Machine Learning and Artificial Intelligence
J.-L. Kim (Ed.)
© 2023 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA230774

Usage of Machine Learning Methods for Cause-Effect Graph Feasibility Prediction

Ehlimana KRUPALIJA [a,1], Emir COGO [a], Damir POZDERAC [a], Aya ALI AL ZAYAT [a], and Ingmar BEŠIĆ [a]

[a] *Department of Computer Science and Informatics, Faculty of Electrical Engineering, University of Sarajevo, 71 000 Sarajevo, Bosnia and Herzegovina*

Abstract. Cause-effect graphs (CEGs) are usually applied for black-box testing of complex industrial systems. The specification process is time-consuming and can result in many errors. In this work, machine learning methods were applied for predicting the feasibility of CEG elements. All information was extracted from graphs contained in CEGSet, a dataset of CEGs. The data was converted to two different formats. The Boolean features format represents relations as separate data rows, whereas the Term-Frequency times Inverse-Document-Frequency (TF-IDF) format represents graphs as data rows. Eight machine learning models were trained on this data. The results of testing by using the 80-20 holdout method indicate that important information is lost when converting the graphs to the TF-IDF format, whereas the Boolean feature format enables 100%-accurate predictions of ensemble methods. The achieved results indicate that pre-trained models can be used as help for domain experts during the CEG specification process.

Keywords. cause-effect graph, natural language processing, machine learning, software quality

1. Introduction

The recent advances in technology have led to a significant increase in available computer memory and processing power, as well as the possibility of usage of CPU and GPU parallelization techniques. This has led to a wide usage of machine learning (ML) algorithms and neural networks on large amounts of data in order to be able to accurately predict future events based on past observations. These concepts have also proved to be useful in the field of software quality, with successful application of ML for multiple purposes such as prediction of fault-proneness [1], automatic generation of black-box tests [2], and detection of code smells [3].

Cause-effect graphs (CEGs) [4] are a black-box testing method which is often applied on complex, industrial, safety-critical or real-time systems (e.g. access control policies [5], high-speed trains [6], and quantum programming [7]) where the usage of other methods would be very time-consuming or impossible due to different limitations. Creating a CEG specification does not require programming knowledge or familiarity

[1]Corresponding Author: Ehlimana Krupalija, University of Sarajevo, 71 000 Sarajevo, Bosnia and Herzegovina; E-mail: ekrupalija1@etf.unsa.ba.

with software quality metrics. However, due to the complexity of systems on which this method is often applied, CEG specifications are usually manually created and this process is very error-prone. It could be made easier if the user received feedback on which CEG elements they added are infeasible, so that mistakes could easily be corrected.

This work presents the first attempt at using ML methods for the detection of CEG feasibility. A summary of related work is given in Section 2. Natural language processing (NLP) methods were used for converting the existing data into numerical matrices to be used as ML model input, as described in Section 3. The achieved results are summarized in Section 4. After calculating the most relevant performance metrics, different ML models were compared and conclusions are given in Section 5.

2. Related work

The majority of previous works focusing on the CEG technique represent case studies where software requirements for different types of systems are manually converted to CEG specifications. A small number of works propose the usage of ML techniques on CEGs. Vogel-Hauser et al. [8] used a dataset of expert knowledge provided by an industrial partner and trained logistic regression and decision tree models for predicting product quality based on CEG elements specified in the dataset. Unfortunately, this dataset is not publicly available. Jang and Kim first proposed the usage of NLP methods (Cause/Conjunction/Clause (C3Tree) models) for converting system requirements into CEG specifications [9], after which they developed Korean Requirement Analyzer for Cause-Effect Graph (KRA-CE), a software tool for automatizing this process [10].

In our recent work [11], we created a compilation of CEGs in the form of a dataset, named CEGSet. It contains 65 CEG specifications manually collected from previous papers and specified in our previously introduced graphical software tool, ETF-RI-CEG [12]. These CEG specifications vary in size, types of nodes, logical relations and dependency constraints. They can be used for training ML models for CEG feasibility prediction. All CEG specifications are available in the TXT format, allowing for the usage of NLP methods in order to convert these specifications into other forms which are adjusted for training of different ML models (e.g. decision tree, naive Bayes, and support vector machine - SVM) [13] and neural networks (e.g. long-short-term memory - LSTM).

After importing the CEG specifications from TXT documents, they are contained in a textual format. This format cannot be directly used for training ML models, which expect numerical matrices as inputs. Many different methods can be used for converting natural language specifications to the numerical format. One such method is Term-Frequency times Inverse-Document-Frequency (TF-IDF) [14] defined in Eq. 1. This method uses the term frequency (TF) t_i in document d, and inverse term t_i frequency (IDF) in the document corpus D. Document corpus represents the entire dataset, a single document represents a CEG and a single term represents a relation of a CEG.

$$TF-IDF(t_i,d,D) = \frac{count(t_i)\ in\ d}{len(d)} \cdot \log_2\left(\frac{len(D)}{count(d)\ containing\ t_i}\right) \quad (1)$$

3. Proposed data formats

Data preprocessing methods were applied on the original TXT representations of the 65 CEGs in order to be able to apply different ML algorithms on CEGSet. CEGSet only contains feasible CEG specifications, so undersampling was performed to formulate new, infeasible CEG specifications. Random numbers were used for ensuring that undersampling would not affect the data in a negative way. A random number of relations was added, smaller than or equal to the number of feasible relations. Relation types were randomly chosen, and a random number of nodes was activated. In order for the relations to be infeasible, multiple cause nodes were always activated, and constraints were applied on wrong node types. A random number of relations was assigned to each new graph. The undersampling resulted in 573 new relations (equal to the size of feasible data) and 25 new infeasible CEGs (27.78% of total dataset size). Afterwards, preprocessing was applied for generating two different data models:

Boolean features format, which is shown on Figure 1, containing a total of 1,146 rows. All logical relationships of CEGs, which are usually represented as Boolean expressions [15], are dataset features in this format. Every data row contains the following 51 data features: *index* (the number of CEG in which the relation is defined), *relation type* (Y for logical relations, N for dependency constraints), *relation name* (logical relations - DIR, NOT, AND, OR, NAND, NOR, dependency constraints - EXC, INC, EXCINC, REQ, MSK), separate features for all cause, intermediate and effect nodes C_i, I_i, E_i (0 - the node is inactive, 1 - the node is an active cause, 2 - the node is an active effect) and *outcome* (Feasible or Infeasible). The dataset was standardized by using the maximum number of cause, intermediate and effect nodes.

	Index	Relation type	Relation name	C1	C2	C3	C4	C5	C6	C7	C8	C9	...	E4	E5	E6	E7	E8	E9	E10	E11	E12	E13	E14	Outcome
0	1	Y	AND	0	1	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	0	Feasible
1	1	Y	AND	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0	0	Feasible
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
571	90	Y	NAND	0	1	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0	0	Infeasible
572	90	Y	NOT	0	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	2	0	0	1	Infeasible

1146 rows × 51 columns

Figure 1. Visualization of the Boolean features data format

TF-IDF graph format, which is shown on Figure 2, containing a total of 90 rows. First a vocabulary of all different relations in the dataset was generated, containing a total of 969 different terms used as data features. The TF-IDF graph format represents each CEG as a single row, and data features represent the presence of the corresponding term from the vocabulary in the given CEG. The value of each data cell is calculated by using the TF-IDF method from Eq. 1.

	0	1	2	3	4	5	6	...	963	964	965	966	967	968	969
0	1.19077	1.417193	1.417193	1.417193	1.417193	1.417193	1.417193	...	0.0	0.0	0.0	0.0	0.0	0.0	Feasible
1	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	Feasible
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
88	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	Infeasible
89	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	Infeasible

90 rows × 970 columns

Figure 2. Visualization of the TF-IDF graph data format

In order to verify that the usage of undersampling did not result in examples without any real value for training of ML models on CEGSet, Pearson correlation was calculated

for all data features of the Boolean features format. The resulting correlation is shown on Figure 3, where the visualization on the left shows the matrix of Pearson correlation coefficients for all 51 data features, whereas the visualization on the right shows only the data features for which the correlation is higher than 25%. It can be easily discerned by the lack of red color on the heatmaps that there is no high correlation in the dataset. The highest correlation is 30% (Index-E12), which indicates that the usage of undersampling yielded random incorrect CEG elements which did not affect the data in a negative way.

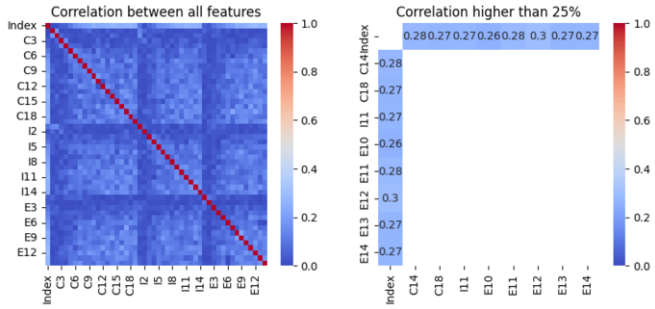


Figure 3. The visualization of Pearson correlation coefficients for the Boolean features data format

4. Experimental results

Different ML models were trained and tested on both data formats of the previously preprocessed dataset: traditional models (naive Bayes, SVM and logistic regression), ensemble models (random forest, voting ensemble, AdaBoost and XGBoost) and neural networks (LSTM). The data was split into two subsets used for training and testing the models by using the holdout method (80% training - 20% testing). The results achieved on the test subset are summarized in Table 1. Performances were measured by using the confusion matrix [13], a table containing results of ML models with the following values: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The following performance metrics were used: accuracy ($\frac{TP+TN}{TP+TN+FP+FN}$), precision ($\frac{TP}{TP+FP}$), recall ($\frac{TP}{TP+FN}$), and F-score (the harmonic mean of precision and recall).

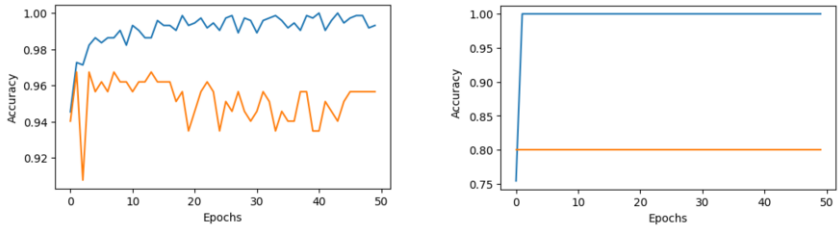
Out of all traditional ML models, SVM achieved the highest average accuracy. This is most likely a consequence of its geometric nature and the ability to correctly determine the non-linear hyperplane that separates data points. Logistic regression did not achieve high accuracy, which means that the hyperplane surface is not linear. Naive Bayes also did not achieve high performances because it assumes probabilistic independence between different rows of data, however data rows are not independent as multiple data rows belong to the same CEG. Only the voting ensemble achieved high performances on the TF-IDF graph format. This means that the usage of decision trees in random forests and XGBoost, and optimization of weak classifiers in AdaBoost, did not help for inferring data relationships from CEGs. The best performances were instead achieved by averaging decisions of multiple regression classifiers in the voting ensemble. Multiple models trained by using the Boolean features data format managed to achieve accuracy of 100%. On the contrary, neither of the models trained by using the TF-IDF graph for-

mat managed to achieve an accuracy of 90% or higher. This indicates that the usage of TF-IDF method, which is based on the frequency of different relations in the dataset, does not help in deducing important relationships between data features.

Table 1. Performance metrics of machine learning models for both data formats

Model	Format	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	Avg. acc. (%)
Naive Bayes	Boolean	89.57	94.02	86.61	90.16	89.23
	TF-IDF	88.89	100	85.71	92.31	
SVM	Boolean	100	100	100	100	**94.45**
	TF-IDF	88.89	100	85.71	92.31	
XGBoost	Boolean	100	100	100	100	88.89
	TF-IDF	77.78	77.78	100	87.5	
Random forest	Boolean	100	100	100	100	88.89
	TF-IDF	77.78	77.78	100	87.5	
AdaBoost	Boolean	100	100	100	100	86.11
	TF-IDF	72.22	100	64.29	78.26	
Voting ensemble	Boolean	100	100	100	100	**94.45**
	TF-IDF	88.89	100	85.71	92.31	
Logistic Regression	Boolean	99.13	99.21	99.21	99.21	88.46
	TF-IDF	77.78	77.78	100	87.5	
LSTM	Boolean	96.09	100	92.91	96.33	86.94
	TF-IDF	77.78	77.78	100	87.5	

The neural-network-based LSTM model was trained during 50 epochs (iterations in which all data samples are used for updating model parameters) and an additional 20% validation subset was used during the training. The results for both data formats are visualized on Figure 4. The TF-IDF graph format did not contain enough useful information for training the model through the epochs. Although the training accuracy quickly rose to 100%, the validation accuracy did not increase from the initial 80%. Contrarily, the Boolean features format was very useful for training the model, achieving validation accuracy of around 96%. However, after around 20 epochs the validation accuracy started constantly dropping, indicating that overfitting had occured.



(a) Boolean features data format

(b) TF-IDF graph data format

Figure 4. Visualization of training accuracy (blue) and validation accuracy (orange) for LSTM model

5. Conclusion

Two different data formats for converting CEGs to numerical matrices were proposed. The Boolean features format resulted in 100% accuracy of ensemble models, whereas the TF-IDF graph format did not result in accuracy of 90% or higher. The most likely cause of this difference is the usage of undersampling for adding infeasible relations. However, due to random indexing of the generated data rows, this did not result in a balanced number of graphs. This led to TF-IDF graph format being unable to capture all relevant information for accurate detection of CEG feasibility. The achieved results are promising and indicate that pre-trained ML models can be incorporated in ETF-RI-CEG or another similar software tool. The pre-trained model would be able to notify the user of the CEG specification feasibility in the tool in real-time. This would reduce the number of errors in the CEG specification process, improving the speed of this process and helping domain experts create CEG specifications for complex systems more quickly.

References

[1] Boucher A, Badri M. Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison. Information and Software Technology. 2018;96:38-67. DOI: 10.1016/j.infsof.2017.11.005.

[2] Kiraç MF, Aktemur B, Sözer H, Şahin Gebizli C. Automatically learning usage behavior and generating event sequences for black-box testing of reactive systems. Software Quality Journal. 2019;27(2):861-83. DOI: 10.1007/s11219-018-9439-1.

[3] Guggulothu T, Moiz SA. Code smell detection using multi-label classification approach. Software Quality Journal. 2020;28(3):1063-86. DOI: 10.1007/s11219-020-09498-y.

[4] Myers GJ, Badgett T, Sandler C. The Art of Software Testing. John Wiley and Sons, Inc.; 2012.

[5] Khdairat Y, Sabri KE. Generating test cases from role-based access control policies using cause-effect graph. Journal of Software. 2018;13(9):497-505. DOI: 10.17706/jsw.13.9.497-505.

[6] Dou L, Yang WD. Design of test case for ATP speed monitoring function based on cause-effect graph. In: Proceedings of 2019 11th CAA SAFEPROCESS. Xiamen; 2019. p. 246-50. DOI: 10.1109/SAFEPROCESS45799.2019.9213325.

[7] Oldfield N, Yue T, Ali S. Investigating quantum cause-effect graphs. In: Proceedings - 3rd International Workshop on Quantum Software Engineering, Q-SE 2022. Pittsburgh; 2022. p. 8-15. DOI: 10.1145/3528230.3529186.

[8] Vogel-Heuser B, Karaseva V, Folmer J, Kirchen I. Operator knowledge inclusion in data-mining approaches for product quality assurance using cause-effect graphs. In: 20th IFAC World Congress. Toulouse; 2017. p. 1358-65. DOI: 10.1016/j.ifacol.2017.08.233.

[9] Jang WS, Kim RYC. Automatic generation mechanism of cause-effect graph with informal requirement specification based on the Korean language. Applied Sciences (Switzerland). 2021;11(24):1-13. DOI: 10.3390/app112411775.

[10] Jang WS, Kim RYC. Automatic cause-effect graph tool with informal Korean requirement specifications. Applied Sciences (Switzerland). 2022;12(18):1-16. DOI: 10.3390/app12189310.

[11] Krupalija E, Cogo E, Bećirović Š, Prazina I, Pozderac D, Bešić I. CEGSet: Collection of standardized cause-effect graph specifications. In: 2023 12th Mediterranean Conference on Embedded Computing (MECO). Budva; 2023. p. 1-4. DOI: 10.1109/MECO58584.2023.10155063.

[12] Krupalija E, Cogo E, Bećirović Š, Prazina I, Pozderac D, Bešić I. New graphical software tool for creating cause-effect graph specifications. Journal of Communications Software and Systems. 2022;18(4):311-22. DOI: 10.24138/jcomss-2022-0076.

[13] Smolyakov V. Machine Learning Algorithms in Depth. Manning Publications; 2023.

[14] Lane H, Dyshel M. Natural Language Processing in Action. Manning Publications; 2023.

[15] Ayav T, Belli F. Mutation-based minimal test suite generation for Boolean expressions. International Journal of Software Engineering and Knowledge Engineering. 2023;33(6):865-84. DOI: 10.1142/S0218194023500183.