Performance Comparison of Two Deep Learning Algorithms in Detecting Similarities between Manual Integration Test Cases

C. Landin, L. Hatvani, S. Tahvili, H. Haggren, M. Längkvist, A. Loutfi and A. Håkansson

cristina.landin@ericsson.se

cristina.landin@oru.se

PTD – PDU Ericsson AB

Örebro University Sweden

About me

- Working at Ericsson AB for 6 years as RF Integration Engineer
- 2nd year Ph.D. student in Computer Science at Örebro University, AASS MPI group
- Electronic Engineer background, specialized in Telecommunications
- Passionate about emerging technologies and robotics

My work:

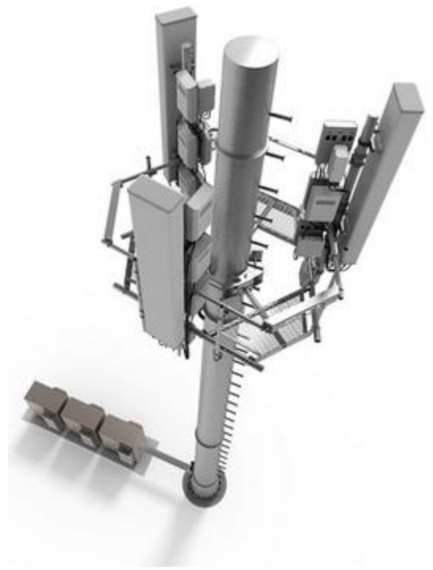
- My research area is about optimizing the test process at the production of telecommunication products. The test optimization can be done by different methods, in the beginning of my studies we focused on NL describing manual integration test cases and the clustering of these test cases by their semantic similarities in order to find redundant test cases which ground the base for test suite minimization

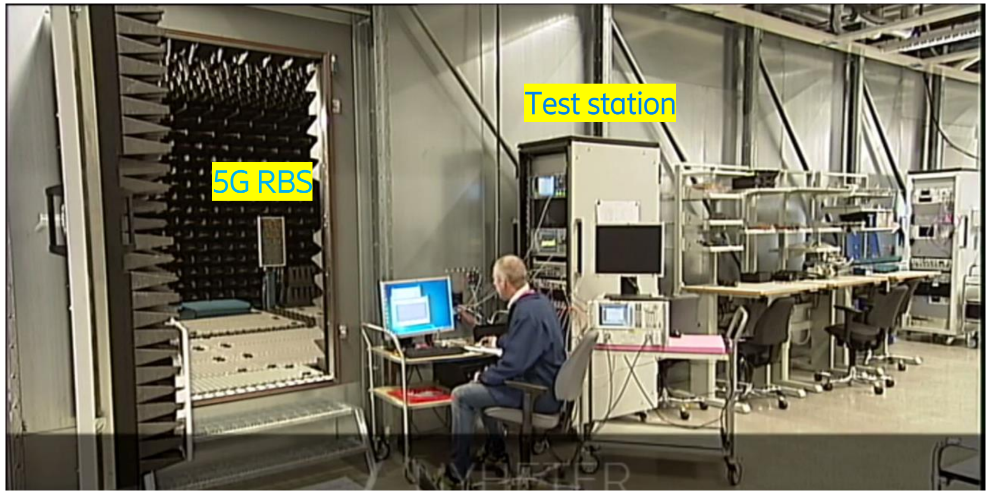
Outline

How and what do we test?

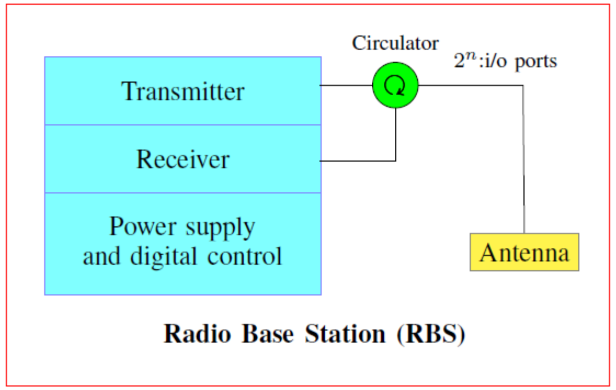
Problem statement

- Radio Base Station (RBS) at production test
- Test specifications in a non-formal natural language (NL)
- Manual test generation from the test specifications
- A large set of test cases are generated for testing each RBS
- Missing gap between the test case and test-code generation
- AI for test optimization (analyze large amount of data)


4G RBS



Test production during test code generation and test execution



RBS and its simplified block diagram, 2n

Background

Test station: HW interface between the user and the RBS

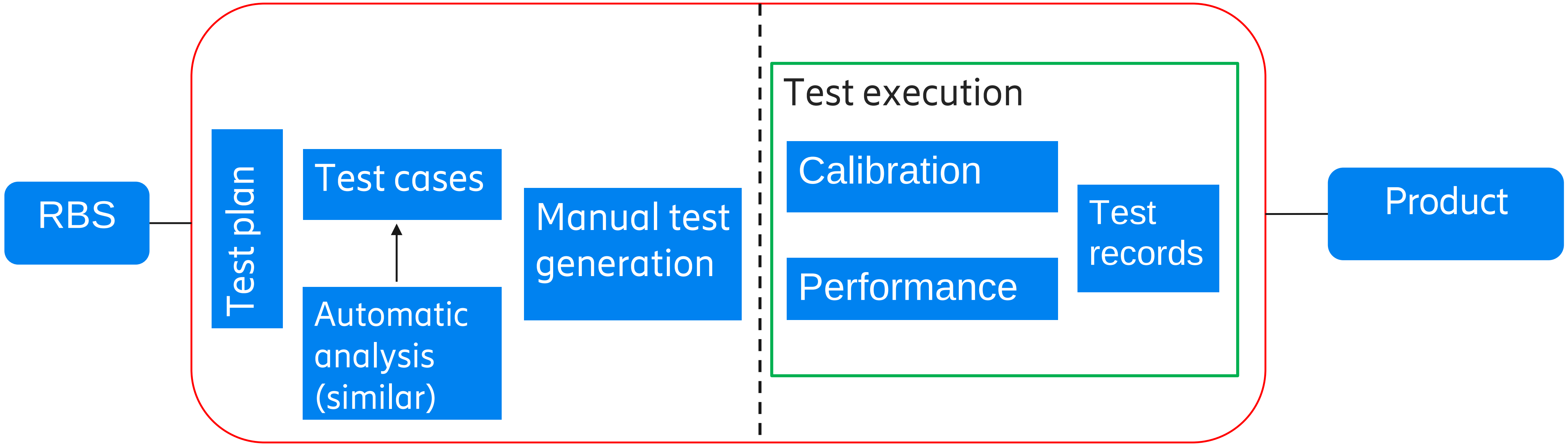


Figure 1. Block diagram of the test process of an RBS

Test code - generation

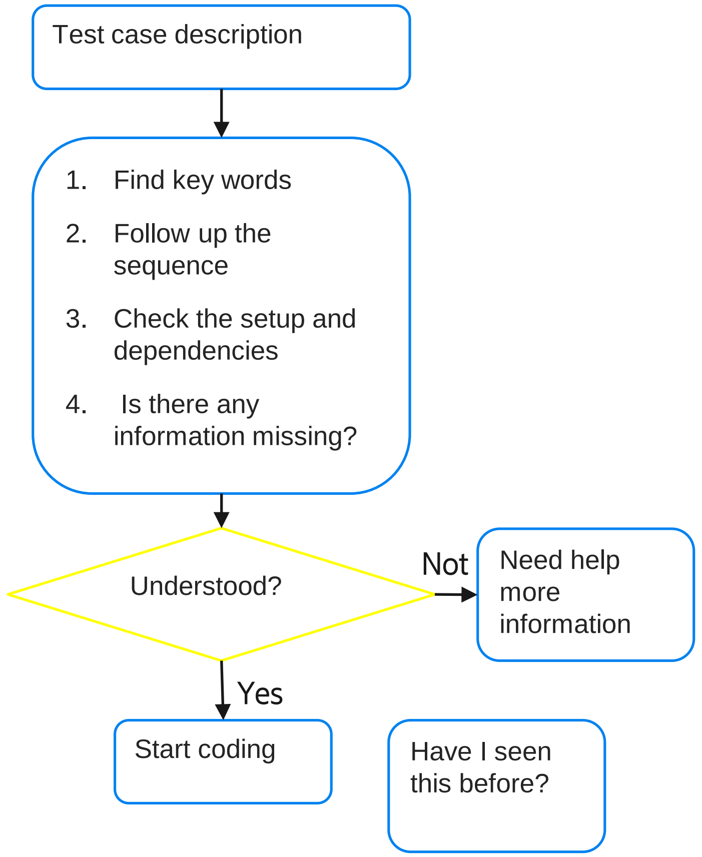

Test case description
1. Find key words
2. Follow up the sequence
3. Check the setup and dependencies
4. Is there any information missing?
Understood?
Not
Need help more information
Yes
Start coding
Have I seen this before?

Fig.2 Test flow diagram – time consuming and ambiguity

How do we plan to solve this problem?

Proposed approach

- Test optimization by parallel test execution of similar TCs

Definition 1. *Test case TC_1 and TC_2 are similar if they are designed to test the same functionality or they have the same preconditions, execution requirements (installation), system setup.*

- Use Deep Learning algorithms to find semantic similarities between TCs descriptions
- Sequential test execution, considering all variables:

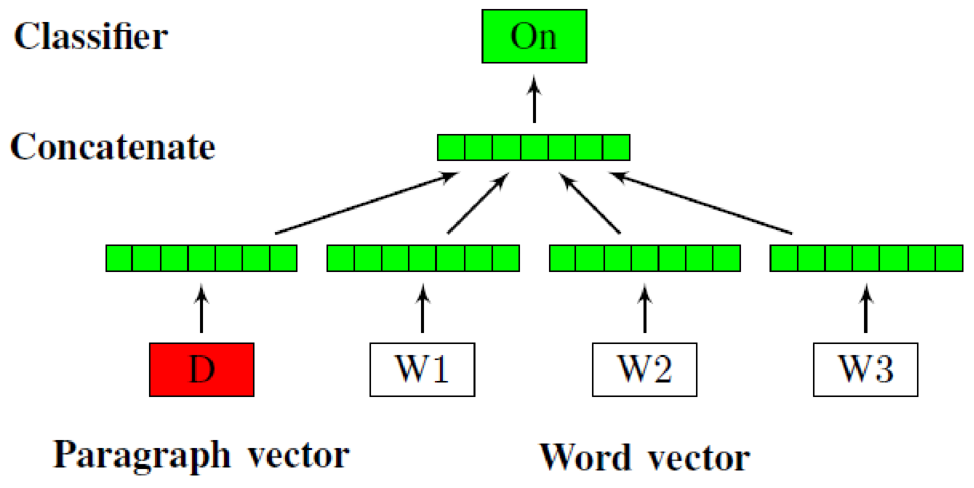
$$T = h * 2^n * \sum_{i=1}^{m} t_i \qquad (1)$$

- Save time by parallel test execution (setup), explained in our first paper:

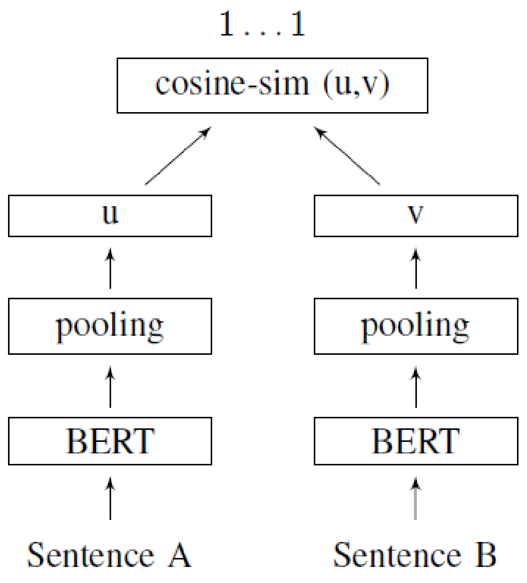
$$ST = \frac{(np - 1)}{np} \times 100 \qquad (2)$$

Proposed approach

- Deep learning algorithms:



(a)

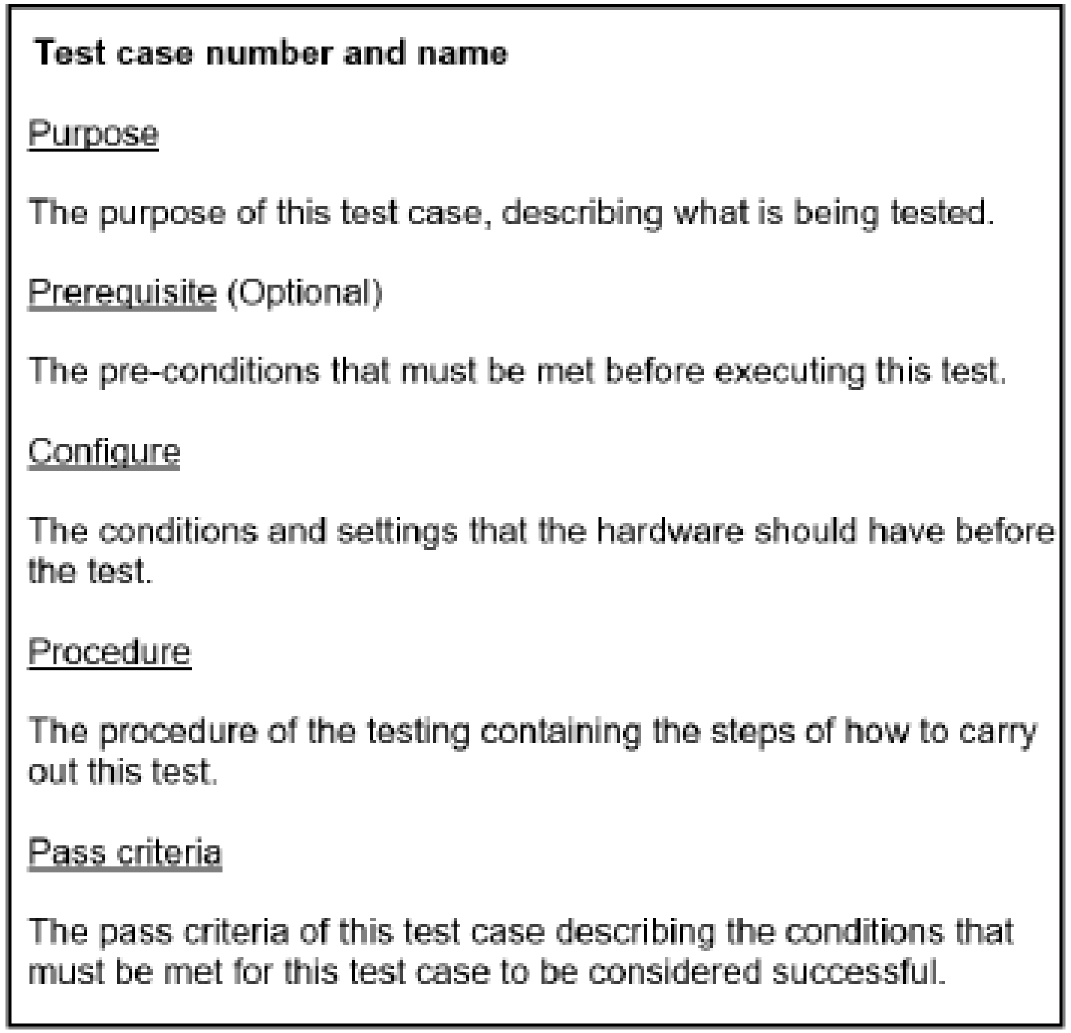


(b)

Fig.3 (a) Doc2vec and (b) SBERT on finding semantic similarities of text

Dataset

- Test requirements for five 4G RBSs
- Test case number and name follow a pre-defined internal standards
- Procedure can have different test case steps which are translated as test points
- The test case contains different lengths of text, from 2 lines to many pages and the description is very technical
- 444 test cases



Test case number and name

Purpose

The purpose of this test case, describing what is being tested.

Prerequisite (Optional)

The pre-conditions that must be met before executing this test.

Configure

The conditions and settings that the hardware should have before the test.

Procedure

The procedure of the testing containing the steps of how to carry out this test.

Pass criteria

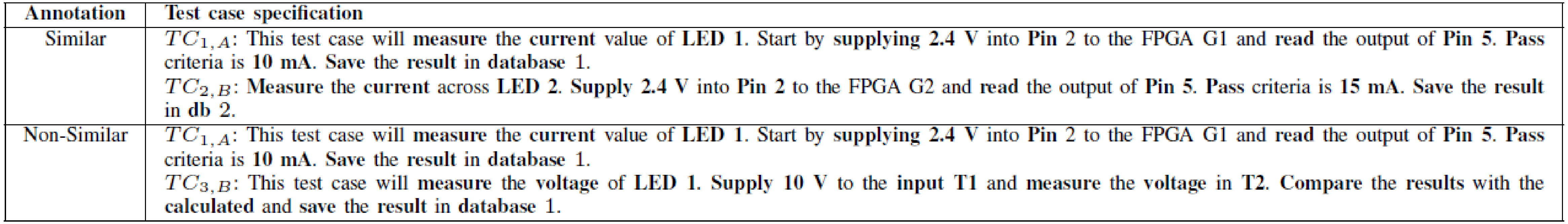
The pass criteria of this test case describing the conditions that must be met for this test case to be considered successful.

Fig.4 A typical test case description within Ericsson [1)]

[1)] Source: Test case generation from specifications using NLP, Master thesis in Computer Science and Engineering, 2020, Alzahraa Salman

Dataset

- All TCs are grouped according their semantic similarities, similar descriptions
- Test case example represents the test case procedure of each test case. LED (Light Emitting Diode)

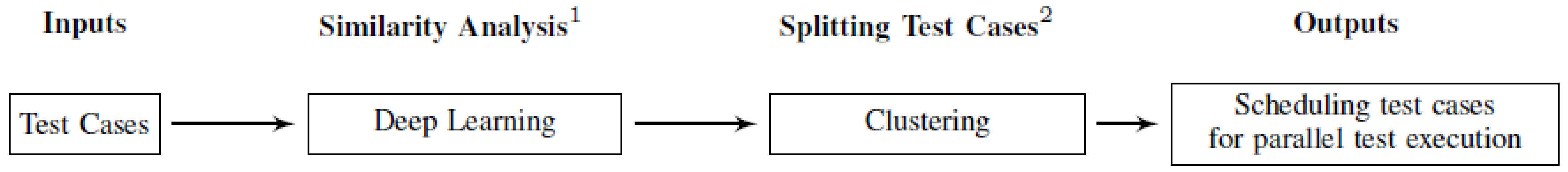


Annotation	Test case specification
Similar	$TC_{1,A}$: This test case will measure the current value of LED 1. Start by supplying 2.4 V into Pin 2 to the FPGA G1 and read the output of Pin 5. Pass criteria is 10 mA. Save the result in database 1. $TC_{2,B}$: Measure the current across LED 2. Supply 2.4 V into Pin 2 to the FPGA G2 and read the output of Pin 5. Pass criteria is 15 mA. Save the result in db 2.
Non-Similar	$TC_{1,A}$: This test case will measure the current value of LED 1. Start by supplying 2.4 V into Pin 2 to the FPGA G1 and read the output of Pin 5. Pass criteria is 10 mA. Save the result in database 1. $TC_{3,B}$: This test case will measure the voltage of LED 1. Supply 10 V to the input T1 and measure the voltage in T2. Compare the results with the calculated and save the result in database 1.

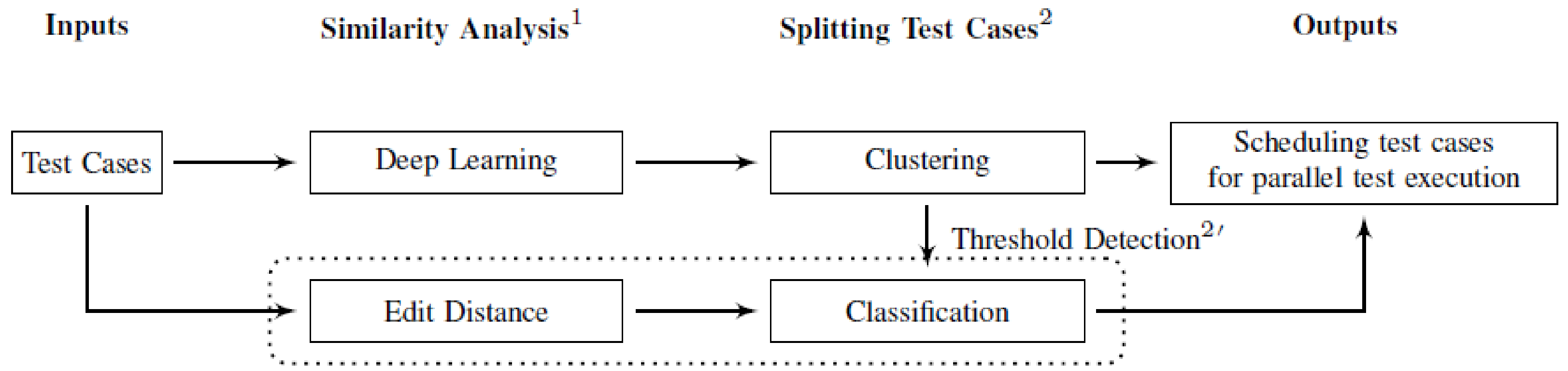
Table I . Examples of similar and non-similar test cases

How to apply that approach?

Solution Approach



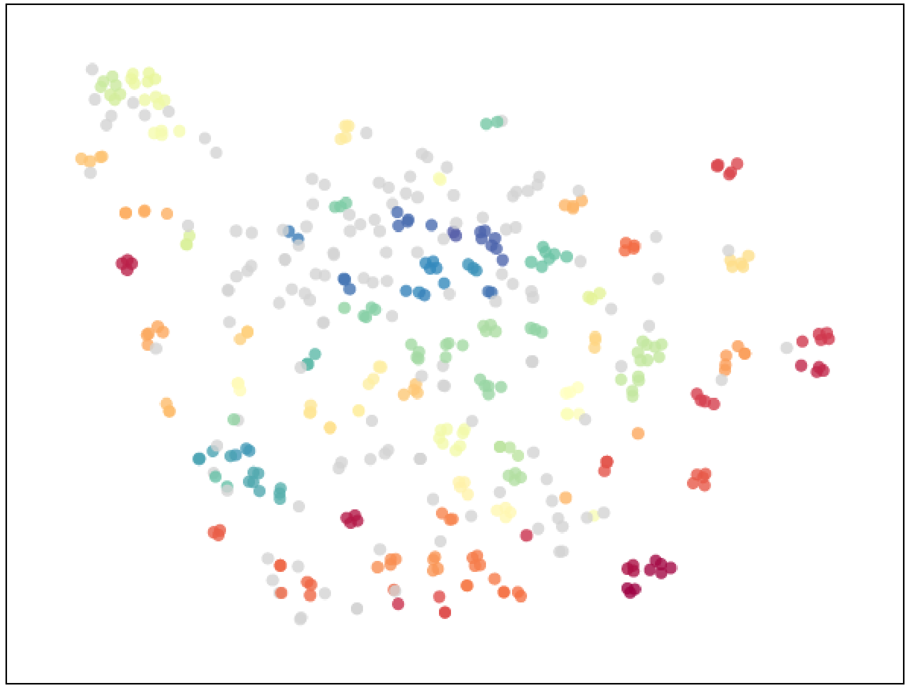
(a)



(b)

Fig.5 (a)Test steps of the solution approach by using two known DL algorithms and (b) an extension of the study comparing the results with the obtained in an earlier publication and find a new data-driven threshold

Results


(a) The clustered test case using the generated vectors by the SBERT.

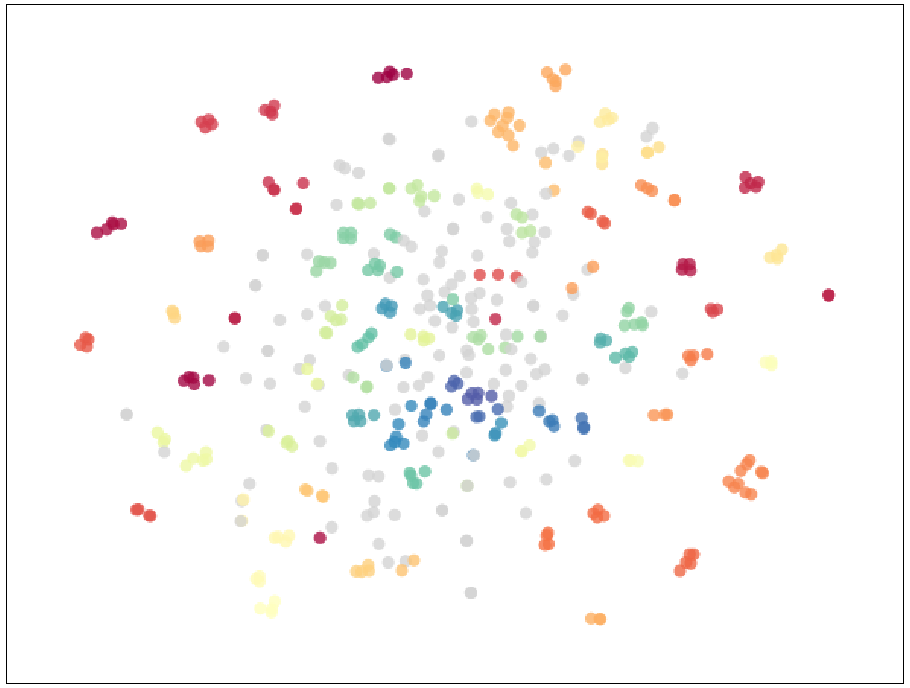
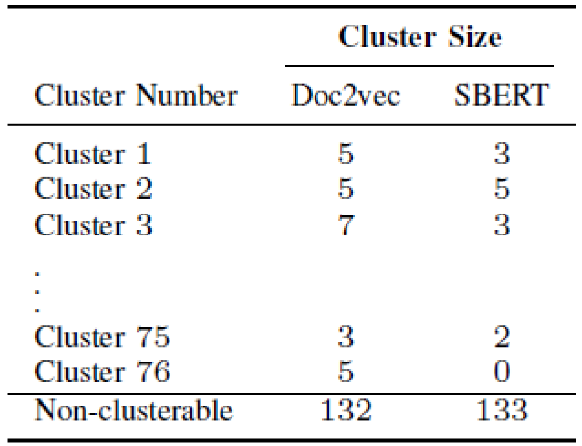

(b) The clustered test case using the generated vectors by the Doc2vec.

Figure 4. Clustered test cases by their semantic similarity. The grey points are the outliners and may be in different dimensions

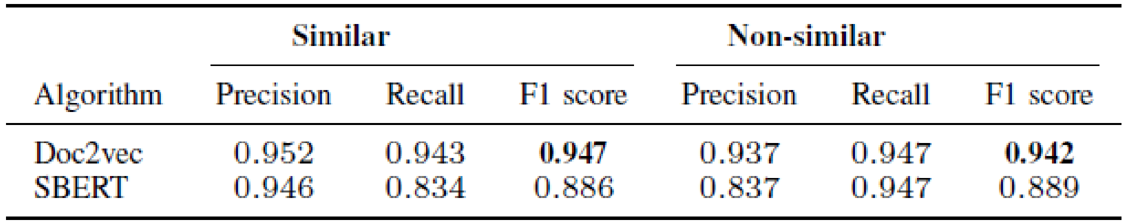
Results

- 76 and 75 clusters of similar test cases, stable and robust
- Comparison against labeled data (pair-wise)
- New threshold found



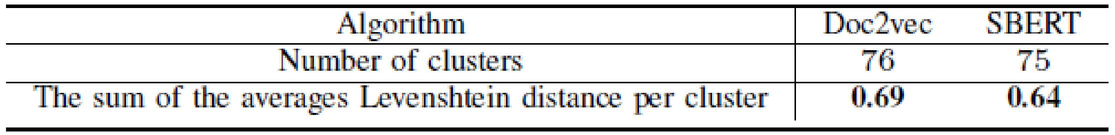
	Cluster Size	
Cluster Number	Doc2vec	SBERT
Cluster 1	5	3
Cluster 2	5	5
Cluster 3	7	3
.		
.		
.		
Cluster 75	3	2
Cluster 76	5	0
Non-clusterable	132	133

Table II. Cluster distribution of both DL methods



	Similar			Non-similar		
Algorithm	Precision	Recall	F1 score	Precision	Recall	F1 score
Doc2vec	0.952	0.943	**0.947**	0.937	0.947	**0.942**
SBERT	0.946	0.834	0.886	0.837	0.947	0.889

Table III. Performance comparison



Algorithm	Doc2vec	SBERT
Number of clusters	76	75
The sum of the averages Levenshtein distance per cluster	**0.69**	**0.64**

Table IV. Data driven threshold, after applying Fig.5b proposed approach

Threats to validity

Words of caution

- Infrastructure – parallel test execution
- Risk for data reliability – same data in test description or ambiguity
- Sensitivity proper of non formal NL nuances
- Examples:
- Follow sequence of [6] and execute (2)
- Follow sequence of [9] and execute (2)

What is left to study?

Discussion and Future Work

- Compare the performance with other DL methods and apply different kind of dimension reduction
- Test the same algorithms with different kind of datasets and/or different kind of technical domains
- How robust is the model?
- Compare the results using automatic classification based on semantic similarities and the rule-based methods, how much could this improves the performance?
- Get more data, documents and databases in order to get more information and reduce the number of hand-crafted work

- About the new threshold, is it viable? What does it imply and is it possible to generalize

Conclusions

Conclusion

- We have tested NLP approach to find similarities between manual integration test cases and clustered the test cases for parallel test execution in two relevant groups
- A comparison with labelled data and the results of the NLP approach is done, the results shows that Doc2vec have better performance in finding similar test cases automatically
- Based on the results of parallel test execution, considerable test time can be saved by executing once the system setup of similar TCs and avoiding redundant TCs. Iff, the system allows certain assumptions as the order of execution is not important and there are not TCs dependencies
- The proposed approach can handle large set of data, which helps the test engineers in the test suite generation and make a better test plan

Thanks for listening!

cristina.landin@oru.se

cristina.landin@ericsson.com

www.ericsson.com

Extra material

- Assumptions made in this paper:
- There are not dependencies between test cases
- The order of execution is not important in the whole test process