IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 May 2023, accepted 14 June 2023, date of publication 12 July 2023, date of current version 19 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3294840

SURVEY

Requirements Engineering in Machine Learning Projects

ANA GJORGJEVIKJ, KOSTADIN MISHEV, LJUPCHO ANTOVSKI, AND DIMITAR TRAJANOV, (Member, IEEE)
Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, 1000 Skopje, North Macedonia
Corresponding author: Ana Gjorgjevikj (gjorgjevikj.ana@students.finki.ukim.mk)

ABSTRACT Over the last decade, machine learning methods have revolutionized a large number of domains and provided solutions to many problems that people could hardly solve in the past. The availability of large amounts of data, powerful processing architectures, and easy-to-use software frameworks have made machine learning a popular, readily available, and affordable option in many different domains and contexts. However, the development and maintenance of production-level machine learning systems have proven to be quite challenging, as these activities require an engineering approach and solid best practices. Software engineering offers a mature development process and best practices for conventional software systems, but some of them are not directly applicable to the new programming paradigm imposed by machine learning. The same applies to the requirements engineering best practices. Therefore, this article provides an overview of the requirements engineering challenges in the development of machine learning systems that have been reported in the research literature, along with their proposed solutions. Furthermore, it presents our approach to overcoming those challenges in the form of a case study. Through this mixed-method study, the article tries to identify the necessary adjustments to (1) the best practices for conventional requirements engineering and (2) the conventional understanding of certain types of requirements to better fit the specifics of machine learning. Moreover, the article tries to emphasize the relevance of properly conducted requirements engineering activities in addressing the complexity of machine learning systems, as well as to motivate further discussion on the requirements engineering best practices in developing such systems.

INDEX TERMS Machine learning, requirements engineering, software engineering, software requirements.

I. INTRODUCTION

Artificial intelligence (AI) and its sub-field machine learning (ML) have had significant research activity and commercial use for decades, but over the last decade, they have become significantly more popular and accessible to the wider community. To a large extent, that has happened as a result of the significant progress made in the ML sub-field known as deep learning (DL) [1], which relies on deep neural networks to learn meaningful representations from raw data and bypasses the need for manual feature engineering. The significant achievements that DL has made possible in many fields (e.g., [2], [3]) have been mainly a result of the improvements in the techniques used to train deep neural networks, the availability of larger datasets and more powerful computers, as well as the significantly reduced training time [4]. This progress has gradually made ML algorithms ubiquitous in many areas of our society and everyday activities.

ML methods introduce a different approach to software programming in which, instead of writing problem-solving instructions in software code, learning algorithms learn solutions to problems through data. This new approach generally consists of specifying a goal of the program behavior, e.g., by collecting relevant data, limiting the solution search space through a rough skeleton of code, and letting the learning algorithm find the best solution [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Vicente Alarcon-Aquino.

Although ML systems typically require a significant amount of conventional software code to support the ML models which are at their core [6], the new approach to software programming introduced by ML challenges the established software development process and best practices. The ML software development process is characterized by its data-centricity, non-linearity, and multiple feedback loops between stages, which can become even more complex in systems with multiple ML components that interact in complex ways [7]. Previous experience has shown that while developing and deploying ML systems can sometimes be a relatively fast and inexpensive process, maintaining such systems over time can be challenging and costly, mainly because ML systems are prone to accumulating hidden technical debt [6]. In addition to engineering challenges, AI systems introduce a new set of challenges related to predicting their exact behavior in different situations, predicting their effect on individuals or society, and ensuring their trustworthiness. Sometimes it can be challenging to predict the behavior and outcomes of AI systems precisely because of their complexity, susceptibility to imperfections of the data they learn from, the difficulty in interpreting the functional processes that generate their output, as well as any new behavior arising from their interactions with the world or changes in their environment [8]. In that context, the research literature has reported an example of bias in a commercial ML system that had been discovered only after the system had been released for use, and negative user experiences had been reported [9]. Developing an appropriate solution to a real problem through ML is a complex process that requires meticulous analysis of the system capabilities, behavior, risks, limitations, qualities, and intended/unintended use cases. It also requires analysis of the potential trade-offs between the stakeholders' (sometimes too high) expectations and their feasibility constrained by the available data and resources, between the aspiration for higher model accuracy (often leading to higher model complexity) and the compliance with quality, ethical, and legal constraints, between the time spent on experimenting and the expected time to delivery of an initial value to the stakeholders, to name just a few.

The analysis of ML systems' feasibility, the formulation of their important quality, ethical, and legal attributes, their limitations, constraints, and risks, the decisions on the acceptable trade-offs, and the choice of system validation strategies in agreement with their stakeholders are all activities that belong to the requirements engineering (RE) stage in the conventional software development process. This leads to the conclusion that RE activities are as crucial to the ML development process as they are to the conventional one. However, when ML components replace conventionally programmed ones, the software requirements should correspond to the different development process and the ML specifics. Otherwise, the consequences of incorrectly engineered or missing requirements may be even greater in the case of ML systems, given the effects these systems may have on individuals, the control mechanisms they require, and the ethical or legal requirements they are subjected to. Nevertheless, not much is available on RE for ML systems, nor have the RE activities from the related domain processes, such as Cross-Industry Standard Process for Data Mining (CRISP-DM) and Knowledge Discovery in Databases (KDD), been detailed sufficiently by the RE and ML communities [10]. All of the above was a motivation for this article which tries to answer the following research questions:

1) Are conventional RE activities relevant to the ML development process, what challenges does this process bring them, and what are their necessary adjustments to better fit into this process?
2) What types of requirements are particularly important in addressing the ML systems specifics, do these specifics affect the conventional understanding of the requirements, and how should this understanding be adjusted?

The article answers the research questions through a mixed-method study, i.e., (1) a review of previously published literature in the fields of ML and RE, and (2) a case study involving a research project of the authors of this article [11]. The mixed-method study was primarily motivated by the lack of practical examples of (1) RE activities in ML projects and (2) requirements specifications for ML systems reported in the literature. The case study gave us the opportunity to share the challenges we faced during the RE activities in a research project involving ML, our approach to dealing with those challenges, and excerpts from the requirements specification for the developed ML system.

The objectives of this article are the following:

1) Emphasizing the importance of RE activities in dealing with the complexity of ML systems.
2) Analyzing the aspects of conventional RE that need to be adjusted to the ML specifics.
3) Giving an overview and sharing our experiences on this relatively unexplored to date, but, in our opinion, important topic.

The rest of the article is organized as follows. First, an overview of the related work on RE for AI/ML systems is presented. Next, a description of the methodology used to identify relevant articles for the research questions is given. Two sections dedicated to answering the research questions follow. The article concludes with a discussion of the most important findings and a conclusion.

II. RELATED WORK

The number of research articles dedicated to RE for AI and ML systems is relatively small to date, as noted in [10] and [12] also. Furthermore, in their review Martínez-Fernández et al. [13] have identified only one article, i.e., [10], that covers the whole RE process. For these reasons, this section includes articles that are not entirely dedicated to RE for AI or ML systems but which mention this process as part of the broader software engineering process they analyze.

Belani et al. [14] discuss the RE challenges in the development of systems which the authors call AI-based complex

systems. The article presents the RE4AI taxonomy of challenges related to recognized elements of AI (data, model, and system), which are aligned with the typical RE activities. Vogelsang and Borg [10] present the findings from interviews with four data scientists on their experience with RE activities in the development of ML systems. Among the rest, the authors conclude that requirements engineers should be aware of the new quality requirements and integrate the ML specifics in the established RE process. Chuprina et al. [15] describe their ongoing work on an artifact-based RE method for data-centric systems, i.e., systems that include both AI and ML systems. The authors state that these systems require a new approach to RE and define a conceptual model of artifacts, contents, and relations that should guide the RE process. Heyn et al. [16] identify four challenging RE areas in the development of systems which the authors call AI-intense systems, i.e., systems that fundamentally depend on AI functionalities. The four areas include (1) defining requirements for the context in which the system would operate, (2) defining quality attributes and data requirements, (3) defining performance metrics and monitoring if the system has the guaranteed behavior, and (4) gaining an understanding of the human factors that influence the user acceptance and trust.

Studer et al. [17] extend the CRISP-DM data mining process model to address the specifics of the ML development process. This new process model consists of six phases, i.e., (1) business and data understanding, (2) data preparation, (3) modeling, (4) evaluation, (5) deployment, and (6) monitoring and maintenance. The authors provide a description of the RE activities throughout the phases. While Kästner and Kang [12] describe a course on software engineering for systems which the authors call AI-enabled systems, they also mention software requirements as one of the stages in the software engineering life cycle. In that context, the authors emphasize the lack of specification for AI components, the importance of identifying and measuring quality requirements beyond model accuracy, the importance of defining safety and security requirements, as well as the importance of properly planned error handling. Zhang et al. [18] have surveyed 195 DL practitioners to identify software engineering challenges in DL application development. The authors present 13 findings that reveal the challenges in different development phases, and 7 improvement recommendations. Requirement analysis, integration testing, acceptance testing, and problem definition are identified as the most labor-consuming tasks throughout the process. Requirement analysis is recognized as a more difficult task in DL applications than in conventional ones. Kuwajima et al. [19] study the open problems in engineering safety-critical ML systems, particularly in terms of ML model/system requirements, design, and verification. The authors conclude that ML models are characterized by a lack of requirements specification, design specification, interpretability, and robustness. Through gap analysis of standard quality models and ML model characteristics, they conclude that the lack of requirements specification and robustness have the greatest impact on those models. Rahman et al. [20] present a project which uses ML in detection and correction of transaction errors. In terms of RE, the authors emphasize the need for iterative refinement of the requirements since they evolve frequently. They further emphasize the importance of properly conducted feasibility analysis of ML systems in relation to the available data, as well as the importance of identifying the data requirements before any large data acquisition. Wan et al. [21] present analyses of information obtained from 14 interviewees and 342 survey respondents from 26 countries. The authors' analyses reveal significant differences between the ML and non-ML software development process at different stages (e.g., requirements, design, and testing). Some of the differences related to RE include the need for preliminary experiments while collecting the requirements for ML systems, the greater uncertainty of the requirements, and the need to anticipate any potential performance degradation. Giray [22] presents an overview of research articles on software engineering for ML systems. In terms of RE, the author points to the challenges with proper management of customer expectations, with the requirements elicitation, analysis, and specification, with the new quality attributes, and the new types of requirements such as data requirements. The author suggests that future research should focus on improving the alignment of performance metrics with business objectives, proper integration of the requirements for ML and non-ML components, risk assessment frameworks, and data privacy regulations. Martínez-Fernández et al. [13] provide a review of 248 articles on software engineering for AI systems, of which 17 are dedicated to software requirements. The authors conclude that many of the latter focus on quality attributes, several deal with specification approaches, and only one offers a holistic view of the RE process. They point to the software requirements as one of the underrepresented areas in the entire set of articles, with great potential for further research. In terms of quality, they emphasize that standards developed for conventional software systems should be updated. Serban and Visser [23] analyze software architectures that enable robust integration of ML components through a systematic literature review, interviews, and a survey. They identify RE challenges such as (1) the difficulty in understanding the project and estimating the effort in advance, (2) the difficulty in defining functional requirements for ML components, and (3) the potential regulatory restrictions. Pereira and Thomas [24] analyze the safety challenges in the development of ML-based cyber-physical systems. In terms of RE, the authors indicate that while high-level requirements can be defined explicitly, the low-level requirements are defined implicitly through the dataset, making the requirements traceability inapplicable. They suggest specifying requirements for data management, model development, model testing/verification, and model

deployment. The potential risks include incomplete data definition, incorrect loss function, wrong performance metrics, incompleteness of the testing process, and inadequacy of the safe operation values. Lorenzoni et al. [25] summarize the software engineering practices and challenges in developing ML models. Through their review of research articles, the authors have found an evident lack of techniques related to RE for ML models. Lwakatare et al. [26] present a taxonomy of engineering challenges related to commercial systems containing ML components. Through several case studies, the authors have identified five stages in the evolution of ML components, from an experimental stage to autonomous functioning. Some of the presented challenges, which, in our opinion, are related to RE, are those associated with the problem formulation and the desired outcome specification in the experimental stage, as well as with the failure to evaluate models with business-centric metrics in the noncritical deployment stage. Maass and Storey [27] analyze if ML could benefit from conceptual modeling. Additionally, the authors outline specification languages useful in specifying various types of requirements for ML systems. Villamizar et al. [28] present a catalog of 45 concerns related to ML systems that should help requirements engineers in defining requirements for such systems. The concerns cover five perspectives, i.e., objectives, user experience, infrastructure, model, and data. In a second research article, Villamizar et al. [29] propose an approach for analysis and specification of the five perspectives of ML systems outlined in [28]. The authors provide a diagram of ML tasks and concerns, as well as a specification template. Pei et al. [30] review research articles published from 2016 to 2022 on RE-related collaboration challenges occurring between the different roles involved in ML development. The authors summarize the solutions proposed in the reviewed literature and give an example from the industry. Ahmad et al. [31] present a systematic mapping study of 43 primary studies on RE for AI. The authors analyze (1) the methodologies used in specifying requirements for AI-based software, (2) their limitations, (3) the evaluation method which the primary studies use, and (4) the application domains. The authors also provide recommendations for future research. Ahmad et al. [32] also analyze human-centered approaches in RE for AI software. Their (1) analysis of industry guidelines for AI software and (2) survey of industry practitioners have revealed the current practices and gaps. Jahic et al. [33] propose a textual domain-specific language that facilitates the specification of data requirements and necessary ''recognition skills'' the neural networks should acquire through their training. Through an example, the authors show the benefits of the proposed approach. Through a literature review, De Hond et al. [34] outline guidelines and quality criteria for development and evaluation of AI models for healthcare. The guidelines include many aspects relevant to RE, such as understanding the problem and its context, quality requirements, risk management planning, and similar.

Several research articles focus on non-functional requirements (NFRs) and quality-related aspects of AI/ML systems. Pons and Ozkaya [35] summarize the unique characteristics of several quality attributes of AI systems that are used by the public sector, i.e., security, privacy, data-centricity, sustainability, and explainability. Horkoff [36] outlines a set of challenges associated with NFRs for ML systems, as well as research directions to solve them. The author states that the current knowledge of NFRs should be at least partially rethought in the context of ML, because although many techniques related to NFRs for non-ML systems are still valid, some need adjustment or complete renewal. Kuwajima and Ishikawa [37] analyze the quality attributes relevant to AI systems. The authors try to identify what needs to be modified or added to quality standards for them to be adapted to the ML specifics and the Ethics Guidelines for Trustworthy AI from the European Commission [38]. Siebert et al. [39] present a process for constructing quality models for ML systems, describe the elements of the process, and present a use case from the industry. The authors conclude that some of the existing quality attributes relevant to conventional software systems should be redefined, and new ones relevant to ML systems should be added. Nakamichi et al. [40] propose a requirements-driven method for deriving quality attributes for ML systems. They extend conventional quality attributes with those relevant to ML systems and describe a method that allows deriving quality attributes and measurements dependent on ML systems' goals. Habibullah and Horkoff [41] present findings from interviews with ML industry practitioners regarding ML-relevant NFRs, their measurement, and challenges. The authors conclude that the NFRs for ML systems are neither well structured nor well documented, their measurement is challenging, and although important, their consideration in ML systems is still at an early stage. In a journal article, Habibullah et al. [42] extend these findings by analyzing the importance of different NFRs, their associated challenges, and the different perception of NFRs that exists between practitioners from industry and academia. Habibullah et al. [43] present an exploratory study on the definitions of NFR relevant to ML systems, their shared characteristics, and past research interest in each NFR. The authors conclude that the research interest in different NFRs differs significantly, and they manage to identify six clusters of NFRs sharing similar properties and purpose. Hu et al. [44] address reliability requirements for machine vision components by defining relevant image transformations, classes of reliability requirements, a method for instantiating requirements of each class of reliability requirements using human performance data, and, finally, a method to verify that components satisfy such requirements. The requirements are defined as a tolerated range of visual changes which should not affect the component behavior.

As mentioned at the beginning of this section, a small number of research articles cover the challenges imposed by ML specifics throughout the whole RE process, as it is

done in this article. Compared to [10], which analyzes RE challenges through interviews with data scientists, our article does so through a literature review and a case study. Several research articles [18], [21], [23], [25] cover the challenges associated with the various software engineering activities during ML software development. However, in our opinion, the challenges related to RE activities are not covered as extensively as in our article. Two research articles [13], [22] provide an in-depth overview of the challenges associated with the different stages of ML software development, including those related to RE. Compared to the referenced articles, our article (1) summarizes the challenges not only in terms of the conventional RE activities but also in terms of a variety of conventional types of requirements, (2) presents insights into the reasons for the relevance of those RE activities and types of requirements for ML systems, (3) provides a brief overview of the most relevant definitions and trade-offs for a set of ML-specific quality attributes, and (4) shares our experiences in dealing with those challenges in a real ML project, along with excerpts from its requirements specification. Our article also differs from [31] in the research questions it answers and the method it uses to answer them. Namely, our article (1) focuses particularly on the challenges introduced in the conventional RE process by the ML specifics and on the ways to address them, (2) systematizes them by conventional RE activities and a large set of conventional requirement types, (3) reviews research articles which may not be explicitly devoted to RE for ML, but are implicitly related to a RE activity or requirement type (e.g., articles related to risks, limitations, success metrics, assumptions, constraints, various quality attributes of ML systems), therefore, in our opinion, it gives a broader overview of the topic, and, finally, (4) shares our practical experience in dealing with those challenges through a case study.

III. METHOD

This article answers the research questions through (1) a review of literature in the fields of RE and ML and (2) a case study. The article is organized according to the conventional RE activities and software requirements. Most sections begin with a short definition of the RE activity or software requirement to which they are dedicated, continue with a brief review of the ML domain literature relevant to the activity/requirement, and end with experiences from the case study. The following two sections describe the methodology and its limitations.

A. RESEARCH ARTICLES REVIEW

The conventional RE activities and software requirements were analyzed through well-known publications from the RE domain (e.g., [45]). The impact on the conventional RE activities and software requirements in ML projects was analyzed through a review of previously published research articles that were identified using Google Scholar[1] through the search criteria given in Table 1, in the period of June-July 2021. The article search was repeated in April 2023 to find relevant articles published after the initial search. A description of the process follows.

TABLE 1. Search queries used in identifying relevant research articles.

	Search Criterion	Search Queries
1	General	(machine learning | artificial intelligence) (requirements engineering | software requirements)
2	Requirement types	(machine learning | artificial intelligence) (user | stakeholder | functional | non-functional | quality) requirements
3	ML-specific concepts	(machine learning | artificial intelligence) (ethics | trustworthiness | engineering | challenges | metrics)
4	ML-specific qualities	(machine learning | artificial intelligence) (interpretability | explainability | fairness | robustness | security | privacy | safety)

The initial attempt to identify previously published research articles relevant to our research questions was based on search criteria 1 and 2 in Table 1. However, the query results mainly consisted of articles devoted to the use of ML methods to facilitate RE, which is irrelevant to this article. One of the reasons for such results could be the small number of research articles devoted to RE for ML at the time of searching. Another reason could be the use of inconsistent terminology for certain RE activities or software requirements, like (1) the use of synonyms for the term ''requirement,'' (2) the disagreement over the naming of certain RE activities, e.g., ''requirements validation'' over ''requirements verification'', further discussed in [45], or (3) the disagreement over the nature, terminology, and definition of the non-functional requirements, further discussed in [46]. A third reason could be the significant difference between the conventional and the ML software development process, leading to a potential terminological inconsistency of the second one with the first. The potentially relevant articles were initially selected based on their title and abstract, taking into account only journal, conference, conference workshop articles, and preprints (available on arXiv[2]), all written in English. These initially selected articles were then analyzed more thoroughly from our side. The articles in the final selection were not necessarily dedicated to RE for ML or AI systems in their entirety but contained findings on the subject. Since the number of selected articles was again small, the references and the articles which cited those articles entirely dedicated to RE for ML or AI (e.g., [10], [36]) were analyzed in the same manner to identify other relevant articles. Finally, the selected articles were used to extract and synthesize the answers to the research questions. This process is illustrated in Figure 1.

The search criterion 3 in Table 1 allowed us to identify influential articles in specific sub-fields of ML. Although not explicitly dedicated to RE for ML, some of these articles

[1] https://scholar.google.com/

[2] https://arxiv.org/

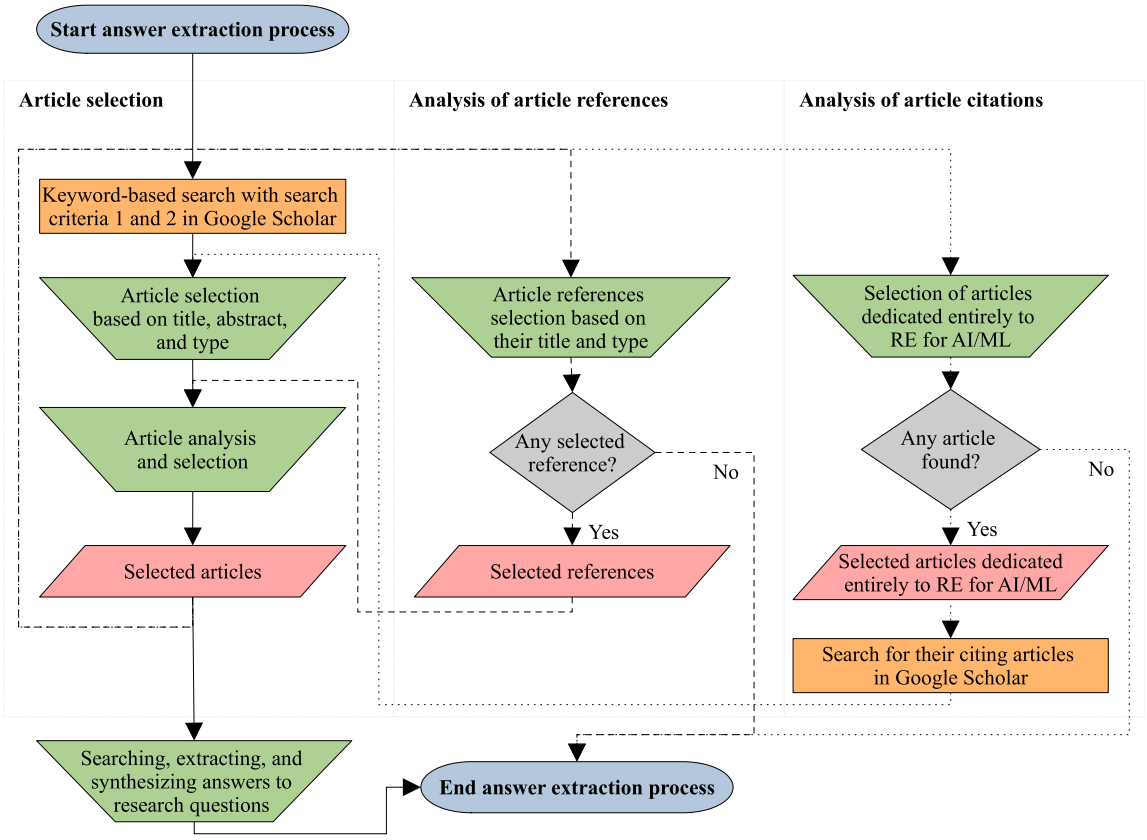

FIGURE 1. General flowchart of the literature review process (search criteria 1 and 2).

contain findings that, in our opinion, should be considered during RE in ML projects. These findings include new types of implementation, ethical or trustworthiness risks, new types of success metrics, assumptions, limitations, and similar. Due to the broadness of the sub-fields this search criterion covers, the articles were selected based on our estimation of their usefulness in answering the research questions. A thorough review of these sub-fields is out of the article's scope, and therefore, throughout this article, we only briefly summarized the findings we considered important. This process is illustrated in Figure 2.

A widely accepted classification of quality attributes relevant to ML systems does not exist at the time of writing, although certain research articles address this challenge (e.g., [37]). Therefore, the research articles dedicated to ML-specific quality attributes were identified through search criterion 4 in Table 1, but this list of quality attributes should not be considered a complete one. Section V-D summarizes the findings we found relevant to RE from a selected set of articles dedicated to each quality attribute, regardless of their mentioning of RE related terminology, since the RE literature indicates that elicitation, prioritization, and specification of quality requirements in a specific, measurable, attainable, relevant, and time-sensitive manner falls in the domain of RE [45]. More recent review articles, which have a large number of citing articles, were prioritized in our selection process. Their references were used to find articles that provide definitions/insights into the relevant quality attributes as well. Furthermore, a brief summary of some of its trade-offs with other quality attributes was compiled for each quality attribute. This way, we tried to emphasize the importance of those quality attributes to ML systems, emphasize the consequences of giving them insufficient attention during the RE activities, and provide the reader with valuable references for further reading. This process is illustrated in Figure 2.

Finally, despite our efforts to identify and include in our review as many of the previously published research articles relevant to RE for ML systems as possible, due to the aforementioned challenges and the volume of articles in certain ML sub-fields (e.g., certain quality attributes), relevant articles may still be missing.

B. CASE STUDY

The object of our case study is an ML system, Academic Disciplines Detector (ADD), which detects concepts defined as academic disciplines by the community editing Wikipedia, based on textual excerpts from their Wikipedia articles and their similarity to the academic disciplines that are part of expert-created classification systems [11]. As an example of an integrative ML system, incorporating several custom-trained and third-party ML models in its core functionalities while attempting to solve a real-world challenge,

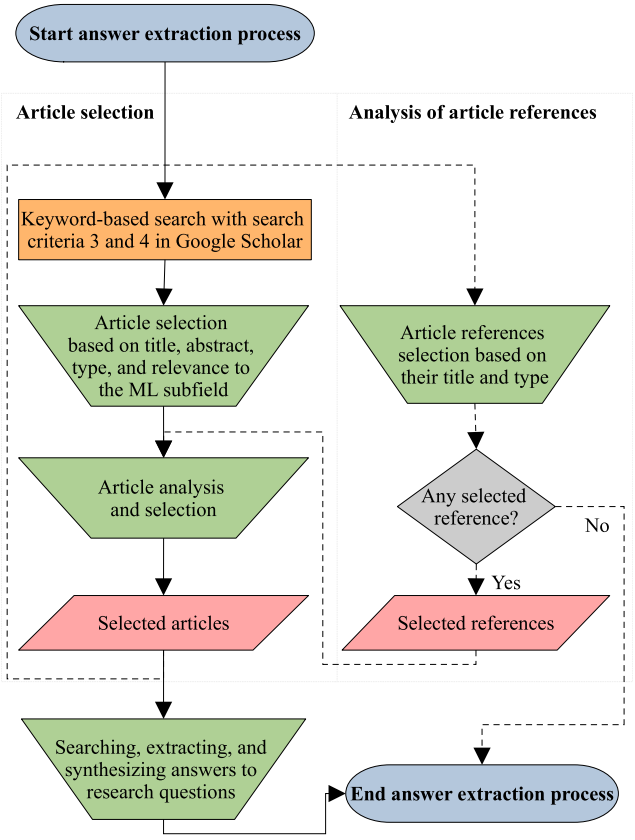

FIGURE 2. General flowchart of the literature review process (search criteria 3 and 4).

we believe that ADD is a suitable object of our case study. Although the inclusion of a single case study may be considered a limitation, we believe that our experience can still be helpful in analyzing the RE challenges in ML projects.

IV. REQUIREMENTS ENGINEERING ACTIVITIES IN MACHINE LEARNING PROJECTS

Requirements engineering covers the activities related to (1) requirements development (requirements elicitation, analysis, specification, and validation) and (2) requirements management, which are inevitable activities in any project regardless of its approach to software development (e.g., waterfall or agile) because they give reassurance that the problem is properly understood and resolved [45]. This section analyzes the research questions related to the relevance of conventional RE activities to the ML development process, the challenges this process brings to the activities, and their necessary adjustments to better fit into this process.

A. REQUIREMENTS ELICITATION AND ANALYSIS

1) LITERATURE REVIEW

As with all other software systems, the success of production-level ML systems depends primarily on their fulfillment of specific business goals or end-user needs. Goodfellow et al. [4] indicate that the definition of goals and performance metrics, as a first step towards successful practical application of ML, should always be guided by the problem to be solved. In that sense, any software development project begins with activities that provide a proper understanding of the problem to be solved, the factors that have motivated the project, and the context in which the system would be used. In conventional RE, the requirements elicitation is the process through which the stakeholders' needs and constraints are identified, and it is intertwined with the requirements analysis and requirements specification activities [45].

Identification of all relevant stakeholders is inevitable for a successful elicitation of the requirements. However, in ML systems, the requirements may depend not only on the stakeholders' needs but on the available data as well. In that sense, data scientists assess the feasibility of the stakeholders' requirements through analysis and experiments, so, as Vogelsang and Borg [10] indicate, they are important stakeholders to be consulted during the requirements elicitation. Certain stakeholders may have unrealistic expectations of the ML systems' performance, adoption process, or functionality, so they should be helped in making their targets more reasonable, as well as in accepting the uncertainty of the time and cost estimates [13]. Stakeholders should be aware that despite its enormous potential, ML introduces nontrivial challenges to the software development process, which can sometimes make it a less suitable (e.g., in terms of interpretability) or a more expensive option (e.g., in terms of time/resources) than other available options. For example, while DL stands out in solving closed-end classification problems with sufficiently large training datasets and test datasets that closely resemble those from training, any deviation from these assumptions or misunderstanding of DL limitations can be a source of problems [47]. Supervised DL algorithms may require at least 10 million labeled examples to achieve or exceed human performance [4], which can hardly be obtained in certain domains. Furthermore, Martínez-Fernández et al. [13] bring attention to the applicability of research results in practice because sometimes they can oversimplify reality and be inapplicable in real conditions. In short, the decision to implement an ML-based solution to a problem should be based primarily on the outcome of the problem-specific analyses.

In addition to the stakeholders' requirements related to the system functionality, it is essential to understand their requirements related to the system quality attributes. For example, these include their interpretability requirements, and when less interpretable classes of models are taken into consideration, their requirements for the system output explainability, as further elaborated in Sections V-B and V-D1. Furthermore, it is important to properly collect the stakeholders' security, privacy, and safety requirements, as well as to identify potential sources of bias that may lead to a discriminatory outcome for a particular group of individuals. Therefore, legal experts are another important group of stakeholders to be consulted during requirements elicitation [10].

D'Amour et al. [48] also highlight that real ML systems typically have behavioral requirements that go beyond generalization to an independent and identically distributed test dataset (e.g., requirements on interpretability, fairness). When those relevant requirements are not well specified and enforced on the ML pipeline, i.e., are ''underspecified'', many near-optimal solutions that fit such incomplete specification but behave differently in different dimensions (e.g., the previously mentioned interpretability or fairness) may exist and be selected over the desired one, which can be a cause of failure in applied ML [48]. In addition, deep neural networks are sometimes prone to learning undesirable ''shortcut'' solutions to problems, i.e., decision rules that perform well on independent and identically distributed test data but fail on out-of-distribution data (data that may be closer to real data) [49]. Avoiding such solutions, therefore, requires a thorough understanding of what makes a particular solution easy to be learned in a given context, the impact of the various factors throughout the ML pipeline, and their interactions [49].

2) CASE STUDY

The ADD project was motivated by the importance of the established disciplinary system to society and the challenges in tracking its changes over time. Our previous work [50], [51] had made us aware of these challenges, so we hypothesized that integrating different data sources into a data-driven methodology could be helpful in addressing them. Additionally, we identified a gap in the field related to the use of Wikipedia and the new ML breakthroughs. Neither a detailed study of Wikipedia's potential in this field nor a study of the potential of those ML breakthroughs when applied to sufficiently large domain-specific datasets was available. Therefore, we hypothesized that if used appropriately, Wikipedia could provide large amounts of data to maximize the capabilities of ML algorithms in studying the disciplines, their relations, and evolution over time [11]. All of the above made ADD conceptually and methodologically different from the similar methodologies proposed in research articles (for more details, see [11]).

Due to the research nature of the ADD project, the requirements elicitation was mainly done through individual activities, e.g., analysis of available classification systems of academic disciplines, reading related literature, and analysis of Wikipedia's policies. To better understand how to make ADD as useful as possible to the communities for which it was intended, the characteristics of its stakeholders were identified first. The stakeholders were classified into four classes, (1) team member, (2) research community member (Knowledge Organization (KO) and related fields), (3) research community member (ML and Natural Language Processing (NLP)), and (4) data consumer. Given the fact that we did not have direct representatives of some of the stakeholder classes, studying their characteristics through imaginary personas [45] helped us better define their (hypothesized) needs and requirements. For example, the research communities for which ADD was intended were divided into two classes due to the hypothesized differences in their interests and level of knowledge in the fields that ADD covered. While we expected that the KO research community members have an extensive knowledge of the achievements related to academic disciplines detection, we did not expect this to be the case with the members of the ML and NLP communities. On the other hand, the latter communities were expected to have greater familiarity with the ML and NLP methods used in ADD, which, among other things, implied different presentation clarity expectations by the different communities. Furthermore, we expected that the ML and NLP community members would be mainly interested in the comparison of the state-of-the-art text encoders based on deep neural networks to conventional text analysis methods on a new domain-specific dataset, while the other communities in the comparison of the detected academic disciplines to previously published results. A separate stakeholder class interested in the end results but with limited knowledge of the technical aspects of ADD was called data consumers.

The analysis of some of the available academic discipline classification systems and Wikipedia's policies resulted in a number of insights that were later incorporated into the functional and non-functional requirements. Several examples include Wikipedia's policies on article titles, lead sections, and life cycle, as well as the ML domain recommendations for imbalanced dataset evaluation metrics (for more details, see [11]). The initial requirements were further refined in the data analysis and experimentation phases, e.g., through analysis of Wikipedia's article titles, article interlinks, category graph, and similar. In addition to refining the already identified requirements, new requirements were discovered in these phases, such as requirements related to data preprocessing.

B. REQUIREMENTS SPECIFICATION

1) LITERATURE REVIEW

In conventional software development, the approach to formal specification of the requirements largely depends on the selected approach to software development, with best practices already in place for each. We are not aware of best practices defined for ML systems specifically. Maass and Storey [27] indicate that the field of conceptual modeling already has proven specification languages for functional, non-functional, and business requirements. Data requirement should use those already available for database systems and linked data, whereas the approaches to specifying performance, ethical, interpretability, and resilience requirements require further refinement [27]. Adaptation of formal methods has been proposed as another possible direction for designing AI systems with provable correctness against mathematically specified requirements [52]. Model-driven engineering principles have also been used in specifying requirements for neural networks [33]. Through a literature review, Ahmad et al. [31] have found that the most

commonly used modeling notations or languages in specifying requirements for AI systems are (1) Unified Modeling Language (UML), (2) Goal-Oriented RE (GORE) and (3) Domain Specific Models, but the authors conclude that they still have limitations when it comes to their use for this purpose.

2) CASE STUDY

The requirements specification approach used in the ADD project was mainly based on best practices from conventional RE, tailored to ML specifics when necessary. A combination of text and visual models was used. Excerpts from the ADD requirements specification are given in Section V.

C. REQUIREMENTS VALIDATION

1) LITERATURE REVIEW

The requirements validation ensures that the right requirements which meet the needs of the stakeholders are captured, and it is performed through activities such as requirements reviews, development of conceptual tests, definition of acceptance criteria and similar [45]. The use of ML affects this RE phase as well, especially in terms of testing approaches. Riccio et al. [53] point to a limitation in the effectiveness of conventional testing approaches when applied to ML systems, primarily because of the program logic dependence on training data and the stochastic nature of the learning process. The authors emphasize the need for novel techniques that address the specifics of ML systems. Since the testing of ML systems is a rapidly progressing research field at the moment, readers are directed to Riccio et al. [53] and Zhang et al. [54] for a more thorough review of the challenges and novel methods.

When it comes to measuring DL models' performance, Geirhos et al. [49] show that measuring performance only on an independent and identically distributed test dataset can sometimes be misleading if the assumption that the data generation and sampling mechanisms are the same is not justified. The authors suggest that testing on out-of-distribution data should become a standard practice in order to distinguish desired solutions from "shortcut" solutions [49]. Furthermore, the results presented in [48] indicate that models should be explicitly tested for any required behavior that is not guaranteed by the independent and identically distributed test dataset, as some required behavior will almost certainly be underspecified. These tests should be application specific and based on the requirements [48].

2) CASE STUDY

The validation of the requirements for ADD was done through reviews, development of test cases for conventionally programmed software components, planning the ML-based components testing, and defining criteria that the components and the system had to meet. The planning of the ML-based components testing included defining criteria for collecting representative training/test datasets, identifying types of potentially ambiguous examples that may cause incorrect predictions (e.g., scientific terminology or notable people related to a particular academic discipline), identifying exceptions (e.g., articles on academic disciplines that do not comply with Wikipedia's policies), defining approaches to test model performance changes over time, and similar [11].

V. REQUIREMENTS FOR MACHINE LEARNING SYSTEMS

This section analyzes the research questions related to the relevance of different types of software requirements in addressing the ML specifics, the impact ML specifics have on their conventional understanding, and the necessary adjustments of this understanding. In conventional software systems, the requirements can be organized into multiple levels of abstraction, where the lower levels refine the higher ones. For example, Wiegers and Beatty [45] suggest a three-level model of requirements consisting of business, user, and functional requirements, accompanied by non-functional and data requirements. This section analyzes some of the well-known types of requirements in the context of ML systems, and uses this model to a limited extent in organizing its subsections (for the exact three-level model of requirements, the readers are referred to the referenced publication). Nevertheless, the analyzed requirements in this section are only a subset of the broader set of requirements and relevant information suggested by the RE literature to date. Furthermore, through the organization of the subsections, we do not attempt to suggest a particular approach to organizing the requirements in requirements specifications, so we direct the readers to publications and standards written for that particular purpose.

A. HIGH-LEVEL (BUSINESS) REQUIREMENTS

The business requirements, usually coming from stakeholders familiar with the reasons for undertaking the project, refer to the needs that initiated the project and the desired outcome [45]. A description of some of the information that may be part of these requirements follows, analyzed in the context of ML systems. It is supplemented with excerpts from the ADD requirements, given in Table 2.

1) OBJECTIVES

a: LITERATURE REVIEW

Defining the objectives to be achieved through the use of a particular software system is an essential factor for the success of the project, regardless of whether it involves ML or not. The specificity of ML systems in terms of their potential impact on individuals, groups, and even society, requires defining objectives aligned with the already recognized ethical principles for this type of systems by the community. In the ethical guidelines and principles for AI systems published recently by the public (e.g., Ethics Guidelines for Trustworthy AI [38]) and private sector (e.g., Google AI Principles[3]), convergence towards several ethical principles has been observed, i.e., transparency, justice and fairness,

[3]https://ai.google/responsibility/principles/

non-maleficence, responsibility, and privacy, together with beneficence, freedom and autonomy, trust, dignity, sustainability, and solidarity [55]. Although some divergence of the ethical principle definitions and uncertainty about their implementation in practice has been highlighted in [55], we firmly believe that the aspiration for development of ethical ML systems based on recognized principles has to be clearly and unambiguously stated in the high-level requirements that guide the project. Consequently, these principles have to be included in the low-level requirements, incorporated in the development practices, implemented in the system, validated, and monitored.

b: CASE STUDY

An excerpt from the objectives of ADD is given in Table 2. Some of the listed objectives exactly refer to the development of a thoroughly evaluated system, transparent in terms of its methodology.

2) SUCCESS/PERFORMANCE METRICS

a: LITERATURE REVIEW

The ML community has defined various performance metrics appropriate for different types of ML problems, such as accuracy, precision, recall, f-measure, mean squared error, and others. Performance is usually measured on a dataset unseen during the model training stage to ensure proper functioning of the model on unseen data in a real-world setting. Nevertheless, in certain tasks it may be challenging to find an ML performance metric that corresponds to the desired system behavior, or measuring that behavior may be impractical [4]. In addition, a preferred and realistic level of performance that makes the ML system worthwhile, safe, and useful has to be determined [4]. It is recommendable to document the approaches to uncertainty and variability, e.g., k-fold cross-validation, as well [9]. Some articles [10] indicate that it is the requirements engineer's job to translate the customer expectations to appropriate metrics.

b: CASE STUDY

ADD success metrics were defined at two levels of abstraction, i.e., (1) high-level success metrics that refer to the system's overall success in achieving its objectives and (2) ML performance metrics specified for each ML component separately, together with the expected performance. An excerpt from the high-level success metrics for ADD is given in Table 2. The high-level success metrics indicate that the number of detected academic disciplines should be similar to that in expert-created classification systems. They also require processing of multiple Wikipedia exports over a period of four years, in order to demonstrate the low variability of the test performance, and the high overlap of the detected disciplines in adjacent processed exports. Examples of ML performance metrics, along with reasons for selecting them over others, are given in Table 3 and Section V-C.

3) LIMITATIONS

a: LITERATURE REVIEW

ML models are trained and tested under certain assumptions and conditions. Therefore, it should not be assumed that they work equally well in other settings. As the ''no free lunch'' theorem for ML [56] states, no algorithm is universally better than any other, including random guessing, when averaged over all possible tasks [4]. For example, the limitations of DL models, summarized in [47], indicate that they have poor performance when their training data is limited, when their test data differs significantly from their training data, as well as in broad example spaces filled with novelties. The quality attribute definitions have their own limitations as well [57]. Given these facts on the one hand, and ML systems' potential effects on individuals (or even autonomy in some cases) on the other, it is essential that the limitations of ML systems are clearly stated, communicated to its stakeholders, and agreed upon. Communicating clearly what the system outputs mean and what they do not, what the intended and unintended use cases are, helps in avoiding misinterpretations or inappropriate use. As Jacovi et al. [58] highlight, vaguely specifying the expected behavior of an AI system, which users should trust to be upheld (called a ''contract'' by the authors), can lead to unwarranted trust in the system and its misuse, as users may implicitly assume ''contracts'' that during the development of the system have not been considered to be upheld.

b: CASE STUDY

An excerpt from the limitations of ADD is given in Table 2. The limitations in the excerpt primarily refer to the interpretation of the system output, i.e., what the output means and what it does not, and its proper use, i.e., intended and unintended use cases.

4) RISKS

a: LITERATURE REVIEW

Risks are conditions that should be identified, evaluated, and controlled, because they can negatively affect the success of a project in terms of user acceptance, implementation, competition, and similar [45]. ML systems face risks that are not inherent in conventional software systems, like specific ethical, moral, legal, security, and other similar risks. While AI algorithms have the potential to augment human well-being, at the same time, they can sometimes exhibit behavior with unintended and unanticipated consequences by their creators, both positive and negative [8]. As their properties and operating environments become too complex to allow an analytical formalization of some of their behaviors, predicting their effects on individuals and the society becomes challenging as well [8]. In that sense, anticipating any potential risks from the influence these systems have on people and the other way round, although absolutely necessary, can be rather challenging. This section summarizes some of the risks specific to ML systems, like the implementation challenges that may turn into risks. An additional discussion on the

risks associated with various quality attributes is available in Section V-D.

Engineering robust ML systems has specific challenges that are not inherent in other types of software systems. ML models are highly sensitive to changes in their input data distribution and learning hyperparameters, and such changes may lead to model retraining, further affecting all of its dependent models in a way that cannot always be predicted [6]. They may depend on data from external systems or models, changes of which may be beyond our control, and be sensitive to changes in the environment which they interact with [6]. Inadequate model update frequency in frequently changing environments can be a risk factor to its performance, as can failing to evaluate the model performance on an important data slice, especially if it differs from the overall performance [59]. Reproducibility, debuggability, and auditability are important aspects that require version control of the model specifications [59] and tracking of the data on which the model was trained, but proper data management and versioning are more complex than doing the same for software code [7]. ML systems face specific security, privacy, and safety risks that must be adequately addressed because of their potential consequences. The lack of interpretability or explainability is another risk factor to the stakeholders' trust and acceptance of the system. In the context of DL, many cases of failure can be attributed to so-called "shortcut" solutions [49]. Underspecifying relevant behavior to be learned by the ML pipeline can lead to such "shortcut" or otherwise undesirable solutions, because the ML pipeline can choose one such solution over another which has the same test performance and much greater compliance with the "unspecified" but desired behavioral requirements [48], [49].

b: CASE STUDY

An excerpt from the risks of the ADD project is given in Table 2. The risks in the excerpt refer to the potential nonacceptance of the system by the users (due to its significant differences with previously published systems and methodologies), as well as to the difficulties to precisely define the ground truth in the evaluation process (due to imprecisely defined domain-specific terminology and nonexistence of a widely-accepted finite set of academic disciplines).

TABLE 2. An excerpt from the high-level requirements for ADD. The implementation of the requirements is described in [11].

Objectives
1. Develop a thoroughly evaluated and accurate methodology for detecting academic disciplines based on state-of-the-art methods for text representation at the time of development.
2. Make a detailed qualitative and quantitative comparison of state-of-the-art methodologies for text representation and conventional bag of n-grams methods on the task.
3. Make the methodology transparent by publishing the source code of the core system modules under appropriate license(s).
Success/Performance Metrics
1. The number of detected academic disciplines is comparable to their number in an expert-created classification system.
2. The performance on the test dataset does not vary by more than 1-2% between the processed Wikipedia exports over a period of four years.
3. At least 90% of the academic disciplines detected in a Wikipedia export overlap with those detected in the nearest previous and subsequent Wikipedia export processed by the system.
Limitations
1. The thematic structure ADD detects is an academic discipline, and any other thematic structure is out of scope.
2. ADD does not suggest any sort of ordering or subordination of the academic disciplines it detects.
3. ADD is an experimental software. Therefore, its results should not be used as a substitute for expert-created academic discipline classification systems, but only as a source of data that could complement them.
Assumptions
1. The scientific disciplines are a subset of the academic disciplines.
2. The academic discipline is a more generic thematic structure than the scientific/research field, topic, or term.
Risks
1. Domain-specific terminology that lacks a precise definition, e.g., academic/scientific (sub)discipline, research field, or (sub)specialty. Users may have a different understanding of the core thematic structure detected by ADD, regardless of the provided definitions.
2. Unavailability of a precisely defined ground truth, i.e., a finite set of widely recognized academic disciplines, which could be used for an accurate evaluation of the methodologies for detecting academic disciplines.
3. Acceptance of the system and its methodology by the users for which it is intended, considering the significant conceptual differences with the competing systems and methodologies.

B. USER REQUIREMENTS

1) LITERATURE REVIEW

Since ML systems may have user interactions that affect users and their acceptance of the system, collecting the user expectations from such systems can reveal useful interactions and quality requirements. For example, studies have shown that the perception of an ML system interpretability depends on the audience to which the explanations are presented and the task [60], [61]. Although different types of post-hoc explanations may be appropriate to different end users in different tasks (e.g., textual explanations, visual explanations, local explanations, explanations by example, and etc., further discussed in [60]), it is essential that they are aligned with the user mental models, needs, and use cases [62]. Amershi et al. [63] state that in many ML systems, their users have been able to come up with new possibilities for explanations, other than the ones they have received. Heyn et al. [16] emphasize the importance of understanding user needs and interactions with the system during RE, in order to provide users with functionalities they would accept, trust, and use properly.

Nevertheless, in the more general context of business analytics projects, Wiegers and Beatty [45] state that elicitation of the user expectations from such systems is insufficient to reveal the complex knowledge needed to develop them. The same is true for ML systems. Moreover, features in ML systems are introduced not only as a result of user needs, but for other reasons as well, like the availability of certain data, the need to collect additional data through user interactions,

and similar [64]. These systems may provide outcomes based on user data, and their behavior may evolve as they collect more data, so in this context, Yang et al. [65] distinguish four types of AI systems based on two factors, i.e., their capability uncertainty and output complexity. While the first type has bounded capabilities and a fixed set of outputs, the fourth has evolving capabilities and adaptive open-ended outputs, making it difficult to predict what the fourth type of AI systems can reliably do, when they can fail or how likely the failures are, in order to plan appropriate interactions [65].

2) CASE STUDY

Due to the experimental nature of ADD and the characteristics of its users, it does not have a user interface, but users interact with the application by running its modules with a set of required parameters, after providing them with the necessary files [11]. Users receive the results in local files generated by the modules. In this sense, we were able to identify most of the usage scenarios that involve different classes of users. However, due to the inherent uncertainty of ML models, it is still possible to have model outcomes or failures that have not been anticipated.

C. FUNCTIONAL REQUIREMENTS

1) LITERATURE REVIEW

In general, functional requirements describe what a software system should be capable of doing. Typically, the expected behavior of conventional software components is precisely specified in the functional requirements. This is not the case with ML models, which learn how to relate the input data to the expected outcome through a training process. Nevertheless, ML systems commonly consist of both conventionally programmed functionalities and functionalities implemented by ML models. In that sense, certain functional requirements are defined conventionally, by explicitly specifying the rules that relate inputs to outputs. At the same time, those functionalities that require training an ML model are described through the function that the model is expected to learn and the expected performance. Kuwajima and Ishikawa [37] indicate that while conventional software can be decomposed into smaller functions that have separate requirements, design, and implementation, the functions implemented by ML models are usually large and fuzzy, sometimes accompanied by large datasets. They suggest dividing these large functions into smaller ones by specifying relevant domain-specific conditions/contexts through training/test dataset partitions and then evaluating the models on each of them [37]. Kuwajima et al. [19] suggest that model requirements are specified through the expected operational data distribution, which can then be agreed upon and enable the collection of test data that reflects the real operational conditions. The authors further suggest that in such case, the training data can be designed to allow the achievement of that requirements specification [19]. In a similar context, Mitchell et al. [9] discuss why measuring the overall performance on the entire dataset may be insufficient and why its disaggregation across different data subsets is needed. They suggest identifying factors related to variable performance, like categories of data instances with similar characteristics, instrumentation, environmental conditions etc., and measuring the performance across these factors when possible [9]. Defining the expected performance across the relevant factors and their combinations is essential, as some data subsets may be more critical than others in the context in which the system is used [66], so measuring performance changes over individual factors or their combination becomes possible [9]. An example of environment requirements specification through a data distribution matrix, as well as an example of performance requirements for each environment through a confusion matrix is given in [19].

2) CASE STUDY

ADD consists of several ML models supported by conventional software components. Therefore, the behavior of the conventional components was fully specified in the functional requirements. On the contrary, only the desired behavior, performance expectations, assumptions, constraints, dependencies and similar, were specified for the ML-based components. An excerpt from the functional requirements for the text classification component is given in Table 3. The test dataset was sampled from the operational data according to the expected data distribution across the two classes, as detailed in [11]. Due to the highly imbalanced data distribution, the f-measure was selected over the accuracy, with the expected level of performance defined by class [11].

D. QUALITY REQUIREMENTS

A quality attribute can be defined as a measurable and testable property of a system that shows how well the needs of the stakeholders are met, i.e., the quality requirements are qualifications of certain functional requirements, or qualifications of the whole system [67]. Examples of quality attributes are reliability, efficiency, robustness, usability, scalability, and many others. Different quality attributes can be of different importance to different categories of systems. For example, the specifics of the ML systems require paying particular attention to quality attributes related to ethics and trust. At the same time, these systems face new types of challenges that do not occur in conventional software systems (e.g., in security and privacy), so adaptation of some of the conventional quality attribute definitions, or even defining new attributes may be necessary [36], [37].

In complex systems, quality attributes can hardly be achieved in isolation, without affecting other attributes, so designing a system that meets its predefined quality requirements is partly about making the right trade-offs [67]. The same is true for ML systems, in which while optimizing an explicitly specified objective, the learning algorithm may neglect some other which it was not explicitly instructed to optimize. Therefore, the quality attributes relevant to an

TABLE 3. An excerpt from the functional requirements, assumptions, constraints, and dependencies of the text classification component. The implementation of the requirements is described in [11].

Requirement Type		Excerpt from ADD Text Classification Component Requirements			
Functional Requirements	**Expected Behaviour**	1. The text classification component will accept a short excerpt from a Wikipedia article and predict whether it defines an academic discipline or not. 2. The predictions will be supplemented with the text classification component confidence in the predictions. 3. The text classification component will be able to distinguish articles devoted to concepts or people closely related to the academic disciplines from articles devoted to the academic disciplines themselves. 4. The text classification component will be able to handle the imbalanced data distribution by class.			
	Expected Performance	**Performance Metrics:**	Precision, recall, and f-measure by class.		
		Justification:	The imbalanced data distribution by class makes the accuracy an inapplicable metric.		
		Expected performance by class:	**Class**	**Academic discipline**	**Other**
			Data distribution	1%	99%
			F-measure	95%	99%
Other Requirements	**Assumptions**	1. The approximate ratio of academic discipline articles and all other articles in the data retained after the heuristics-based filtering is 1% / 99%. 2. In this particular task, a limited number of terms carry most of the distinctive power in the bag of n-grams methods, so sparse representations, common for bag of n-grams methods, can be prevented through feature selection methods.			
	Constraints	1. To fulfill the high-level objectives, the experiments for selection of the most appropriate text representation method should include at least the following: - The bag of n-grams method and term frequency – inverse document frequency weighting, with optional text pre-processing. - Third-party state-of-the-art (at the time of development) text encoders based on deep neural networks. 2. To ensure comparability of the text representation methods, required by the high-level objectives, the number of features extracted by the bag of n-grams method must be comparable to the number of features extracted by the text encoders based on deep neural networks.			
	Dependencies	1. Dependency on a selected set of third-party pre-trained text encoders, as well as software libraries required for their proper functioning. 2. Dependency on third-party software libraries for certain NLP and ML functionalities. 3. Dependency on the availability of Wikipedia's XML export files, which ADD uses on input.			

ML system should be identified in cooperation with its stakeholders, formally defined, incorporated into the data and learning algorithm, and evaluated through appropriate data and metrics, while addressing any potential trade-offs. This section briefly reviews quality attributes with specific meaning and relevance to the ML domain. Because of the vague boundaries between certain quality attributes and the rapid progress of ML, the list should not be considered an exhaustive or a complete one. For a more thorough overview of each quality attribute, the reader is referred to the referenced articles. An excerpt from the quality requirements for the ADD system as a whole is given in Table 4. While some requirements refer to the conventional aspects of software development, such as the requirements for system scalability or usability, some particularly address ML specifics, such as the requirements for the ML models interpretability or their robustness to noisy input data.

1) INTERPRETABILITY

a: LITERATURE REVIEW

In the context of ML systems, interpretability can be defined as an ability to explain or present in a comprehensible way to a person [68]. It is related to the barriers to optimization and evaluation that arise from the problem formulation incompleteness in the ML domain, like the discrepancy between the real objective and the one that is actually optimized, the inability to define and evaluate all edge cases, the difficulties in defining ethics or trust requirements, and similar [68]. There are many other terms that are often associated or equated with the term ''interpretability,'' such as ''explainability,'' ''transparency,'' or ''understandability,'' among others. Therefore, this section attempts to summarize their similarities and differences reported in the literature.

While some authors make a distinction between the terms ''interpretability'' and ''explainability,'' others use them interchangeably [69]. However, several research articles have found that the ML community uses the term ''interpretable'' more often than the term ''explainable'' [69], [70]. Lipton [71] points out that interpretability is associated with different notations, i.e., transparency (understanding how the model works at the level of the entire model, its components, or training algorithm), and post-hoc interpretability (giving an explanation of the model decision, which does not necessarily explain how the model came to that decision). Transparent models are understandable to a certain degree by themselves, i.e., simulatable if a person can reason about them as a whole, decomposable if all their parts are understandable to a person without additional tools, and algorithmically transparent if a person can follow the process of producing an output from an input [60], [71]. ML models that lack transparency need a different level of post-hoc explanations, which may even apply to transparent models, based on the audience and their level of complexity [60]. For example, while linear/logistic regression, decision trees, rule-based models, or k-nearest neighbors are considered transparent models, whereas support vector machines or various types of deep neural networks are considered as models that lack transparency [60], high-dimensional linear models, rule-based models with a large number of rules, or deep

decision trees tend to become less interpretable [71]. Nevertheless, quantification measures of model interpretability have yet to be formalized by the community [60], [70].

Carvalho et al. [69] state that interpretability is essentially a subjective concept, so accordingly, when it is defined and addressed, the domain of the problem, the use case, and the needs of the audience asking questions about the model decisions should be considered. For interpretability to be implemented in the right way, it is important to analyze what makes an explanation understandable, reasonable, and human-friendly to its recipients in the specific context [69]. According to Miller [72], while most of the work in the domain relies on the researchers' intuition of what constitutes a good explanation, it may be useful to look at the findings from psychology, philosophy, or cognitive science of the way people give explanations to each other.

The trade-off between interpretability and performance is frequently discussed, because complex models that usually have better performance tend to be less interpretable. However, such a trade-off may not exist in some cases when the data is well structured and the features are of high quality, but even when it exists, the development of sophisticated explainability methods can help overcome it [60]. Herm et al. [73] have shown empirically that this trade-off is less gradual than assumed, when analyzed from the end-user perspective. They have further shown that rather than being a curve, the trade-off exhibits a grouped structure and is context dependent (e.g., on the data complexity) [73].

b: CASE STUDY

Table 4 contains an excerpt from the interpretability requirements for the ADD system. It includes requirements for consideration of inherently interpretable ML models in the experimentation phase, preference for such models when they perform similarly to the less interpretable ones, visualization of the models input/output, and outputting supplementary data which allows further result analysis.

TABLE 4. An excerpt from the quality requirements for the ADD system as a whole. The implementation of the requirements is described in [11].

Interpretability
1. Inclusion of ML models considered as inherently more interpretable in the experimentation phase, e.g., the bag of n-grams model for text representation, or logistic regression and decision trees for classification.
2. Preference for ML models that are more interpretable over those that are less interpretable when their performance is comparable.
3. Visualization of ML models input features and results with applicable methods, e.g., visualization of input features in a low dimensional vector space using dimensionality reduction methods or visualization of decision trees together with the features they decide upon.
4. Outputting a sufficient amount of supplementary data by the classification modules that allows users to interpret and verify the results.
Robustness
1. Validation of the input file format by each module and displaying / logging error messages as needed.
2. Processing records of large input files in batches and writing results in output files only after successfully processing the whole batch.
3. Robustness to exceptions when processing large input files in batches through (1) logging of information that allows users to correct and reprocess the batch at a later time, and (2) proceeding with processing of the next batch.
4. Specifying default values of the modules input parameters whenever possible, in case the user fails to specify some of them.
5. Robustness of ML models to different types of ambiguous input examples which do not represent academic disciplines themselves, but are sufficiently related to them to cause erroneous predictions, e.g., Wikipedia articles devoted to (1) scientific terminology specific to an academic discipline, (2) notable people who have contributed to an academic discipline.
6. Robustness of ML models to non-standard input examples, i.e., Wikipedia articles, which refer to academic disciplines, but may not be recognized as such due to non-compliance with Wikipedia's policies on article titles, lead sections, and similar.
Scalability
1. Ability to handle the constantly growing size of Wikipedia's XML export files.
2. Standardized format of the classification modules input and output files, allowing their seamless modification or replacement in a subsequent release.
Usability
1. As few mandatory user-specified input parameters of the modules as possible.
Reusability
1. Reuse of third-party software libraries which reliably implement non-trivial NLP and ML functionalities, in order to minimize the risk of errors and the testing time.
2. Use of programming language that facilitates the reuse of third-party software libraries which reliably implement non-trivial NLP and ML functionalities.
3. Use of programming language that facilitates the reuse of third-party text encoders based on deep neural networks.

2) FAIRNESS

a: LITERATURE REVIEW

With the increased use of ML algorithms in making decisions about individuals, ensuring an outcome that is fair and non-discriminatory in relation to sensitive characteristics (e.g., gender, race) requires serious attention from the ML practitioners. Fairness can be defined as an absence of prejudice or favoritism towards individuals or groups based on certain inherent or acquired characteristics they possess [74]. The ML community has proposed a number of different formal definitions of fairness. Some target individual fairness, i.e., similarly treating similar individuals, while others target group fairness, i.e., treating different groups equally. However, fairness definitions have their limitations too, as discussed in [57] and [75]. The most basic definition, known as Fairness Through Unawareness, requires protected attributes not to be explicitly used in decision-making processes but, still, it has shortcomings as other features may contain discriminatory information analogous to that in the protected attributes [75]. Demographic Parity, also known as Statistical Parity [76], requires membership in a protected class not to be correlated with the decision. Equalized Odds [77] requires protected and unprotected classes to have equal true-positive and false-positive rates, while Equal Opportunity [77] is a weaker notation than Equalized Odds and requires non-discrimination only over the ''advantaged'' outcome. While the previous three definitions fall in the category of group fairness, the following two belong to the

category of individual fairness. The Individual Fairness [76] definition requires similar individuals to get similar predictions by an algorithm under some carefully chosen similarity metric. The Counterfactual Fairness [75] definition indicates that protected attributes should not be a cause of the predictor in any individual instance.

Bias in ML can be addressed in three stages, i.e., (1) by removing it from the data through pre-processing, (2) by modifying the learning algorithm, and (3) by reassigning the model predictions, when the model is treated as a black-box [74]. One source of bias in ML is the data itself, in which bias can take a variety of forms, like historical bias already existing in the real world and therefore existing in the data, representation bias when the data lacks diversity by a particular criterion, bias in measuring a particular feature, aggregation bias and many other types [74]. One example of bias found in ML results, as well as a discussion of the risk of its inheritance and even amplification by other dependent models, is presented in [78], where a set of widely used word vectors, the distances of which represent relationships between words, have been found to contain a gender bias. Since bias can be inadvertently introduced into ML systems in a number of ways and at various stages of its development, its identification and addressing should begin as early as possible. Properly defined fairness requirements are the right place to start.

3) ROBUSTNESS

a: LITERATURE REVIEW

In line with the general robustness definition, ML algorithms should be capable of learning robust models even in the presence of noisy training data and remain robust at operation time. This makes robustness a rather broad attribute, closely related to many of the other described in this section.

ML systems' ability to stay robust at operation time, when faced with input different from that seen during training, is essential because non-robust ML systems may not only show poor performance but may wrongly assume a good performance and confidently take wrong action [79]. The robustness of ML models has been studied extensively in the context of adversarial examples, inputs designed to force a model to produce erroneous outputs, most commonly through small perturbations which make the new input close to the original one according to a domain-specific distance metric, but misclassified by the model [80]. Evaluating model robustness is important for several reasons, i.e., (1) to prevent models from misbehaving due to adversaries, (2) to use their good worst-case robustness as an evidence that they will not misbehave in the real-world due to unforeseen randomness, and (3) to compare models with human abilities [80]. To address ML models' robustness properly, their performance expectations should be defined rigorously, deviations from such expectations should be prevented, and methods to identify/correct such deviations should be defined, all of which leads to accountability in the ML field [81].

A commonly discussed trade-off is that of models' robustness and accuracy. Recent works have shown that larger and more complex datasets are needed for robust learning versus those needed for standard learning [82], as well as a trade-off between the accuracy of models trained for adversarial robustness and their standard accuracy achieved when trained on unperturbed inputs [83]. They have further shown that the learned feature representations differ in the two settings, but models that encode prior about human perception seem invariant to perturbations to which the humans are invariant [83]. Recent works study methods to mitigate robustness and accuracy trade-offs [84].

b: CASE STUDY

Table 4 contains an excerpt from the robustness requirements for the ADD system. It includes requirements for proper input file format validation, default input values, and robustness to exceptions during processing of large input files. It also includes ML-specific robustness requirements, i.e., such requiring ML models robustness to ambiguous or non-standard input examples.

4) SECURITY

a: LITERATURE REVIEW

The growing use of ML in many different domains, including safety-critical ones, requires an understanding of the new security vulnerabilities that are not present in other types of systems and strengthening the robustness against them. Nevertheless, Carlini et al. [80] indicate that while the studies of adversarial examples in new domains are advancing rapidly, the design of systems robust to such examples is slower.

To analyze the security of a system, it is necessary to identify (1) security goals, i.e., requirements that, if violated, result in a compromise of an asset and (2) a threat model [85]. This model defines the conditions under which a defense is designed to be secure and the security guarantees it provides [80]. Some of the different models proposed in the literature consider the adversary's (1) goal/incentives, i.e., accessing system assets or denying normal operation, and (2) capability, i.e., its knowledge of the system and constraints to its capability [85]. Others consider the adversary's (1) goal, (2) knowledge, i.e., complete knowledge of the model or varying degree of black-box access, and (3) capability [80]. In the context of supervised learning, the violations can be classified across three dimensions, i.e., (1) influence (causative or exploratory), (2) security violation (integrity or availability), and (3) specificity of the adversary's intention (targeted or indiscriminate) [85]. Different examples of learning in adversarial environments have been described in the literature, e.g., in [86] and [87].

Barreno et al. [85] have found that improving the worst-case robustness of an algorithm can make it less effective on average. Based on their analysis of the most common shortcomings of adversarial example defenses, Carlini et al. [80] have defined a set of guidelines for defense

evaluation, emphasizing the extreme caution and skepticism that this process requires.

5) PRIVACY

a: LITERATURE REVIEW

There are certain situations in which the exposure of a model, its parameters, or training data should be prevented due to confidentiality or privacy. Still, ML models' capacity to memorize elements of the training data makes it challenging to provide guarantees that participation in a training set does not affect the privacy of the individuals [86]. The adversaries usually aim at recovering the training data or the model, like recovering partially known inputs with the most probable values or extracting the training data using the outputs [86]. Several methodologies for addressing privacy concerns in the ML domain are differential privacy, federated learning, and data encryption.

Differential privacy represents a mathematically rigorous definition of privacy, which ensures that the output of a database analysis is distributed very similarly to the output of the analysis of another that differs from the first in one row only, while bounding the maximum divergence between the two distributions by a privacy loss parameter [88]. Federated learning refers to a setting where many clients collaboratively train a model while being orchestrated by a central server/service provider and while keeping their training data decentralized [89]. The learning objective is achieved through updates that contain the minimum necessary information for the learning task and which are suitable for an immediate aggregation [89]. Another way to preserve data privacy is to train a model or make inferences on encrypted data using methods like homomorphic encryption or secure multi-party computation. Several examples of their use in the ML domain include customizing ML algorithms to use homomorphic encryption in training and inference stages [90], making predictions with neural networks on encrypted data using homomorphic encryption [91], and others.

In the context of trade-offs that come from the use of privacy-preserving methods, Brundage et al. [92] point to trade-offs between the privacy benefits, the model quality, the developers' experience, and the costs in computation, communication, or energy consumption. Papernot et al. [86] point to a fundamental tension between the security/privacy and the precision in ML systems with a finite capacity. In terms of neural networks and homomorphic encryption, Gilad-Bachrach et al. [91] indicate that adding encryption makes the training process slower, at the same time preventing the data scientists from inspecting the data or tuning the model during training.

6) SAFETY

a: LITERATURE REVIEW

In the context of ML, Varshney [93] defines safety as minimization of the risk and uncertainty associated with harmful events, i.e., events related to sufficiently high cost in some human sense. The author identifies several sources of risk in ML systems, i.e., (1) assumption that the training data comes from the operational data distribution, (2) low probability density of the operational data distribution in certain regions, (3) uncertainty coming from the way the test set was instantiated, and (4) dependence of the loss function on the predicted and actual values only [93]. Several approaches to mitigate these risks include ensuring an inherently safe design, adding safety factors or margins, adding additional procedural safeguards beyond those designed in the core functionality, and ensuring a safe fail [93]. In the context of supervised and reinforcement learning, Amodei et al. [79] have identified several sources of safety risks, i.e., (1) a loss function that inadvertently ignores aspects of complex environments that could be harmful if changed at operation time (2) a loss function minimized by an easy solution during training which was not the designer's true intention, (3) substituting the correct loss function with another one because the former is too expensive for frequent evaluation, (4) failure to ensure safe actions when the system encounters unseen input. Furthermore, Jacovi et al. [58] indicate that adequate verification of the existence of a certain risk (an undesirable but possible event) from the use of an AI system is a prerequisite for verification of the existence of Human-AI trust.

E. OTHER REQUIREMENTS

1) ASSUMPTIONS

a: LITERATURE REVIEW

The assumptions are an almost inevitable aspect of ML system development. For example, assumptions are made when certain aspects of the problem to be solved or its data are not observable. Based on those assumptions, the real problem is translated to an ML problem, and an appropriate class of models is selected to solve it. Other examples of assumptions are those related to the data distribution across different classes in the real dataset, assumptions that the statistical properties are similar across the entire dataset [94], and similar. Assumptions are made about quality attributes as well. Deviations from the assumptions on which a particular class of models is based can be a source of problems, as summarized for DL models in [47]. Furthermore, in DL, the assumptions made about the neural network architecture, training data, loss function, and optimization algorithm not only constrain the problem solutions that can be learned, but determine how easily a particular solution can be learned, therefore, may inadvertently create opportunities to learn an undesirable "shortcut" solution to a problem that does not work well in real-world settings [49]. Unclearly defined or omitted assumptions affect accountability in AI systems, as they leave room for avoiding responsibility for any errors resulting from wrong assumption, by blaming unavoidable and inexplicable software "bugs" [81]. Therefore, identifying and documenting the assumptions prevents stakeholders from neglecting or misinterpreting them in the development process and allows for appropriate addressing of their effects.

TABLE 5. Brief summary of the relevance of conventional RE activities to the ML development process, the challenges this process brings them, and the necessary adjustments to these activities to better fit into the process.

RE Activity	RE Activity Relevance to ML Development Process	RE Activity Challenges Imposed by ML Development Process / Adjustments to Fit Into the Process
Requirements Elicitation and Analysis	- RE activities through which the stakeholders' needs and constraints are collected and analyzed [45]. - A successful practical application of ML starts by defining the goals to be achieved, based on the problem to be solved by the ML system. [4].	- In ML systems, the feasibility of the stakeholders' requirements depends on the available data as well. - New stakeholders are involved in the process (e.g., data scientists, legal experts) [10]. - Stakeholders should be helped to make their expectations from ML more realistic [13]. - Stakeholders should accept the uncertainty in the time and cost estimates [13]. - Each class of ML models has its own assumptions/limitations, and some promising research results may not be directly applicable in real conditions [13], so the decision to implement an ML-based solution to a problem should be based primarily on problem-specific analyses. - When relevant requirements are not well-specified and enforced on the ML pipeline, i.e., are underspecified, many near-optimal solutions that fit that incomplete specification, but behave differently in different dimensions (e.g., interpretability, fairness) may exist and be selected over the desired one [48]. - Preventing the learning of undesirable "shortcut" solutions requires a thorough understanding of what makes a particular solution easy to be learned in a given context, understanding the influence of various factors throughout the ML pipeline and their interactions [49].
Requirements Specification	- RE activities that produce a formal specification of requirements.	- No best practices for specifying requirements for ML systems have been found in the literature. - Best practices from related fields can be reused in formal specification of certain types of requirements, e.g., non-functional, data requirements [27]. - The available approaches to specifying certain types of requirements require additional refinement, e.g., performance, interpretability [27]. - A related research article has found that the most commonly used modeling notations/languages in specifying requirements for AI systems are (1) UML, (2) GORE and (3) Domain Specific Models, but they have limitations when used for that purpose [31].
Requirements Validation	- RE activities that ensure that the right requirements are captured and met [45].	- The literature indicates that ML specifics limit the effectiveness of conventional testing approaches [53]. - Novel testing techniques that address ML specifics are needed [53]. - Models should be explicitly tested for any required behavior that is not guaranteed by the independent and identically distributed test dataset, as some required behavior will almost certainly be underspecified [48]. - The testing of the model behavior should be application specific and developed based on the requirements [48]. - Testing on data that is out-of-distribution can help in distinguishing desired solutions from "shortcut" solutions [49].

b: CASE STUDY

The assumptions for ADD were defined at two levels of abstraction, i.e., high-level assumptions and assumptions related to specific functionalities. An excerpt from the first type is given in Table 2. It includes our assumptions related to the definitions and subordination of the domain-specific thematic structure detected by ADD and its related ones. These statements are considered assumptions due to the lack of precise definitions and widely accepted understanding of their subordination in the domain. By clearly documenting them, we ensured that all stakeholders of ADD share the same understanding. The second type of assumptions includes those related to ML models, e.g., the expected data distribution, relevant features, or appropriate class of models. An excerpt from the assumptions related to the text classification component is given in Table 3.

2) DEPENDENCIES

a: LITERATURE REVIEW

Dependencies are external factors or components on which a project or system depends but are beyond its control, so they can turn into risks if left undefined or inappropriately monitored [45]. As already mentioned in Section V-A4, ML systems are inevitably dependent on data, and they often depend on external models or external software libraries, so the team implementing the system may not always have full control over them. For those reasons, proper documentation of the dependencies and monitoring their effects on the ML system is crucial. Breck et al. [59] have summarized a set of best practices for preventing potential risks arising from dependencies, which can sometimes lead to model misbehavior even without strange enough outputs to trigger monitoring mechanisms.

b: CASE STUDY

Table 3 contains an excerpt from the ADD dependencies defined for the text classification component. They primarily refer to the component dependence on third-party pre-trained text encoders, software libraries for ML and NLP functionalities, and Wikipedia XML export files which ADD uses on input.

3) CONSTRAINTS

a: LITERATURE REVIEW

Constraints are restrictions on the design and implementation choices that the developers can make about a solution, which can result from decisions made by management,

TABLE 6. Brief summary of the importance of different types of requirements (high-level, user, and functional) in addressing the ML specifics, the challenges that ML specifics bring to their conventional understanding, and the necessary adjustments of this understanding.

Requirement Type		Requirements Importance in Addressing ML Specifics	Challenges Imposed by ML Specifics / Adjustments of Requirements Conventional Understanding
High-Level (Business)	Objectives	- Define what needs to be achieved through the use of the ML system.	- The objectives should comply with the ethical principles for ML systems. - The literature points to some divergence in the definitions of ML systems' ethical principles and uncertainty about their implementation in practice [55].
	Success / Performance Metrics	- Allow quantitative evaluation if ML systems meet requirements.	- It may be a challenge to find ML performance metrics that match the desired behavior [4]. - Knowledge of ML performance metrics is required during RE [10].
	Limitations	- Clearly defined limitations prevent over-optimistic expectations related to ML systems' performance in various settings. - Prevent misinterpretations of ML systems' outputs or their improper use.	- ML models have limitations when the conditions and assumptions from the training phase no longer apply [4], [47]. - The definitions of certain quality attributes in the ML domain have limitations [57]. - Vaguely specified behavior which users should trust to be upheld by an AI system, leads to unwarranted trust and system misuse, since users may implicitly assume system behavior (and trust it would be upheld) which has not been considered to be upheld by its developers [58].
	Risks	- Early identification of potential risks allows their assessment and control [45]. - Failure to do so can adversely affect the project success [45].	- The complexity of ML systems and their environment makes it challenging to anticipate the potential risks from their behavior or interactions [8]. - ML models are highly sensitive to changes in their data, hyperparameters, environment, or other ML models on which they depend [6]. - There may be important slices of data that have different performance than the overall one in certain conditions [59]. - Proper data management/versioning is more challenging than doing the same with software code [7]. - ML systems face new risks related to security, privacy, safety, lack of interpretability, biased decisions and similar. - Underspecifying relevant behavior to be learned by the ML pipeline, leaves space for the ML pipeline to choose an undesired solution over the desired one, if they both have the same test performance, but different compliance with the "unspecified", desired requirements [48], [49].
User		- Reveal user expectations from the system (e.g., the most appropriate type of decision explanations). - Ensure user acceptance and trust in the system [16].	- In business analytics projects, elicitation of the user expectations from the system is insufficient to reveal the complex knowledge needed to develop it [45]. The same is true for ML systems as well. - The decision explanations should be tailored to the use case and the audience receiving them [60]–[63]. - It is difficult to predict failures and plan appropriate interactions if the ML system's behavior evolves over time [65].
Functional		- Define the expected behavior of the ML model and its level of performance. - Guide the data collection process, the selection/configuration of the learning algorithm, and the evaluation.	- ML systems usually consist of conventionally programmed functionalities and functionalities implemented by ML models [6]. - The functions implemented by ML models are usually complex and accompanied by large datasets, as opposed to the decomposable functionalities in conventional software [37]. - The rules connecting the input to the output of the ML models are not explicitly stated in the requirements, but learned from data. - It can be challenging to formally specify the expected behavior/performance of an ML model in all possible conditions [19]. - The expected performance may vary across different data subsets, and some may be more critical than others in certain conditions [9], [66].

requirements from external stakeholders, requirements for compliance with standards or agreements, and a variety of other reasons [45]. In the context of ML systems, examples include policy constraints that may enforce certain requirements, e.g., on privacy [23]. Other examples include data constraints which describe meaningful feature ranges, feature dependencies, or invariants, ensuring the data validity after its transformation [27]. Constraints may encode certain prior knowledge, a preference towards a simpler class of models [4], or in other ways guide the ML pipeline in learning models that satisfy a broader set of behavioral requirements that are sometimes not covered by the standard ML testing process (e.g., requirements related to interpretability or fairness) [48]. For ML systems that continuously learn and change their behavior, hard-coding rules for system behavior that prevent it from learning behavior that does not conform to the relevant standards agreed upon among stakeholders has been suggested also [95].

b: CASE STUDY

Table 3 contains an excerpt from the constraints of the ADD project. They refer to certain imposed experimental choices to the developers of the text classification component in order to meet the high-level objectives of ADD related to the comparison of state-of-the-art and conventional text representation methods.

VI. DISCUSSION

This section offers a summary of all previously stated findings in the article, related to the importance of the RE activities in the development of ML systems, the importance of certain types of requirements, the challenges associated

TABLE 7. Brief summary of the importance of different types of quality requirements in addressing the ML specifics, the challenges that ML specifics bring to their conventional understanding, and the necessary adjustments of this understanding.

Requirement Type		Requirements Importance in Addressing ML Specifics	Challenges Imposed by ML Specifics / Adjustments of Requirements Conventional Understanding
Quality	General	- Requirements related to measurable and testable properties of the system, i.e., quality attributes [67].	- The quality requirements should address new concerns, atypical to conventional software systems [36], [37]. - In complex systems, meeting quality requirements is partly about making the right trade-offs between quality attributes [67].
	Interpretability	- Define the expectations from the explanations of the ML systems decisions. - Crucial to give stakeholders a better understanding of ML systems' decision-making process.	- Many terms are used interchangeably, e.g., "interpretability," "explainability," "transparency," "understandability". - The concept of interpretability encompasses both (1) understanding how the model works, and (2) explaining decisions without necessarily explaining the background decision-making process [71]. - Even inherently interpretable models (e.g., linear models, decision trees) may lose this property as their complexity increases [71]. - Interpretability quantification measures have yet to be formalized by the community [60], [70]. - The implementation of interpretability requires an understanding of the problem domain, the characteristics of the explanation recipients, and the use case [69]. - Complex ML models tend to be less interpretable [60], so a trade-off between performance and interpretability is often reported in the literature. - From end-user perspective, this trade-off seems less gradual than assumed and is context dependent (e.g., dependent on data complexity) [73].
	Fairness	- Ensure the absence of prejudice or favoritism towards individuals or groups based on the inherent or acquired characteristics they possess [74].	- In the ML domain, several formal definitions of fairness are available, each with its own limitations [57], [75]. - In ML, bias can be inadvertently introduced at different stages of the learning process and may even exist in the data itself [74]. - If it goes unnoticed, dependent ML models may inherit/amplify bias [78].
	Robustness	- Ensure proper functioning of ML systems from different aspects (e.g., robustness to unforeseen input at operation time).	- Non-robust ML systems may show poor performance when faced with new input at operation time, or may even wrongly assume good performance and take wrong action [79]. - Adversarial examples are a common challenge for ML systems [80]. - Properly defined and addressed robustness requirements implicate accountability in the ML field [81]. - The trade-off between ML models' robustness and their accuracy in the standard sense is often discussed in the literature [82]–[84].
	Security	- Ensure that vulnerabilities of ML systems are identified in advance and addressed accordingly.	- ML systems face new types of security vulnerabilities that are not present in other types of systems. - The studies of new types of adversarial examples in new domains are advancing rapidly [80]. - Designing and implementing a secure ML system requires defining its security goals and threat model, i.e., analyzing adversaries' knowledge and capabilities from many different aspects [80], [85]. - The literature emphasizes the need for great caution and skepticism when designing/evaluating adversarial examples defenses [80].
	Privacy	- Ensure that parameters of ML models or their training data are not exposed when they need to remain confidential or private.	- ML models can memorize parts of their training data, making the preservation of training data privacy challenging [86]. - Privacy-preserving methods may require making trade-offs with the model performance or training time, and may even make training data inaccessible to data scientists [86], [91], [92].
	Safety	- Minimize the risks and uncertainties associated with harmful events [93].	- Many sources of risks have been identified in the literature, and some may be difficult to address [93]. - Examples of safety risk sources include making incorrect assumptions about the training and real data distribution, defining an incorrect loss function, ignoring relevant aspects of complex environments in the loss function, failing to secure safe actions and safe fail [79], [93]. - A prerequisite to verify the existence of Human-AI trust, is an adequate verification of the existence of certain risk from the use of an AI system [58].

with RE activities, and those associated with the conventional understanding of the requirements.

ML systems have become ubiquitous in many segments of our lives due to the numerous benefits from their use. Nevertheless, ML systems are complex systems which learn their behavior from data. Since data can be imperfect or reflect historical human biases, ML systems are at risk of acquiring these imperfections through the learning process. Furthermore, ML systems can implement complex decision functions, which depend on many factors and which may lead to outcomes that cannot always be predicted with certainty. Therefore, the importance of identifying, analyzing, documenting, and validating the expected behavior of an ML system, the intended and unintended use cases, the risks, limitations, assumptions, the performance and quality expectations, or the required compliance with

TABLE 8. Brief summary of the importance of different types of requirements (assumptions, dependencies, and constraints) in addressing the ML specifics, the challenges that ML specifics bring to their conventional understanding, and the necessary adjustments of this understanding.

Requirement Type		Requirements Importance in Addressing ML Specifics	Challenges Imposed by ML Specifics / Adjustments of Requirements Conventional Understanding
Other	Assumptions	- Documenting the assumptions about the problem to be solved, ensures that the correct class of ML models is selected.	- The assumptions on which the selected class of ML models is based should correspond to the assumptions made about the problem to be solved. - Deviations from the assumptions on which the selected class of ML models is based can lead to problems [47]. - Assumptions related to neural networks architecture, training data, loss function, and optimization algorithm, not only constrain the solutions that can be learned but determine how easily a particular solution can be learned [49] - Wrong assumptions may inadvertently create an opportunity to learn an undesirable "shortcut" solution to a problem, not working well in a real-world setting [49]. - Unclearly defined/omitted assumptions affect accountability, leaving room to avoid responsibility for errors resulting from such assumptions and blaming unavoidable/inexplicable software "bugs" [81].
	Dependencies	- ML systems have many dependencies (e.g., external systems, ML models, data, software libraries) [6]. - Dependencies may turn into risks if handled improperly [45].	- In the development of ML systems, having a complete control over the dependencies may not be possible (e.g., external systems, data) [6], so appropriate monitoring is required. - The risks arising from dependencies can sometimes lead to misbehavior of the model even without strange enough outputs to trigger the monitoring mechanisms [59].
	Constraints	- Documenting the constraints ensures that none of them is neglected.	- ML system development is often constrained by different standards/policies (e.g., security, privacy, ethical, legal) [23]. - During data transformation, the data constraints that ensure its validity should be taken into account [27]. - ML systems may have a broader set of behavioral requirements (e.g., on interpretability, fairness), so enforcing proper constraints, guides the ML pipeline in learning models that satisfy those requirements [48]. - For ML systems that continuously learn/change their behavior, hard-coding rules that prevent them from learning a behavior that does not conform to relevant standards agreed upon among stakeholders has been suggested [95].

ethical/legal constraints should not be underestimated. On the contrary, these RE activities should be given attention as early as possible in the ML development process. The literature review provided in this article confirms that carefully conducted RE activities can add value to the rather complex ML development process, in the same way that they add value to the conventional software development process.

Nevertheless, the ML development process has its own specifics that affect the already established RE best practices. The results of the literature review and the case study are consistent in terms of the significant impact that the ML development process has on the conventional, well-established RE activities, but they also highlight the benefits of these activities in dealing with the complexity of the process. ML introduces new activities through which requirements are identified and refined (e.g., data analysis and experimentation), introduces non-trivial challenges to be anticipated in the RE phase, and makes some of the established RE best practices inapplicable. However, at the time of writing, RE best practices for ML systems do not exist and have yet to be defined by the community. Table 5 briefly summarizes the findings related to (1) the relevance of conventional RE activities to the ML development process, (2) the challenges that this process brings them, and (3) their necessary adjustments to better fit into this process, all of which presented in the previous sections of this article.

The literature review also confirms the importance of each of the requirement types considered in this article to ML systems. However, the conventional understanding of some of them (e.g., functional requirements or certain quality attributes) may require adjustment in the context of ML. Table 6, Table 7, and Table 8 briefly summarize the findings related to (1) the importance of the different types of requirements in addressing the ML specifics, (2) the challenges that ML specifics bring to the conventional understanding of these requirements, and (3) the necessary adjustments of this understanding, all of which presented in the previous sections of this article.

Finally, given the current prevalence of ML in software development, we believe that the number of research articles on this topic will continue to grow in the coming years, offering experiences from real ML projects, as well as new or adjusted methodologies that better fit the ML development process. However, until widely accepted RE best practices for ML systems are available, we believe that the already established RE models, applied with awareness of the ML specifics, provide a solid foundation for a thorough and shared understanding of what needs to be implemented in and what is expected from an ML system, while minimizing the risk of neglecting important requirements.

VII. CONCLUSION

Machine learning has become a common choice in modern software development across many domains. Nevertheless, while it can provide data-driven solutions to many problems that people find difficult to solve, at the same time, it challenges the well-established software development best practices. Furthermore, machine learning introduces new technical and ethical challenges of which the stakeholders must be fully aware even before the project begins. Since the requirements engineering activities provide a proper understanding of the problem and ensure implementation of an appropriate solution, these are the right activities where solving machine learning challenges should begin. As the requirements engineering activities are also influenced by the machine learning specifics, but best practices do not exist yet, this article aims to analyze the impact that machine learning has on conventional requirements engineering activities and types of requirements, to emphasize the importance of proper requirements engineering in machine learning projects, and to share our experience through a case study. Most importantly, the purpose of this article is to motivate further discussion and sharing of practical experiences on this important topic because, in the future, machine learning systems will become even more present in our daily lives.

The presented literature review and case study findings confirm that the machine learning development process affects the conventional, well-established requirements engineering activities, but they also confirm the relevance of these activities to the process. Furthermore, the findings confirm the relevance of the different requirement types considered in this article to machine learning systems, as well as the necessary adjustment of the conventional understanding of some of them in the context of machine learning (e.g., functional requirements or certain quality attributes). Therefore, we believe that the future research should continue focusing on adjusting (1) the requirements engineering activities and (2) the understanding of the different requirement types so they fit even better into the machine learning development process, as well as on presenting requirements engineering experiences from real machine learning projects.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[5] A. Karpathy. (2017). *Software 2.0*. Accessed: May 15, 2023. [Online]. Available: https://karpathy.medium.com/software-2-0-a64152b37c35

[6] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 2503–2511.

[7] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Softw. Eng. Pract. (ICSE-SEIP)*, May 2019, pp. 291–300.

[8] I. Rahwan et al., "Machine behaviour," *Nature*, vol. 568, no. 7753, pp. 477–486, 2019.

[9] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru, "Model cards for model reporting," in *Proc. Conf. Fairness, Accountability, Transparency*. New York, NY, USA: Association for Computing Machinery, Jan. 2019, pp. 220–229.

[10] A. Vogelsang and M. Borg, "Requirements engineering for machine learning: Perspectives from data scientists," in *Proc. IEEE 27th Int. Requirements Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 245–251.

[11] A. Gjorgjevikj, K. Mishev, and D. Trajanov, "ADD: Academic disciplines detector based on Wikipedia," *IEEE Access*, vol. 8, pp. 7005–7019, 2020.

[12] C. Kästner and E. Kang, "Teaching software engineering for AI-enabled systems," 2020, *arXiv:2001.06691*.

[13] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. Maria Vollmer, and S. Wagner, "Software engineering for AI-based systems: A survey," 2021, *arXiv:2105.01984*.

[14] H. Belani, M. Vukovic, and Ž. Car, "Requirements engineering challenges in building AI-based complex systems," in *Proc. IEEE 27th Int. Requirement Eng. Conf. Workshops (REW)*, Sep. 2019, pp. 252–255.

[15] T. Chuprina, D. Mendez, and K. Wnuk, "Towards artefact-based requirements engineering for data-centric systems," 2021, *arXiv:2103.05233*.

[16] H.-M. Heyn, E. Knauss, A. P. Muhammad, O. Eriksson, J. Linder, P. Subbiah, S. K. Pradhan, and S. Tungal, "Requirement engineering challenges for AI-intense systems development," 2021, *arXiv:2103.10270*.

[17] S. Studer, T. Binh Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters, and K.-R. Mueller, "Towards CRISP-ML(Q): A machine learning process model with quality assurance methodology," 2020, *arXiv:2003.05155*.

[18] X. Zhang, Y. Yang, Y. Feng, and Z. Chen, "Software engineering practice in the development of deep learning applications," 2019, *arXiv:1910.03156*.

[19] H. Kuwajima, H. Yasuoka, and T. Nakae, "Engineering problems in machine learning systems," *Mach. Learn.*, vol. 109, no. 5, pp. 1103–1126, May 2020.

[20] M. Saidur Rahman, E. Rivera, F. Khomh, Y.-G. Guéhéneuc, and B. Lehnert, "Machine learning software engineering in practice: An industrial case study," 2019, *arXiv:1906.07154*.

[21] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?" *IEEE Trans. Softw. Eng.*, vol. 47, no. 9, pp. 1857–1871, Sep. 2021.

[22] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *J. Syst. Softw.*, vol. 180, Oct. 2021, Art. no. 111031.

[23] A. Serban and J. Visser, "Adapting software architectures to machine learning challenges," 2021, *arXiv:2105.12422*.

[24] A. Pereira and C. Thomas, "Challenges of machine learning applied to safety-critical cyber-physical systems," *Mach. Learn. Knowl. Extraction*, vol. 2, no. 4, pp. 579–602, Nov. 2020.

[25] G. Lorenzoni, P. Alencar, N. Nascimento, and D. Cowan, "Machine learning model development from a software engineering perspective: A systematic literature review," 2021, *arXiv:2102.07574*.

[26] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, "A taxonomy of software engineering challenges for machine learning systems: An empirical investigation," in *Proc. Int. Conf. Agile Softw. Develop.* Cham, Switzerland: Springer, 2019, pp. 227–243.

[27] W. Maass and V. C. Storey, "Pairing conceptual modeling with machine learning," *Data Knowl. Eng.*, vol. 134, Jul. 2021, Art. no. 101909.

[28] H. Villamizar, M. Kalinowski, and H. Lopes, "A catalogue of concerns for specifying machine learning-enabled systems," 2022, *arXiv:2204.07662*.

[29] H. Villamizar, M. Kalinowski, and H. Lopes, "Towards perspective-based specification of machine learning-enabled systems," 2022, *arXiv:2206.09760*.

[30] Z. Pei, L. Liu, C. Wang, and J. Wang, "Requirements engineering for machine learning: A review and reflection," in *Proc. IEEE 30th Int. Requirements Eng. Conf. Workshops (REW)*, Aug. 2022, pp. 166–175.

[31] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, "Requirements engineering for artificial intelligence systems: A systematic mapping study," *Inf. Softw. Technol.*, vol. 158, Jun. 2023, Art. no. 107176.

[32] K. Ahmad, M. Abdelrazek, C. Arora, M. Bano, and J. Grundy, "Requirements practices and gaps when engineering human-centered artificial intelligence systems," 2023, *arXiv:2301.10404*.

[33] B. Jahić, N. Guelfi, and B. Ries, "SEMKIS-DSL: A domain-specific language to support requirements engineering of datasets and neural network recognition," *Information*, vol. 14, no. 4, p. 213, 2023.

[34] A. A. H. de Hond, A. M. Leeuwenberg, L. Hooft, I. M. J. Kant, S. W. J. Nijman, H. J. A. van Os, J. J. Aardoom, T. P. A. Debray, E. Schuit, M. van Smeden, J. B. Reitsma, E. W. Steyerberg, N. H. Chavannes, and K. G. M. Moons, "Guidelines and quality criteria for artificial intelligence-based prediction models in healthcare: A scoping review," *NPJ Digit. Med.*, vol. 5, no. 1, pp. 1–12, Jan. 2022.

[35] L. Pons and I. Ozkaya, "Priority quality attributes for engineering AI-enabled systems," 2019, *arXiv:1911.02912*.

[36] J. Horkoff, "Non-functional requirements for machine learning: Challenges and new directions," in *Proc. IEEE 27th Int. Requirements Eng. Conf. (RE)*, Sep. 2019, pp. 386–391.

[37] H. Kuwajima and F. Ishikawa, "Adapting SQuaRE for quality assessment of artificial intelligence systems," 2019, *arXiv:1908.02134*.

[38] *Ethics Guidelines for Trustworthy AI*, AI HLEG, Eur. Commission, 2019.

[39] J. Siebert, L. Joeckel, J. Heidrich, A. Trendowicz, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, and M. Aoyama, "Construction of a quality model for machine learning systems," *Softw. Quality J.*, vol. 2021, pp. 1–29, Jan. 2021.

[40] K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, L. Joeckel, J. Siebert, and J. Heidrich, "Requirements-driven method to determine quality characteristics and measurements for machine learning software and its evaluation," in *Proc. IEEE 28th Int. Requirements Eng. Conf. (RE)*, Aug. 2020, pp. 260–270.

[41] K. M. Habibullah and J. Horkoff, "Non-functional requirements for machine learning: Understanding current use and challenges in industry," in *Proc. IEEE 29th Int. Requirements Eng. Conf. (RE)*, Sep. 2021, pp. 13–23.

[42] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-functional requirements for machine learning: Understanding current use and challenges among practitioners," *Requirements Eng.*, vol. 28, no. 2, pp. 283–316, Jun. 2023.

[43] K. M. Habibullah, G. Gay, and J. Horkoff, "Non-functional requirements for machine learning: An exploration of system scope and interest," 2022, *arXiv:2203.11063*.

[44] B. C. Hu, L. Marsso, K. Czarnecki, R. Salay, H. Shen, and M. Chechik, "If a human can see it, so should your system: Reliability requirements for machine vision components," in *Proc. IEEE/ACM 44th Int. Conf. Softw. Eng. (ICSE)*. New York, NY, USA: Association for Computing Machinery, May 2022, pp. 1145–1156.

[45] K. Wiegers and J. Beatty, *Software Requirements*. London, U.K.: Pearson, 2013.

[46] J. Eckhardt, A. Vogelsang, and D. M. Fernández, "Are 'non-functional' requirements really non-functional? An investigation of non-functional requirements in practice," in *Proc. 38th Int. Conf. Softw. Eng.* New York, NY, USA: Association for Computing Machinery, 2016, pp. 832–842.

[47] G. Marcus, "Deep learning: A critical appraisal," 2018, *arXiv:1801.00631*.

[48] A. D'Amour et al., "Underspecification presents challenges for credibility in modern machine learning," *J. Mach. Learn. Res.*, vol. 23, no. 1, pp. 10237–10297, 2022.

[49] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Mach. Intell.*, vol. 2, no. 11, pp. 665–673, Nov. 2020.

[50] A. Gjorgjevik, R. Stojanov, and D. Trajanov, "SemCCM: Course and competence management in learning management systems using semantic web technologies," in *Proc. 10th Int. Conf. Semantic Syst.* New York, NY, USA: Association for Computing Machinery, Sep. 2014, pp. 140–147.

[51] A. Dimitrovski, A. Gjorgjevikj, and D. Trajanov, "Courses content classification based on Wikipedia and CIP taxonomy," in *ICT Innovations 2017*, D. Trajanov and V. Bakeva, Eds. Cham, Switzerland: Springer, 2017, pp. 140–153.

[52] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Toward verified artificial intelligence," *Commun. ACM*, vol. 65, no. 7, pp. 46–55, Jul. 2022.

[53] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empirical Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, Nov. 2020.

[54] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 1–36, Jan. 2022.

[55] A. Jobin, M. Ienca, and E. Vayena, "The global landscape of AI ethics guidelines," *Nature Mach. Intell.*, vol. 1, no. 9, pp. 389–399, Sep. 2019.

[56] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996.

[57] S. Corbett-Davies and S. Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," 2018, *arXiv:1808.00023*.

[58] A. Jacovi, A. Marasović, T. Miller, and Y. Goldberg, "Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in AI," in *Proc. ACM Conf. Fairness, Accountability, Transparency*. New York, NY, USA: Association for Computing Machinery, Mar. 2021, pp. 624–635.

[59] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML test score: A rubric for ML production readiness and technical debt reduction," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 1123–1132.

[60] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[61] R. Tomsett, D. Braines, D. Harborne, A. Preece, and S. Chakraborty, "Interpretable to whom? A role-based model for analyzing interpretable machine learning systems," 2018, *arXiv:1806.07552*.

[62] S. R. Hong, J. Hullman, and E. Bertini, "Human factors in model interpretability: Industry practices, challenges, and needs," *Proc. ACM Hum.-Comput. Interact.*, vol. 4, no. CSCW1, pp. 1–26, May 2020.

[63] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, J. Teevan, R. Kikin-Gil, and E. Horvitz, "Guidelines for human-AI interaction," in *Proc. CHI Conf. Hum. Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–13.

[64] F. Girardin and N. Lathia, "When user experience designers partner with data scientists," in *Proc. AAAI Spring Symp.*, 2017, pp. 1–15.

[65] Q. Yang, A. Steinfeld, C. Rosé, and J. Zimmerman, "Re-examining whether, why, and how human-AI interaction is uniquely difficult to design," in *Proc. CHI Conf. Hum. Factors Comput. Syst.* New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–13.

[66] V. Chen, S. Wu, A. J. Ratner, J. Weng, and C. Ré, "Slice-based learning: A programming model for residual learning in critical data slices," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9392–9402.

[67] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*. Boston, MA, USA: Addison-Wesley, 2003.

[68] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017, *arXiv:1702.08608*.

[69] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, no. 8, p. 832, Jul. 2019.

[70] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[71] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery," *Queue*, vol. 16, no. 3, pp. 31–57, Jun. 2018.

[72] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, Feb. 2019.

[73] L.-V. Herm, K. Heinrich, J. Wanner, and C. Janiesch, "Stop ordering machine learning algorithms by their explainability! A user-centered investigation of performance and explainability," *Int. J. Inf. Manage.*, vol. 69, Apr. 2023, Art. no. 102538.

[74] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," 2019, *arXiv:1908.09635*.

[75] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," 2017, *arXiv:1703.06856*.

[76] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proc. 3rd Innov. Theor. Comput. Sci. Conf.* New York, NY, USA: Association for Computing Machinery, Jan. 2012, pp. 214–226.

[77] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, Dec. 2016, pp. 3315–3323.
[78] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? Debiasing word embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 4349–4357.
[79] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," 2016, *arXiv:1606.06565*.
[80] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," 2019, *arXiv:1902.06705*.
[81] A. F. Cooper, E. Moss, B. Laufer, and H. Nissenbaum, "Accountability in an algorithmic society: Relationality, responsibility, and robustness in machine learning," in *Proc. ACM Conf. Fairness, Accountability, Transparency*. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 864–876.
[82] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," 2018, *arXiv:1804.11285*.
[83] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," 2018, *arXiv:1805.12152*.
[84] A. Raghunathan, S. Michael Xie, F. Yang, J. Duchi, and P. Liang, "Understanding and mitigating the tradeoff between robustness and accuracy," 2020, *arXiv:2002.10716*.
[85] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, 2010.
[86] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," 2016, *arXiv:1611.03814*.
[87] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.
[88] C. Dwork and G. N. Rothblum, "Concentrated differential privacy," 2016, *arXiv:1603.01887*.
[89] P. Kairouz et al., "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.
[90] L. J. M. Aslett, P. M. Esperança, and C. C. Holmes, "Encrypted statistical machine learning: New privacy preserving methods," 2015, *arXiv:1508.06845*.
[91] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 201–210.
[92] M. Brundage et al., "Toward trustworthy AI development: Mechanisms for supporting verifiable claims," 2020, *arXiv:2004.07213*.
[93] K. R. Varshney, "Engineering safety in machine learning," in *Proc. Inf. Theory Appl. Workshop (ITA)*, Jan. 2016, pp. 1–5.
[94] A. L'Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.
[95] S. Lo Piano, "Ethical principles in machine learning and artificial intelligence: Cases from the field and possible ways forward," *Humanities Social Sci. Commun.*, vol. 7, no. 1, pp. 1–7, Jun. 2020.

ANA GJORGJEVIKJ received the bachelor's degree in computer science and engineering and the master's degree in computer networks and e-technologies from Ss. Cyril and Methodius University in Skopje, in 2010 and 2014, respectively, where she is currently pursuing the Ph.D. degree in computer science and engineering, with a particular focus on deep learning and natural language processing. She has more than ten years of experience as a software engineer. Her research interests include data science, machine learning, and natural language processing.

KOSTADIN MISHEV received the master's degree in computer networks and e-technologies and the Ph.D. degree in computer science and engineering from Ss. Cyril and Methodius University in Skopje, in 2016 and 2023, respectively. He is currently an Assistant Professor with the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje. His research interests include data science, semantic web, natural language processing, enterprise application architectures, web technologies, and computer networks.

LJUPCHO ANTOVSKI is currently a Professor in software engineering with the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje, teaching courses related to project management, architecture of computers, mobile applications and platforms, design and architecture of software, and software requirements engineering. He has more than 20 years of experience in the IT area with a vast consultancy experience in projects related to application of technology in elections, electronic and mobile government, public and corporate information systems, software project management, and secure IT systems.

DIMITAR TRAJANOV (Member, IEEE) received the Ph.D. degree in computer science. From March 2011 until September 2015, he was the founding Dean of the Faculty of Computer Science and Engineering. He is currently a Full Professor with the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje and Visiting Research Professor with Boston University. He is also the Leader of the Regional Social Innovation Hub established, in 2013, as a cooperation between UNDP and the Faculty of Computer Science and Engineering. He is the author of more than 190 journals and conference papers and seven books. He has been involved in more than 70 research and industry projects, of which in more than 40 projects as a project leader. His research interests include data science, machine learning, natural language processing, FinTech, semantic web, open data, social innovation, e-commerce, technology for development, and climate change.