A Meta-Language Approach for Machine Learning

Nicholas Caporusso[1], Trent Helms[1] and Peng Zhang[1]

[1] Fort Hays State University, 600 Park Street,
67601 Hays, United States
n_caporusso@fhsu.edu, {tehelms, p_zhang15_sia.se}@mail.fhsu.edu

Abstract. In the last decade, machine learning has increasingly been utilized for solving various types of problems in different domains, such as, manufacturing finance, and healthcare. However, designing and fine-tuning algorithms require extensive expertise in artificial intelligence. Although many software packages wrap the complexity of machine learning and simplify their use, programming skills are still needed for operating algorithms and interpreting their results. Additionally, as machine learning experts and non-technical users have different backgrounds and skills, they experience issues in exchanging information about requirements, features, and structure of input and output data.
This paper introduces a meta-language based on the Goal-Question-Metric paradigm to facilitate the design of machine learning algorithms and promote end-user development. The proposed methodology was initially developed to formalize the relationship between conceptual goals, operational questions, and quantitative metrics, so that measurable items can help quantify qualitative goals. Conversely, in our work, we apply it to machine learning with a two-fold objective: (1) empower non-technical users to operate artificial intelligence systems, and (2) provide all the stakeholders, such as, programmers and domain experts, with a modeling language.

Keywords: End-user development · Artificial Intelligence · Machine Learning · Neural Networks · Meta-design · Goal Question Metric

1 Introduction

In the recent years, artificial intelligence (AI) systems [1] ended a long incubation phase and became mainstream technology. Nowadays, different types of machine learning (ML) algorithms are utilized for a variety of tasks (e.g., computer vision, natural language processing, and audio and speech recognition) in several different applications in diverse contexts and domains (e.g., robotics, manufacturing, and healthcare) [2]. Indeed, the design and implementation of ML algorithms, and specifically, neural networks (NNs), involve a number of challenges, such as, selecting the most relevant features and creating a representative model, identifying the type of task (i.e., supervised or unsupervised learning), choosing which family of algorithm to use (e.g., logistic regression, Support Vector Machines, or Artificial Neural Networks), selecting the most effective approach (e.g., recurrent or feed forward NNs), configur-

ing parameters (e.g., number of input and hidden layers), and fine-tuning the system for increasing its performance (e.g., introducing modifications, pre-processing input, or using hybrid approaches) [1]. Consequently, the overwhelming complexity of ML restrained its progress, in the last decades. To this end, communities in the field of AI joined their efforts and created software libraries that package ML systems into ready-to-use toolkits. These, in turn, enable a larger spectrum of professionals to choose among many available algorithms and to conveniently use them by simply customizing their behavior. As a result, developers do not have to deal with the complexity and risks of programming a system from scratch. Examples include Tensorflow [3], Apache MXnet [4], Microsoft Cognitive Toolkit (CNTK) [5], Scikit-learn [6], Pytorch [7], and many other toolkits that empower developers to operate ML systems and incorporate them in projects. As several software libraries are released under Open Source licenses, they foster the growing adoption of ML algorithms and, simultaneously, contribute to addressing current open issues and to improving performance. Moreover, the Open Source community, as well as organizations, released additional libraries. Nowadays, packages (e.g., sample datasets, visual or command-line tools, software automatic feature selection) can be layered on top of ML engines to achieve higher levels of abstraction and streamline the most tedious operations. In addition, by hiding complexity, toolkits render ML systems accessible to less-expert users, increase their awareness on AI, and promote their willingness explore the field [8].

Nevertheless, understanding and operating ML libraries still require some degree of programming literacy, which prevents many individuals who lack coding skills from being able to approach the use of machine learning algorithms and even understand their dynamics. Also, this is due to several factors that contribute to rendering ML obscure [9], such as, the inherent properties of the hidden layers of a NN. As a consequence, the intrinsic and intentional opacity of algorithms and requirements in terms of technical literacy increase the current divide between programmers who are expert about AI but lack domain knowledge [10] and customers who should not be required to be proficient in ML and yet, are not provided with any actionable framework for approaching it. This, in turn, leads to unrealistic expectations, communication barriers, and slower adoption rate, especially among small businesses. As the number of organizations that demand or are offered ML-based solutions is increasing, facilitation tools are needed to benefit both domain experts and AI scientists. Regardless of the experience level and the availability of powerful tools, given the novelty of this field, there is a lack of meta-design methodologies that improve the ergonomic aspects of machine learning and enable end-user development (EUD) of AI systems.

In this paper, we focus on the socio-technical dimension of artificial intelligence, and we propose a meta-language and design framework that (1) supports individuals with a broader spectrum of expertise to approach and leverage AI, (2) facilitates communication and information exchange between stakeholders having different backgrounds and heterogeneous skills, and (3) provides users with an actionable language for representing the meta-design of a ML system. To this end, we adopt the Goal-Question-Metric (GQM) approach as a viable framework for supporting end-user development across multiple disciplines. We describe the model and we detail examples of its implementation.

2 Related Work

Artificial Intelligence has the objective of creating systems that enable computers to supplement or replace human intelligence in solving problems that require some degree of decision-making [11]. In the last decades, research led to algorithms that, in addition to processing data, learn from them, that is, they are able to modify their internal structure based on the information they are trained with. By doing this, they can cope with certain degree of ambiguity and variation, fill-in data gaps, and leverage existing knowledge to take decisions in unprecedented scenarios. As an example, neural networks are being utilized in the healthcare domain to automatically identify cancer [12], monitor eye conditions [13], detect movement disorders [14], or support physicians' decisions in rehabilitation programs [15]. Moreover, recent advances in AI produced deep learning systems that are able to autonomously design their internal structure based on the input data. As a result, they achieve higher accuracy and performance in tasks that involve unstructured information (e.g., text, sound, and images) [16], though they require longer training time or more powerful hardware resources.

Although several groups pursuit the development of a general AI, that is, a unique system that can deal with any type of task, most of the problems are solved with narrow algorithms, each focused at one specific aspect. To this end, toolkits are extremely useful, as organizations can access a collection of methods and use them as building blocks, on a case by case basis. Although well-curated packages are more functional and easier to use than pure algorithms, they are not fully self-operational [17] and their implementation requires highly-skilled AI scientists, though there are early attempts to create high-level programming languages that support end-users [18].

To this end, several methodologies for the operationalization of conceptual models are available. The goal-question-metric paradigm is a structured approach for translating high-level objectives into actionable, quantitative metrics [19]. Most of the work using the GQM method is related to software engineering [20] [21]: as an example, the authors of [22] proposed a solution based on GQM for classifying well-known design patterns using machine learning. However, their goals and metrics consist in a perfect bijection. Hence, there is little advantage in using a predictor and in approaching the problem with the use of machine learning. Conversely, in [23], the authors describe a framework for risk management: the GQM methodology is utilized to determine which metrics affect projects. To this end, they envision a high-level goal, identify questions that need to be answered in order to evaluate whether the objective is met, and specify metrics that help take a quantitative approach to addressing each of the questions. After identifying the key metrics, they use them as features to train an artificial neural network (ANN) to produce a regression function estimate and to establish the success probability of a project based on the comparison between the threshold value and the regression function. The authors of [24] utilize ML techniques to evaluate the quality of software metrics defined using a GQM approach. Nevertheless, the structure of GQM itself can be utilized as an interpretation framework for artificial reasoning and, specifically, for describing the structure and functioning of neural networks, though it can be applied to other families of AI algorithms as well, as detailed in the next Section.

3 GQM as a Meta-language for Machine Learning

GQM was initially designed as a language for translating business goals into questions and, ultimately, into metrics, in a top-down fashion. As a result, it can be utilized as a tool for supporting the top management in taking business decisions and developing assessment plans: specifying questions that enable middle managers to determine whether certain goals are met helps teams focus on the most important aspects, that is, improving the metrics that are meaningful to reaching their objectives. Simultaneously, data collected from low-level processes can be interpreted by middle managers, who can report to the executives in a bottom-up fashion. This type of hierarchical structure can be represented as a pyramid of connections: the graph resulting from a GQM model and its closed-loop process are shown in Figure 1.

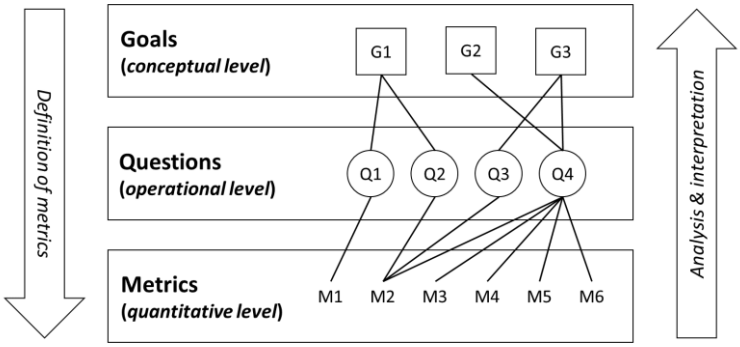

Fig. 1. An example of the logical structure of GQM, adapted from [25]: conceptual goals are specified into questions that can be answered by evaluating quantitative elementary or aggregated metrics. The model can be approached both in a top-down and in a bottom-up fashion to close the information loop between the conceptual, operational, and quantitative abstraction levels.

Although the GQM method is primarily considered as a top-down approach to problem solving, it can be utilized in a bottom-up fashion to identify the questions and goals that can be addressed using data and metrics that are already available: the simplicity of this model makes it very powerful, because it provides a versatile, bi-directional generative process for (1) quantifying high-level concepts and define which metrics are needed and, simultaneously, (2) for analyzing available measurements to generate hypotheses and theories. Furthermore, it results in a closed-loop system: information from goals and metrics can be utilized to evaluate whether the structure consistent and, in case, to make changes that can be reflected in the process.

Indeed, current applications of the GQM methodology rely on human intelligence and domain knowledge for processing information. However, its logic is similar to a neural network: a set of quantitative features (metrics) collected from an input layer (quantitative level) are fed into an internal layer (operational level) in which nodes (question) enable processing data and generating outputs (goals). Both in GQM and in

NNs, metrics (features), either in elementary or aggregated form, serve as relevant predictors for the goal (output). The specificity of the intelligent system relies in the operational level: humans use domain knowledge for specifying the connections that link the conceptual and the quantitative levels (and vice versa) and build the GQM graph, whereas the hidden layer of a neural network incorporates methods, such as, activation function and backpropagation, that introduce non-linearity and enable weighing nodes and pruning connections. The analogy applies to most families and types of ML algorithms. The main difference between the goal-question-metric approach and a neural network is in that the former enables specifying multiple goals, whereas the outputs of a NN typically belong a homogeneous set.

In our work, we propose the GQM approach as a conceptual framework for the operationalization of systems and processes and, specifically, as a meta-language for designing applications based on machine learning. Specifically, we aim at leveraging it as a user-friendly and algorithm-agnostic modeling instrument that can abstract the lower-level components of a ML system and their technical implementation, so that AI experts and non-technical users can utilize it as a meta-design and collaboration tool.

3.1 Meta-language

In its traditional use, GQM serves two purposes in representing systems and processes: as a top-down modeling instrument, its objective is helping identify measurable elements (i.e., metrics); on the other hand, after the quantitative level has been structured, it is a bottom-up assessment tool that uses data to measure qualitative goals. Although the state of the art primarily considered goal-question-metric for its applied outcome, its conceptual design is extremely valuable as a meta-language. As with other modeling languages utilized in software engineering, such as, the Unified Modeling Language (UML) [26], it can be applied to achieve a formal representation of a system. However, GQM is especially relevant to the ML domain from a system design standpoint: as its elements and graph-like structure mimic the structure and functioning of artificial intelligence algorithms, from machine learning to deep learning, they can be utilized as components of a meta-language for modeling the relationship between input features and output goals.

Self-explaining, cross-domain definitions. The use of straightforward vocabulary, such as, metrics and goals, makes the language easy to understand by non-technical users, as the terms are already utilized in business rules across different backgrounds. Moreover, abstracting from the concepts of inputs and outputs removes any potential suggestion of an implicit direction, enables approaching the model in a top-down as well as in a bottom-up fashion, and supports its use for data-driven specification and for conceptual description and analysis.

Design process as a meta-representation of how ML works. As discussed earlier, the fundamental components of GQM especially mimic the structure of a neural network. Specifically, defining the conceptual model with the goal-question-metric approach involves a process in which a human agent with sufficient expertise of a domain de-

fines the layers and nodes and trains the resulting network: questions, which represent nodes in the hidden layer, leverage explicit and tacit knowledge and facilitate connecting input features and outputs. As human experience enables to directly prune nodes and connections that have low relevance, the graph resulting from GQM modeling directly translates into a network in which all links are significant and have the same weight. Although the similarity with NNs is straightforward, the goal-question-metric approach can similarly be applied to any ML algorithm.

Actionable by all the stakeholders. On a merely conceptual level, the GQM meta-language, as other types of modeling tools, supports communication and exchange of information between domain experts and engineers. In addition, differently from instruments that serve a representational function only, structures obtained using the GQM meta-language are immediately actionable: the defined metrics can directly be utilized, either for helping humans measure goals or as an input dataset for a machine learning algorithm. Therefore, non-technical users can describe objectives, formalize domain knowledge, and explain the information needed for the assessment of goals. Conversely, AI experts can utilize the quantitative level as a model of the dataset; questions in the operational layer can become criteria for deciding the family and type of algorithm, whereas goals suggest the overall complexity of the problem and help refine implementation choices.

Interoperable and robust to iterations. Although the GQM meta-language can be utilized as a shared collaboration tool by any stakeholder, it supports separation of concerns and removes any interdependence between the model and its implementation. As a result, decisions regarding the configuration of a specific application do not affect how the graph structure is defined, and vice versa, changes in the GQM diagram can be treated as iterations and managed using a versioning system. This, in turn, enables reflecting them at an application level. Furthermore, the GQM meta-language is algorithm-agnostic: as changes in the machine learning engine do not affect the model, they can be implemented without impacting non-technical end users. Consequently, it is an interoperable, reusable, scalable, and suitable tool for designing systems that cope with high degree innovation and for domains that are exposed to intrinsic and extrinsic change factors.

Extensible. The goal-question-metric meta-language supports adding features and components that can improve its expressiveness. For instance, metrics and questions in a GQM graph can be associated with different weights, so that the structure can include a specification of the importance of each element, regardless of whether their relevance is decided by a human agent or calculated by a machine. This, in turn, can be utilized by non-technical stakeholders to be more specific in assessing the goal, or by AI experts, who can take weights into consideration when designing the network (e.g., for training purposes and for fine-tuning it). Moreover, software implementations of the GQM meta-language could interact with packages that incorporate ML algorithms, data visualization libraries, and performance analysis tools.

3.2 Domain-Knowledge Sharing Tool

One of the crucial activities in the development of software, especially if they incorporate some form of AI, is the collection and analysis of system requirements. As an example, numerous applications use artificial NNs in medical diagnosis, though interaction between stakeholders is often inefficient because health professionals and AI scientists approach data collection differently, structure information in dissimilar fashion, and present concepts using their own jargons. This is especially relevant to the outcome of a ML-based system: not being able to capture important domain features is among the main reason of poor performance.

To this end, GQM can be utilized as a tool that facilitates communication between ML-experts and non-technical users. The methodology inherently gathers information and structures it in an easy-to-understand and non-ambiguous written format that supports knowledge transfer across domains and among stakeholders. In addition, it can be utilized to elicit and formalize tacit knowledge, that is, information that would otherwise be difficult to collect or transfer using traditional means (e.g., automatic knowledge based on acquired expertise). Simultaneously, this approach results in better project documentation and helps track, revise, and improve software requirements and specifications over project iterations. In this context, the GQM meta-language can help engineer a lean participatory design and development process, increase communication and awareness among all stakeholders, and produce project documentation for future use. Indeed, other methodologies and meta-languages (e.g., UML) can be utilized for this purpose. However, in comparison, the GQM meta-language can especially capture the key items of a system based on machine learning, and it has the unique advantage of incorporating the conceptual and operationalization layers in a single model that completely describes the system and its conceptual, operational, and quantitative abstraction levels, while remaining independent from a specific implementation.

Incorporating the three layers in a single GQM meta-model results in a self-explanatory, actionable instrument that encapsulates and describes a problem in a way that is useful for the community of domain experts as well as for ML developers. Consequently, it can be utilized to create shared repositories of reusable practices and processes in which users could publish a formal description of the problem together with datasets that represent the defined metrics. This, in turn, would result in a self-contained package that provides the ML community with the necessary information for developing or customizing algorithms and contributing solutions. Simultaneously, this could give an impulse and stimulate different approaches in the context of AI.

3.3 Feature selection

Indeed, having too many irrelevant features in a dataset affects the accuracy of AI models and causes overfitting. Moreover, the more data available, the longer it takes to train the system, especially in the case of deep neural networks. Therefore, feature selection, that is, preparing the dataset to remove redundant and misleading information, is among the most crucial tasks in machine learning. Although there are tools that automate this activity, this step is often the most labor-intensive part of building an AI system for a two-fold reason: (1) most businesses collect data in formats that

are not ready for being utilized in ML (e.g., scarce, scattered in multiple databases, or stored in unstructured and ambiguous formats); moreover, domain knowledge is required for distinguishing relevant features from less significant ones, depending on the objective. In this regard, the GQM meta-language inherently helps highlight the relationship between relevant metrics and goals and empowers domain experts to contribute to the process and make it more efficient. Furthermore, it provides all stakeholders with a formal protocol that describes the type of information that needs to be collected for each goal and the format required for each metric.

3.4 Network of Machine Learning Agents

As discussed earlier, GQM supports designing models that include multiple goals. This contrasts with most implementations of narrow AIs, which typically have one objective, only. Nevertheless, this feature is advantageous in that a single structure defined in the GQM meta-language can be utilized to represent the entire set of multi-faceted aspects in the problem space: the resulting graph describes a network of ML agents that use a shared pool of metrics to achieve their individual objectives. By doing this, the meta-language provides a way to measure the complexity of the model and results in a better understanding of the algorithms that will be required for implementing a solution. Furthermore, it creates a single representation of the several datasets that are required for assessing the different objectives. This, in turn, optimizes consistency, efficiency, and data reuse. Simultaneously, an overview of the entire problem space might help stakeholders identify alternative assessment and implementation strategies.

3.5 End-user Development

The simple and intuitive structure of GQM provides an interpretation framework for ML and it can help non-technical users understand how AI algorithms work without overwhelming them with non-necessary implementation details. Moreover, the GQM meta-language is suitable for supporting end-user development and for empowering domain experts to become active contributors. The introduction of UML as a modeling tool for software resulted in systems that support end-user development, which, in turn, simplified programming and helped refine modeling tools. The GQM meta-language offers a similar opportunity for system-system co-evolution in the context of ML. In this regard, software using the GQM meta-language as a specification format could interact with ML toolkits to enable users to dynamically instantiate agents that implement a specific algorithm depending on the goal. This would result in the possibility of empowering non-technical users to design the GQM graph, select an option from the list of available agents, train it using datasets defined in accordance with the metrics, and compare its performance with other algorithms.

Finally, from a human-system co-evolution perspective, reviewing the structure of a GQM model based on the performance of ML algorithms can result in a self-reflection process that supports domain experts in improving the definition of their objectives, the questions that must be asked to meet goals, and the metrics that must be collected to answer the questions, thus making the overall process more efficient.

4 Conclusion

Artificial intelligence is expected to be one of the disciplines that will impact society the most, in the next years. However, implementing machine learning algorithms still requires sophisticated programming skills: though packages, libraries and toolkits are available, most of their potential is accessible to experts and developers.

In this paper, we proposed the goal-question-metric approach as a meta-language for modeling machine learning systems. Although GQM has primarily been utilized to evaluate software metrics, we demonstrated that it is especially suitable as a description language for the meta-design of ML applications. We detailed its advantages for non-technical users, that is, it wraps the complexity of the implementation of machine learning in an easy-to-access abstraction layer, simplifies the understanding and use of algorithms, and supports end-user development. Moreover, we detailed how it can be utilized by AI experts an interoperable, reusable, and scalable modeling tool. Being algorithm-agnostic, the GQM meta-language is independent from any technical implementation and, thus, can facilitate communication among all the stakeholders involved in the development of a ML-based system, regardless of their level of proficiency with AI or programming languages. Finally, we outlined the main differences with other modeling languages, such as UML, and we presented examples of the proposed methodology as a viable instrument for gathering requirements, for feature selection, and for dynamically reflecting changes to the model.

References

1. Nasrabadi, N.M., 2007. Pattern recognition and machine learning. Journal of electronic imaging, 16(4), p.049901.
2. Sebastiani, F., 2002. Machine learning in automated text categorization. ACM computing surveys (CSUR), 34(1), pp.1-47.
3. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. Tensorflow: a system for large-scale machine learning. In OSDI (Vol. 16, pp. 265-283).
4. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. and Zhang, Z., 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274.
5. The Microsoft Cognition Toolkit (CNTK), [online] Available: https://cntk.ai.
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825-2830.
7. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A., 2017. Automatic differentiation in pytorch.
8. Erickson, B.J., Korfiatis, P., Akkus, Z., Kline, T. and Philbrick, K., 2017. Toolkits and libraries for deep learning. Journal of digital imaging, 30(4), pp.400-405.
9. Burrell, J., 2016. How the machine 'thinks': Understanding opacity in machine learning algorithms. Big Data & Society, 3(1), p.2053951715622512.
10. Bach, M.P., Zoroja, J. and Vukšić, V.B., 2013. Determinants of firms' digital divide: A review of recent research. Procedia Technology, 9, pp.120-128.

11. Jordan, M.I. and Mitchell, T.M., 2015. Machine learning: Trends, perspectives, and prospects. Science, 349(6245), pp.255-260.
12. De Pace, A., Galeandro, P., Trotta, G.F., Caporusso, N., Marino, F., Alberotanza, V. and Scardapane, A., 2017, April. Synthesis of a Neural Network Classifier for Hepatocellular Carcinoma Grading Based on Triphasic CT Images. In Recent Trends in Image Processing and Pattern Recognition: First International Conference, RTIP2R 2016, Bidar, India, December 16–17, 2016, Revised Selected Papers (Vol. 709, p. 356). Springer. doi:10.1007/978-3-319-60483-1_13.
13. Bevilacqua, V., Uva, A.E., Fiorentino, M., Trotta, G.F., Dimatteo, M., Nasca, E., Nocera, A.N., Cascarano, G.D., Brunetti, A., Caporusso, N. and Pellicciari, R., 2016, December. A Comprehensive Method for Assessing the Blepharospasm Cases Severity. In International Conference on Recent Trends in Image Processing and Pattern Recognition (pp. 369-381). Springer, Singapore. doi: 10.1007/978-981-10-4859-3_33
14. Bevilacqua, V., Trotta, G.F., Loconsole, C., Brunetti, A., Caporusso, N., Bellantuono, G.M., De Feudis, I., Patruno, D., De Marco, D., Venneri, A. and Di Vietro, M.G., 2017, July. A RGB-D Sensor Based Tool for Assessment and Rating of Movement Disorders. In International Conference on Applied Human Factors and Ergonomics (pp. 110-118). Springer, Cham. doi: 10.1007/978-3-319-60483-1_12
15. Bevilacqua, V., Trotta, G.F., Brunetti, A., Caporusso, N., Loconsole, C., Cascarano, G.D., Catino, F., Cozzoli, P., Delfine, G., Mastronardi, A. and Di Candia, A., 2017, July. A Comprehensive Approach for Physical Rehabilitation Assessment in Multiple Sclerosis Patients Based on Gait Analysis. In International Conference on Applied Human Factors and Ergonomics (pp. 119-128). Springer, Cham. doi: 10.1007/978-3-319-60483-1_13
16. LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), p.436.
17. Gerbert, P., Hecker, M., Steinhäuser, S. and Ruwolt, P., 2017. Putting artificial intelligence to work. BCG Henderson Institute. Munich, Germany: The Boston Consulting Group. Retrieved January, 22, p.2018.
18. Almassy, N., Kohle, M. and Schonbauer, F., 1990, June. Condela-3: A language for neural networks. In Neural Networks, 1990., 1990 IJCNN International Joint Conference on (pp. 285-290). IEEE.
19. Basili, V.R., 1992. Software modeling and measurement: the Goal/Question/Metric paradigm.
20. Caldiera, V.R.B.G. and Rombach, H.D., 1994. Goal question metric paradigm. Encyclopedia of software engineering, 1, pp.528-532.
21. Fontana, F.A., Zanoni, M., Marino, A. and Mantyla, M.V., 2013, September. Code smell detection: Towards a machine learning-based approach. In Software Maintenance (ICSM), 2013 29th IEEE International Conference on (pp. 396-399). IEEE.
22. Uchiyama, S., Washizaki, H., Fukazawa, Y. and Kubo, A., 2011, March. Design pattern detection using software metrics and machine learning. In First International Workshop on Model-Driven Software Migration (MDSM 2011) (p. 38).
23. Sarcià, S.A., Cantone, G. and Basili, V.R., 2007. A Statistical Neural Network Framework For Risk Management Process. Procedings of ICSOFT, Barcelona, SP, 2007.
24. Werner, E., Grabowski, J., Neukirchen, H., Röttger, N., Waack, S. and Zeiss, B., 2007, September. TTCN-3 quality engineering: using learning techniques to evaluate metric sets. In International SDL Forum (pp. 54-68). Springer, Berlin, Heidelberg.
25. Gasson, S., 2012. Analyzing key decision-points: problem partitioning in the analysis of tightly-coupled, distributed work-systems. International Journal of Information Technologies and Systems Approach (IJITSA), 5(2), pp.57-83.
26. Rumbaugh, J., Booch, G. and Jacobson, I., 2017. The unified modeling language reference manual. Addison Wesley.