A taxonomy of software engineering challenges for machine learning systems: An empirical investigation

A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation

Lucy Ellen Lwakatare[1(✉)], Aiswarya Raj[1], Jan Bosch[1], Helena Holmström Olsson[2], and Ivica Crnkovic[1]

[1] Department of Computer Science and Engineering, Chalmers University of Technology, Hörselgången 11, 412 96 Gothenburg, Sweden
{llucy,aiswarya,jan.bosch,ivica.crnkovic}@chalmers.se
[2] Department of Computer Science and Media Technology, Malmö University, Nordenskiöldsgatan, 211 19 Malmö, Sweden
helena.holmstrom.olsson@mau.se

Abstract. Artificial intelligence enabled systems have been an inevitable part of everyday life. However, efficient software engineering principles and processes need to be considered and extended when developing AI- enabled systems. The objective of this study is to identify and classify software engineering challenges that are faced by different companies when developing software-intensive systems that incorporate machine learning components. Using case study approach, we explored the development of machine learning systems from six different companies across various domains and identified main software engineering challenges. The challenges are mapped into a proposed taxonomy that depicts the evolution of use of ML components in software-intensive system in industrial settings. Our study provides insights to software engineering community and research to guide discussions and future research into applied machine learning.

Keywords: Artificial intelligence · Machine learning · Software engineering · Challenges

1 Introduction

Artificial intelligence (AI) has gained much attention in recent years. Software-intensive companies, such as Facebook [5], are increasingly employing machine learning techniques in development of intelligent applications. Machine learning (ML), as a rapidly developing branch of AI, provides the companies with key capabilities for improving and accelerating innovation in their offerings based on operational system data. The application areas of ML to real-world problems are vast and range from large use in recommendation systems of social [9] and e-commerce [10] services, to highly regulated products, such as autonomous vehicle prototypes. The development of AI-enabled applications in real-world settings is

P. Kruchten et al. (Eds.): XP 2019, LNBIP 355, pp. 227–243, 2019.
https://doi.org/10.1007/978-3-030-19034-7_14

non-trivial and the development process differs from that of traditional software. At present, there is a growing interest and need to understand how AI-enabled applications are developed, deployed and maintained over-time in real world commercial settings.

It is observed that three distinct approaches, namely requirements-driven, out-come driven and AI-driven, are used to create software [3]. AI-driven approach in operational commercial software is the least covered approach in literature. The development process of AI-enabled applications that employ ML techniques, including its subset deep learning (DL), involve creation of ML models based on data. When creating ML models, typically several experiments are conducted prior to selecting the final ML model. During ML model creation, learning algorithms are applied to a dataset to train and evaluate the accuracy and performance of created ML models. Although in academia much focus is given to theoretical breakthroughs of learning algorithms, empirical studies show that they constitute only a small part of the operational ML system [20]. As a consequence, several challenges are encountered in practice during development and maintenance of ML systems [6]. To address the problem, emerging evidence highlights the need to take into consideration and extend established software engineering (SE) principles, approaches and tools in development of ML systems [11,19].

The main objective of this study is to identify and classify engineering challenges for developing and deploying ML systems in real world commercial settings. Using a multiple-case study approach, we explore the development of seven ML components of commercial software-intensive systems. The main contributions of the paper are threefold. First, the paper provides a description of the development process of six AI-enabled applications across various domains. Second, it presents a taxonomy to depict evolution in the use of ML components in commercial software-intensive systems. Third, using the taxonomy a classification of most important challenges at each stage of the evolution in the use of ML components in software-intensive systems is presented.

2 Background and Related Work

The research area of this study is applied ML, wherein the focus is to create verifiable knowledge pertaining to the design of software systems that incorporate ML techniques [14]. In our study, the considered software systems not only incorporate ML techniques to real world problems but are in operational use in commercial settings. This is in contrast to application of ML techniques to activities of software development process in field of SE [23], such as fault prediction and localization in software testing, which also gives numerous benefits in practice [16].

There exists empirical studies [6] and experience reports [7,15,19,21] published across different disciplines that present an end-to-end development process and challenges of operational AI-enabled applications. In a field study of how intelligent systems are developed, Hill et al. [6] describe a high-level process

that includes the following activities that are not necessarily sequential: *defining problem, collecting data, establishing ground truth, selecting algorithm, selecting features and creating and evaluating ML model*. Most of the challenges identified by the authors [6] at each activity of the ML process as well as cross-cutting issues are reported in other empirical reports [1,19]. For instance, the use of informal methods to manage dataset and common artifacts (trained models, feature sets, training jobs) during ML model selection experiments is a challenge that is commonly observed and presents difficulties to quickly reproduce and compare different experiments [18]. In addition to using agile approach for quick iterations [19], among the solutions proposed include using versioning in ML pipelines [22] and automating tracking of metadata and provenance information of the common artifacts [18]. However, some challenges are yet to be addressed, such as tracking provenance of complex final model that combines variety of models trained on different dataset [18] and data generated and processed through highly-heterogeneous infrastructure [13]. Concerning ML infrastructure several challenges are encountered, such as the ability to train models with large data volumes [5].

Using *technical debt* metaphor of SE, Sculley et al. [20] bring to awareness the different trade-offs involved, and require careful consideration, when maintaining ML systems overtime in real-world industrial settings. According to the authors [20], technical debt in real-world ML systems is attributed to maintenance problems of application code and issues specific to ML, such as data dependencies. ML systems have various sources of variability that need to be stabilized otherwise they can cause significant differences between ML models [8]. On the other hand, difficulties in debugging DL systems is currently one of the challenging topic that is gaining much focus in research [4]. Our study seeks to provide a taxonomy that can be used to consolidate the different challenges reported in prior empirical reports.

3 Research Method

For the purpose of this study we conducted an interpretive multiple-case study, following the guidelines by Runeson and Höst [17], to provide a deeper understanding of SE challenges for developing and operating ML systems in real-world commercial settings. The overall research design and process is described below.

3.1 Multiple-case Study Design

The multiple-case study research method was selected because it allowed us to explore SE challenges for developing ML systems in real-world settings both within and between cases. A case in our study pertains to a software-intensive system that incorporates ML component(s) developed at an organization.

Primarily we used semi-structured interviews to collect qualitative data. At the initial phase, we planned the study by describing the kind of data and practitioners that are relevant to the study. Based on the research objective and

Table 1. Description of domain of studied software-intensive systems, ML components, and roles of interviewees (*previous work)

Case	Domain	Use case of ML components	Interviewed experts	
			ID	Role
A	Automotive	Interpreting sensor data to understand the contained information at a high level	P1	Manager of DL organisation
B	Web	(i) Automating tagging of sentiments in online music library (ii) predicting quality of end product based on different measurements from machines, IoT devices of pulp processing and quality measures	P2	Data scientist
			P3	Head of data science team
			P4	Data scientist
			P5	UX lead
C	Web	Collaborative annotation of training data and predicting quality of annotations	P6	Co-founder
			P7	ML engineer
D	Telecom	Predicting failures at site to give insights into mobile network operations	P8	Project Tech lead
			P9	Senior researcher
			P10	Researcher
E	Web	Automating information extraction from out-of-office reply to optimize communication between sales reps and prospects	P11	VP data science
F	Web	Models of ML components of e.g., web search engine, are compared using important measures through A/B tests	P12	Data scientist, experimentation
			P11*	Principal data scientist

preliminary literature review, an interview guide was developed and reviewed in two iterations by the authors. The interview guide had a total of 18 questions structured in six sections. Background and context information of interviewee and ML system were inquired in the first two sections. Section three focused on the general development process of ML system. Sections four and five inquired in detail about data management, feature engineering, model building and deployment. Section six focused explicitly on perceived challenges of SE for ML systems. The interview guide was piloted with an external researcher prior to data collection. For the actual data collection, practitioners with experiences in developing real-world ML systems were sought from different organizations.

3.2 Data Collection

Our primary qualitative data collection process started by sending e-mails containing a short description of study's objective to different company representatives of Software Center[1] and others based on authors' personal networks. The email requested for their company's participation in the study and selecting suitable persons for the interview. Semi-structured interviews were conducted

[1] Software Center: https://www.software-center.se/.

with practitioners between August and December 2018. Altogether 15 interviews were carried out with professionals and allowed researchers to reach saturation of knowledge. Considering acknowledgment in literature (and our experiences when soliciting interviewees) that there are few experienced practitioners skilled in the area of intersection between ML and SE, the vast experience of our interviewees from different companies across multiple domains as shown in Table 1 is of great advantage to this study.

Three interviews were excluded from the study as they focused on the application of ML to the activities of software development process (see Sect. 2 for this study's research area). All interviews were face-to-face except for two interviews, which were done via teleconference. The duration of each interview ranged between 45 to 70 min. All interviews were recorded with interviewees' approval and were later transcribed for analysis. An opportunity for the follow-up questions was also agreed at the end of all interviews. After each interview session, a summary was written and discussed among authors. Secondary qualitative data was collected at a workshop held in December 2018 with practitioners where our initial research results were shared. Excluding researchers, the workshop had an attendance of 10 practitioners from different companies in automotive and telecommunications fields. Notes were taken by authors during workshop.

3.3 Data Analysis

Thematic analysis was used to analyse qualitative interview data. Interview transcripts were coded in NVivo by two researchers in two iterations. First coding iteration was done by coding the challenges using a set of six predefined high-level themes that depict the development of ML system. The six high-level themes were: *(i) data management and pre-processing, (ii) create model, (iii) train and evaluate model, (iv) model deployment, and (v) organizational issues*. First, the two researchers familiarized with the data and discussed the coding procedure at high level themes. Thereafter, the researchers coded separately a similar transcript in order to determine inter-rater reliability measure. A good agreement level was determined (overall Kappa value = 0.72)[2] and the researchers discussed disagreements when reviewing outputs of code comparison in NVivo. Coding for the rest of the transcripts proceeded by having each transcript coded by one researcher. Second iteration of the coding involved identifying challenges at each high-level theme. This was done through joint discussion between the two researchers. The notes taken during the workshop were reviewed and important aspects were taken into consideration in the reported taxonomy.

4 Case Study Findings

An overview of the cases and findings of the challenges from each case are presented. Several dimensions, such as learning task and source of training dataset

[2] Interpretation of kappa value in NVivo: Poor agreement (Below 0.40), Good agreement (0.40–0.75), Excellent agreement (Over 0.75).

[23], can be used to describe and differentiate ML systems. For each case, description of the software-intensive system incorporating ML component(s) is presented first and then followed by descriptions of the ML use case, source of training data, training and deployment of ML models. The ending paragraph of each case description presents a summary of the main challenges for developing ML system as perceived by the interviewed practitioners.

4.1 Case A: *Software for Automated Driving*

A joint venture company established by two large companies in automotive domain is developing software for automated driving (AD). DL models are used in development of software for AD where the main use case is perception. Perception is interpretation of sensor (e.g., camera and LIDAR) data to understand at a high level the contained information, such as presence of objects (e.g., pedestrians and vehicles) among other. DL organization consists of about 50 persons software engineers and ML experts responsible for training DL models, developing DL infrastructure and managing large data storage.

Large amounts of data are collected from the fleet of vehicles going on expeditions in different parts of the world. The collected data is transported and stored at a big data center where pre-processing methods are used to extract images that are to be annotated by an external company. ML experts use annotated data to train offline DL models. To quickly train DL models and rapidly iterate product development cycle, several graphics processing units (GPUs) are used. Software engineers develop different tool-chains, such as schedulers for training jobs, diagnostic and monitoring tools to the highly scalable DL infrastructure. Perception inference serves as a foundation and input to other layers of the system with decision and control are developed by other teams. These teams together with other stakeholders of DL organisation have specifications that give inputs to key performance indicators (KPI) used in model evaluations. The primary target for deployment of DL model in AD vehicle was GPUs, like the NVIDIA Drive Xavier [12].

While Case A poses issues and extreme requirements for storage and preprocessing of large data sets for creating DL models in automotive domain, one main engineering challenge perceived by the DL organization manager was difficulty in building DL infrastructure, as expressed below. This is in addition to problems related evolving requirement definition for AD vehicles, which affect model training and evaluations. At the time of the interview, there was limited support for quickly recreating the different training results.

> *There are no tool-chains you can download in an infrastructure with deep learning like this. And we realized after the mistakes and discussions with our new IT that they didn't really have the expertise to be able to deliver this to us. So we had to create new teams, which took the responsibility of creating both the infrastructure, but also the software tool-chain to be able to train deep learning networks within a reasonable amount of time.*

4.2 Case B: *AI Web Platform*

AI web platform is developed by a company that simplifies the development of ML applications. At the time of interview, the beta version of the platform had active customers using the platform from various industries. Two main use cases of ML focused on our study were for: (i) automatic tagging of sentiments in online music catalogue, and (ii) predicting quality of manufactured products (e.g., carton, cardboard, paper) based on measurements from different machines, IoT devices and microscopic images of wood fibres for pulp processing. AI platform clients typically know beforehand their ML use cases and have data available. Software developers in product team are developing the platform in collaboration with a data science team. The data science team, consisting of eight persons, communicate requirements to the product team, provides internal AI education and uses the platform to do projects with external companies.

In the studied ML use cases, a data scientist receives a training dataset, which is uploaded onto the AI platform. The training dataset is explored, curated and checked for quality on the platform. In the training set, tags initially applied by humans' (e.g., content managers) are assumed to be satisfactory to the client. Since the customer in the paper mill industry had efficient data pipelines for collecting various measurements from pulp processing and quality, data science team were able to get data from past several years. Using the training dataset, different ML models are built, trained and evaluated offline on the platform. For automatic sentiment tagging, the selected final DL model is deployed to a Kubernetes cluster in cloud infrastructure allowing among other, scaling. It exposes a REST API that can be called via JavaScript fetch from the online music catalogue application. At the time of interview, trained ML models of the output quality predictions of pulp processing had not been deployed in production but yielded feedback in form of report given to clients about features indicative of quality. The later was among factors considered in decision to buy a new machine.

In addition to challenges of developing AI platform, such as managing design trade-offs in customization of platform functionalities, other challenges concerned handling of data drifts in uploaded data, invalidation of models e.g., due to changes in data sources, and the need to monitor models in production for staleness.

> *You're trying to simultaneously build reproducibility, collaboration and ease of use at the same time you're trying to give people as much customization as possible. It's the difference between giving somebody a notebook where they can do anything they want and giving a higher level tool that has a lot of built-in functionality. It's there that I see most challenges*

4.3 Case C: *Collaborative Annotation Web Platform*

The collaborative annotation web platform is for creating training dataset of supervised learning used in the development process of customers' ML systems.

The company's clients are mostly automotive OEM companies. In addition, the platform incorporates ML model to predict reliability of an annotations. An annotation process designed by the company is collaborative through iterative development of annotation guideline that incorporates quick feedback between human annotators and the customer's stakeholders. Through the annotation guideline, customers express the desired outcome at an acceptable standard and level of error tolerance. At the time of the interviews, the company had seven employees.

The dataset from the customer is uploaded on the platform and a sample of it is given to both the customer and human annotators to annotate. This is done to determine uncertainty level using for example heat maps. Depending on the results, the customer gets an opportunity to improve annotation guideline thereby shortening the feedback-loop between customer and human annotators. Human annotators use the improved guideline to annotate dataset on the platform. While doing the annotations, meta-data is recorded e.g., time taken to annotate, number of clicks etc. From this data and reviews given by peers, a detailed Bayesian model is developed for each annotator to estimate the quality of the annotations and predict the probability that an annotator is able to produce what the customer wants. The model is running in a Google cloud environment and hooked to the platform through client calls that get executed whenever human annotators finish annotations.

Main SE challenges identified from Case C is the need for processes and tools for forming accurate and consistent annotations in large dataset, especially when the system has no self-labelling instrumentation. Furthermore, there are difficulties in negotiating interpretations and dealing with poor inter-rater agreement across a large group of annotators. Customers using the annotated dataset, often do not have other mechanisms to know if the annotations were done correctly.

> *"So the challenging part of creating large amounts of examples is that it's usually ambiguous. You have a distributed group of people and you need very low error tolerance, because if you're going to have production grade machine learning systems, their performance will be governed by the quality of the data"*

4.4 Case D: *Mobile Network Operations*

A large telecommunication company is enabling intelligent operations of networks by introducing ML techniques that help to predict issues at a source. For network operations centre (NOC) personnel, this allows them to automate and proactively evaluate, prioritize and take preventative actions on issues that might arise. ML use cases focused in this study are those from a project where a research team doing thought leadership at the company is involved and the main goal is to predict what can go wrong at a site (i.e., a building that has base stations, antennas, auxiliary power sources etc). Example of specific ML use cases, include predicting degradation of KPI e.g., latency and throughput, to facilitate remote troubleshooting; and predicting site's sustenance to power

outage from auxiliary power e.g., using frequency of battery charging as input data. NOCs that are operated by the company are for about 400 client operator companies distributed in different locations.

Depending on the use case, and whether the team is allowed to move data, datasets of varying sizes are used to train models. In extreme scenario with a datasets of 3TB per day and where data is not allowed to be moved outside a country, federated learning is used. In federated learning an initial model is built locally and then it gets trained and improved at the edge. The training dataset is curated and features engineered by data scientists prior to training. ML models are trained while also residing in the CI-CD pipeline since the company supports many customers across different locations. When training the models care is taken not to mix data of different clients. The ML models are packaged as Docker images that are deployed on Kubernetes in the cloud and monitored for model usage and accuracy, in addition to CPU usage and memory, using a tool called Prometheus.

The main engineering challenges for Case D are related to data collection and model localization particularly in areas where data movement is constrained, as elaborated in quote below by the Tech Lead.

> *I think really the challenge is actually getting data and that is why we are investing so much in federated learning because in some cases the data cannot leave the country. And also in some cases the links that you have are not strong enough to carry the data that you want because they are used by other things. So that is really the key challenge here and that is why we are looking into the techniques such as federated learning and reinforcement learning so that we can improve on it.*

4.5 Case E: *Sales Engagement Platform*

Sales engagement platform primarily enables and optimizes communication between sales representatives (henceforth sales reps) and potential prospects. Sales communication occurs in natural language via different communication channels, including emails. The ML use case of focus was concerned with extracting automatically entities, such as date, using natural language processing (NLP) techniques from out-of-office emails. Specifically, for information extraction, emails are parsed and processed to understand the contained information, such as people, dates and best contact information from out-of-office emails. The information allows sales reps to take relevant actions, such as pausing sequences of automated steps. The data science team at the company consists of ten persons responsible for data analysis, ML, A/B testing and insights reporting.

All email communication done by sales reps is stored in a communication database out of which a few of these are labelled and form the validation dataset. Due to some factors, such as limited labelled data to train models, open sourced pre-trained models are used. Prior to extraction of entities, pre-processing steps, such as handling of different encoding, are conducted to get the email text. The

step is followed by the entity extraction, which applies different pre-trained models to extract entities as well as construct relationship tree around the entities, for example to suggest the person with whom the phone number left in out-of-office email belongs. The pre-trained models are evaluated using the validation dataset and tuned to improve their accuracy in consideration of company's dataset. In addition, measures of actual user experience through A/B testing are gathered to provide feedback into the training of the model. Databricks tool is used to build and deploy the models, which are typically saved as a single library and are version controlled.

Prior to their recent use of the Databricks tool, the team faced challenges related to the lack of standardized approaches for reproducing model selection experiments quickly and scaling models in production.

> *We also needed to worry about scaling because the volume of messages differs a lot with time. Generally, on Monday sales people send hundreds of thousands of emails to new prospects. We had to either do it manually by deploying more copies of the model, and then bringing them down to not use-up resources, or leave the model and then there will be a queue. We did not do this manual approach and the queue would get to almost 24 hours long. So those emails will only get processed on Tuesday because of the volume.*

4.6 Case F: Online Experimentation Platform

An experimentation platform is developed by a large company to support various product teams in running trustworthy experiments, such as A/B tests. While the platform is also used by applications that do not incorporate ML/DL components, those that do use, such as web search engine, use it to compare trained models with important measures through A/B tests. The team in charge of the platform provide training and support to product teams to set-up, run experiments in addition to developing and maintaining the platform. The team consists of about consists of about one hundred persons, who among them are data scientists and software developers.

Experimentation platform consists of mainly four components namely experimentation portal, experiment execution service, log processing service, and analysis service. A good logging system that captures correct events, at correct time and identify targets is important for running experiments on the platform. This is because every product user is every single point in time in several experiments and logs need to be annotated with information of which experiments users in addition to using the data to (re)train models. Users can run their experiments on platform as per their requirement either with the help of some predefined templates or without templates by which they can eventually find a better performing trained model. This is important because the models are compared using measures that the business care for because users are using system functionality and not the models.

Among the challenges identified from practitioners of Case E are difficulties with complex and poor logging mechanisms as well as in designing experiments, including interleaving experiments often done in ML components and interpreting experiment results.

> *If product teams want to have good informative experiments they need to log the correct things. Logging in the past was done to understand if a product has crashed or not, or why it has crashed. This is not sufficient if you want to compute good business metrics in the end of the day*

5 A Taxonomy of SE Challenges for ML Systems

In this section, insights into SE challenges for ML system are presented using a taxonomy that depicts evolution of use of ML components in software-intensive system in industrial settings. The insights are based on the findings of our cross-case analysis and the literature presented in Sect. 2.

Based on the study, we have identified five evolution stages of the use of ML component(s) in software systems that follow a pattern wherein they are initially deployed for experimental (or research) purposes until maturing to function autonomously. This progression of stages in the taxonomy occurs at component basis. Essentially, model life-cycle activities (*assemble dataset, create model, (re)train and evaluate, deploy*) are performed at all maturity stages. The taxonomy is visualised in Fig. 1 and a summary of the challenges is given in Table 2.

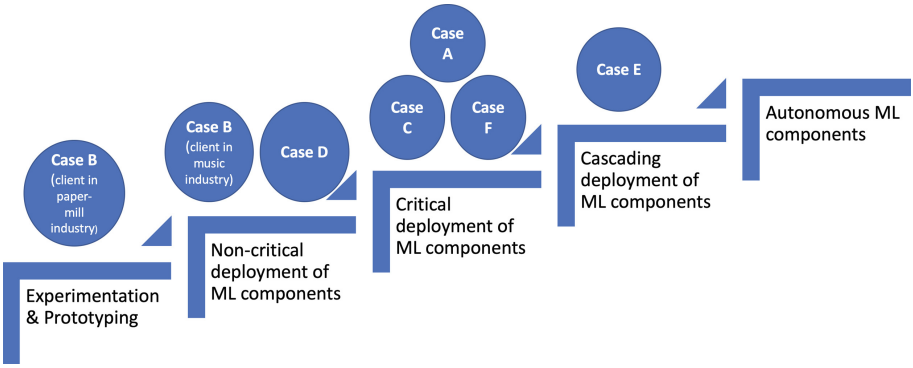

Fig. 1. Evolution of use of ML in commercial software-intensive systems

5.1 Experimentation and Prototyping

Initial application of ML techniques and creation of ML models in industrial settings is often for experimentation and prototyping. At this first step, different innovations and improvements to existing software-intensive systems are

hypothesized through the use of ML techniques. There exists a vague description of ML problem or use case. However, critical decisions are made by designers about the learning algorithm, representation and training dataset. Decisions have profound effects on the success, or appropriateness, of ML components. In our studied cases, Case B's client from paper-mill industry belongs to this stage.

Existing data collection mechanisms and storage of software-intensive system are originally not set-up for ML systems. For example, poor logging and limited data cleaning mechanisms exist prior to the ML initiative. As a result, potentially large efforts are spent on data exploration, in addition to determining and formulating the problem of ML in the respective application domain. Difficulty in formulating the problem for ML is accounted for, among other, by the need to determine beforehand a benchmark or baseline against which ML model will be evaluated for accuracy and performance optimizations. While a variety of big data tools are used in data aggregation and structuring, different design decisions and trade-offs in model creation rely on inputs from domain experts, such as useful features. At this stage, models are not deployed but do provide valuable feedback to the experts about the direct impact of suggested features.

5.2 Non-critical Deployment

After gaining experience with initial use of ML techniques in respective application domain, ML model prototypes, or their revisions, are deployed on non-critical functions of the software-intensive system. Alternatively, inferences of ML components are inspected by human expert. At this stage, the hypothesized improvements through the use of ML are quantified in a production environment. As data pipelines for ML are being initiated, scarcity of labelled data and imbalanced training dataset challenge the creation of models. This is in addition to legal and privacy protection requirement challenges of accessing data. From our studied cases, Case B's client in music industry and Case D belong to this stage.

Data analysis and validation is an initial and critical activity for designers of ML components. Absence of critical analysis on training data results to training and serving skew, which describes differences in performance of ML model at training and deploy. The discrepancies are largely caused by differences in the handling of data distributions and pipelines during training and operations. Data sources or different fields in data e.g., in logs, may come from different components owned by other teams. Major changes to the values invalidate models trained on older data. Techniques and tools for monitoring and tracking data are crucial for developing ML systems, as supported in literature.

5.3 Critical Deployment

Successfully quantified improvements in stage 2 drive deployment of ML models to critical functions of the software-intensive system. For each critical function

implementing ML component, designers take a separate account to their differences in system architecture and data distributions both at training and serving. At this stage, there is typically a co-existence of ML components with other software components developed using traditional techniques, as observed in Case A. Since ML components are developed at the same time as the product definition of the software-intensive system is evolving, the ability to track and adapt to the system changes and optimization objectives is necessary at training and evaluation. There is need to have an effective end-to-end ML pipeline that simplify and make it possible to quickly compare and reliably reproduce different results of model creation, training and evaluations. Case A, Case C and Case F belong to this stage.

For critical deployment of ML components, challenges in the implementation of the end-to-end ML pipeline comes with the need and difficulty in implementing an effective experimentation infrastructure. The experimentation infrastructure is used to evaluate performance improvements and effects of ML models with the use of metrics that are business-centric rather than algorithmic-centric. The ability to design and conduct several experiments on continuous basis is non-trivial. While experiments that are conducted online are exploring and exploiting the models, end-users are not to be affected and need to adhere to the stringent requirements of latency and throughput.

5.4 Cascading Deployment

At the next step is a software system that has cascading ML models whereby outcome(s) of one or more ML components serve as inputs to subsequent ML component. Cascading deployment of ML components was used in [10] to enable the elimination of irrelevant data items in earlier stages and discern relevant ones in later stages. According to the authors [10], the cascading deployment strategy achieves a balance, rather than a trade-off, with respect to effective ranking results from a large number of data items of Alibaba e-commerce and efficiency in terms of good user experience and savings in computational costs. From our cases, Case E, in addition to the studied use case, has models that detect the intent from prospect's email replies.

For the final model in the cascading deployment strategy, the challenge comes from the difficulty in tracking changes in models giving the input features and in performing a sliced analysis to the evaluation results. It is apparent that as the system scales to handle more models, it becomes difficult to identify the cause of poor performance for final system, for example due to undeclared consumers [20]. When final model performance results are not sliced, such as merely focus on accuracy on validation training set, according to [15] important effects are masked and can result in quality improving in one part but degrading in another.

5.5 Autonomous ML Components

At the final step is a system that incorporates ML components that have automatic processes (or minimal human intervention) of ensuring fail-safe outcomes,

Table 2. Summary of the challenges in evolution of use of ML components in commercial software-intensive systems

	Experiment prototyping	Non-critical deployment	Critical deployment	Cascading deployment
Assemble dataset	Issues with problem formulation and specifying desired outcome	Data silos, scarcity of labelled data, imbalanced training set	Limitations in techniques for gathering training data from large-scale, non-stationary data streams	Complex and effects of data dependencies
Create model	Use of non-representative dataset, data drifts	No critical analysis of training data	Difficulties in building highly scalable ML pipeline	Entanglements causing difficulties in isolating improvements
Train and evaluate model	Lack of well-established ground truth	No evaluation of models with business-centric measures	Difficulties in reproducing models, results and debugging DL models	Need of techniques for sliced analysis in final model
Deploy model	No deployment mechanism	Training-serving skew	Adhering to stringent serving requirements e.g., of latency, throughput	Hidden feedback-loops and undeclared consumers of the models

retraining and scalability of ML models. While we did not experience a case at this stage besides being expressed by practitioners as a future direction, concrete work in this direction is presented in existing literature, such as in online targeted-display advertising systems [15]. This stage also considers other learning strategies, such as active learning and reinforcement learning. Case A of our case study findings was considering active learning for automatic selective acquisition of training data and Case D was looking to explore reinforcement learning to eliminate efforts associated with model training and retraining.

For the alternative learning methods, such as active learning, some of the challenges are attributed to the lack of sufficient practical guidance for implementing the learning strategies [2]. Although, successful implementations have been demonstrated to obtain data and annotations automatically through the use of active learning, there remains great need to incorporate practices and tools for monitoring data sources and monitoring different sources of variability.

6 Conclusion

Developing, evolving and operating ML systems in real-world commercial settings is non-trivial. This paper explored engineering challenges for developing and operating supervised ML systems in real-world commercial settings. Multiple cases of ML systems from different application domain are presented, including description of their development process and perceived engineering challenges.

In an effort to energize and focus the discussion of ML systems on SE aspects besides the algorithmic issues, we have presented a taxonomy that depicts maturity stages of use of ML components in commercial software system and mapped the challenges at each stage. The challenges we have identified as most important require a lot of efforts to be managed, and in the future work we will refine the challenges with additional cases and explore possible solutions as well as provide guidance on how to move from one maturity stage to another. Furthermore, we acknowledge that our study has narrowly focused on the development process of ML components and that research into other SE topics, such as challenges related to software architecture are still of great interest and needed.

References

1. Arpteg, A., Brinne, B., Crnkovic-Friis, L., Bosch, J.: Software engineering challenges of deep learning. In: 44th Euromicro Conference on Software Engineering and Advanced Applications, pp. 50–59. IEEE (2018). https://doi.org/10.1109/SEAA.2018.00018
2. Attenberg, J., Provost, F.: Inactive learning? Difficulties employing active learning in practice. ACM SIGKDD Explor. Newsl. **12**(2), 36–41 (2011)
3. Bosch, J., Olsson, H.H., Crnkovic, I.: It takes three to tango: requirement, outcome/data, and AI driven. In: International Workshop on Software-Intensive Business: Start-Ups, Ecosystems and Platforms, pp. 177–192 (2018)
4. Hains, G., Jakobsson, A., Khmelevsky, Y.: Towards formal methods and software engineering for deep learning: security, safety and productivity for dl systems development. In: 2018 Annual IEEE International Systems Conference, pp. 1–5. IEEE, April 2018. https://doi.org/10.1109/SYSCON.2018.8369576
5. Hazelwood, K., et al.: Applied machine learning at Facebook: a datacenter infrastructure perspective. In: International Symposium on High Performance Computer Architecture, pp. 620–629. IEEE (2018). https://doi.org/10.1109/HPCA.2018.00059
6. Hill, C., Bellamy, R., Erickson, T., Burnett, M.: Trials and tribulations of developers of intelligent systems: a field study. In: Symposium on Visual Languages and Human-Centric Computing, pp. 162–170. IEEE (2016). https://doi.org/10.1109/VLHCC.2016.7739680
7. Kumar, R.S.S., Wicker, A., Swann, M.: Practical machine learning for cloud intrusion detection: challenges and the way forward. In: 10th Workshop on Artificial Intelligence and Security, pp. 81–90. ACM (2017). https://doi.org/10.1145/3128572.3140445
8. Lefortier, D., Truchet, A., de Rijke, M.: Sources of variability in large-scale machine learning systems. In: Machine Learning Systems (NIPS 2015 Workshop) (2015)
9. Lin, J., Kolcz, A.: Large-scale machine learning at Twitter. In: SIGMOD International Conference on Management of Data, pp. 793–804. ACM (2012). https://doi.org/10.1145/2213836.2213958
10. Liu, S., Xiao, F., Ou, W., Si, L.: Cascade ranking for operational e-commerce search. In: International Conference on Knowledge Discovery and Data Mining, pp. 1557–1565. ACM (2017). https://doi.org/10.1145/3097983.3098011

11. Murphy, C., Kaiser, G.E., Arias, M.: An approach to software testing of machine learning applications. In: 19th International Conference on Software Engineering and Knowledge Engineering, pp. 167–172. Knowledge Systems Institute Graduate School (2007)
12. NVIDIA: Nvidia drive hardware for self-driving cars. https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/. Accessed 11 Jan 2019
13. Polyzotis, N., Roy, S., Whang, S.E., Zinkevich, M.: Data management challenges in production machine learning. In: International Conference on Management of Data, pp. 1723–1726. ACM (2017). https://doi.org/10.1145/3035918.3054782
14. Provost, F., Kohavi, R.: Guest editors' introduction: on applied research in machine learning. Mach. Learn. **30**(2), 127–132 (1998). https://doi.org/10.1023/A:1007442505281
15. Raeder, T., Stitelman, O., Dalessandro, B., Perlich, C., Provost, F.: Design principles of massive, robust prediction systems. In: International Conference on Knowledge Discovery and Data Mining, pp. 1357–1365. ACM (2012)
16. Rana, R., Staron, M., Hansson, J., Nilsson, M., Meding, W.: A framework for adoption of machine learning in industry for software defect prediction. In: 9th International Conference on Software Engineering and Applications, pp. 383–392. IEEE (2014)
17. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**(2), (2008)
18. Schelter, S., Böse, J.H., Kirschnick, J., Klein, T., Seufert, S.: Automatically tracking metadata and provenance of machine learning experiments. In: NIPS Workshop on Machine Learning Systems (2017)
19. Schleier-Smith, J.: An architecture for agile machine learning in real-time applications. In: International Conference on Knowledge Discovery and Data Mining, pp. 2059–2068. ACM (2015). https://doi.org/10.1145/2783258.2788628
20. Sculley, D., et al.: Hidden technical debt in machine learning systems. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2503–2511. Curran Associates, Inc. (2015)
21. Tata, S., et al.: Quick access: building a smart experience for Google drive. In: 23rd International Conference on Knowledge Discovery and Data Mining, pp. 1643–1651. ACM (2017). https://doi.org/10.1145/3097983.3098048
22. van der Weide, T., Papadopoulos, D., Smirnov, O., Zielinski, M., van Kasteren, T.: Versioning for end-to-end machine learning pipelines. In: 1st Workshop on Data Management for End-to-End Machine Learning, pp. 2:1–2:9. ACM (2017). https://doi.org/10.1145/3076246.3076248
23. Zhang, D., Tsai, J.J.: Machine learning and software engineering. Softw. Qual. J. **11**(2), 87–119 (2003). https://doi.org/10.1023/A:1023760326768