47.25%	47.94%	59.86%
50%	52.98%	54.82%	72.48%
60%	69.95%	68.35%	80.96%
70%	73.62%	70.87%	83.03%

Table 1. Coverage of Bug Tickets by Percentage of Tests Run for Different Methods

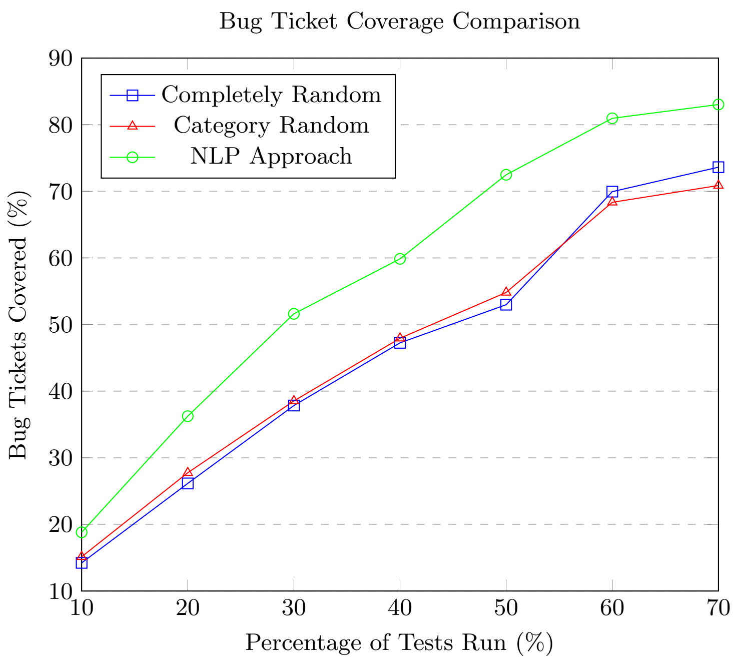

Fig. 1. Comparison of different test selection approaches in terms of bug tickets covered.

5 Discussion

Especially for larger percentages, clustering takes quite a substantial amount of time -¿ worth it?

6 Threats to Validity

7 Conclusion

You can also reference other parts of the document, e.g., sections or subsections. In Section **??** we briefly introduced something, whereas in Subsection **??**, we motivated something else.

Make sure to capitalize chapters, sections or subsections when referencing them.

References

1. Li, L., Li, Z., Zhang, W., Zhou, J., Wang, P., Wu, J., He, G., Zeng, X., Deng, Y., Xie, T.: Clustering test steps in natural language toward automating test automation. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. p. 1285–1295. ESEC/FSE 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3368089.3417067, `https://doi.org/10.1145/3368089.3417067`
2. Sutar, S., Kumar, R., Pai, S., BR, S.: Regression test cases selection using natural language processing. In: 2020 International Conference on Intelligent Engineering and Management (ICIEM). pp. 301–305 (2020). https://doi.org/10.1109/ICIEM48762.2020.9160225
3. Viggiato, M., Paas, D., Buzon, C., Bezemer, C.P.: Identifying similar test cases that are specified in natural language. IEEE Transactions on Software Engineering **49**(3), 1027–1043 (2023). https://doi.org/10.1109/TSE.2022.3170272