Noname manuscript No.
(will be inserted by the editor)

A Multi-solution Study on GDPR AI-enabled Completeness Checking of DPAs

Muhammad Ilyas Azeem[a] · Sallam Abualhaija[b*]

Received: date / Accepted: date

Abstract Specifying legal requirements for software systems to ensure their compliance with the applicable regulations is a major concern of requirements engineering. Personal data which is collected by an organization is often shared with other organizations to perform certain processing activities. In such cases, the General Data Protection Regulation (GDPR) requires issuing a data processing agreement (DPA) which regulates the processing and further ensures that personal data remains protected. Violating GDPR can lead to huge fines reaching to billions of Euros. Software systems involving personal data processing must adhere to the legal obligations stipulated both at a general level in GDPR as well as the obligations outlined in DPAs highlighting specific business. In other words, a DPA is yet another source from which requirements engineers can elicit legal requirements. However, the DPA must be complete according to GDPR to ensure that the elicited requirements cover the complete set of obligations. Therefore, checking the completeness of DPAs is a prerequisite step towards developing a compliant system. Analyzing DPAs with respect to GDPR entirely manually is time consuming and requires adequate legal expertise. In this paper, we propose an automation strategy that addresses the completeness checking of DPAs against GDPR provisions as a text classification problem. Specifically, we pursue ten alternative solutions which are enabled by different technologies, namely traditional machine learning, deep learning, language modeling, and few-shot learning. The goal of our work is to empirically examine how these different technologies fare in the legal domain. We computed F_2 score on a set of 30 real DPAs. Our evaluation shows that best-performing solutions yield F_2 score of 86.7% and 89.7% are based

[a] This work was done while the author was affiliated with SnT, University of Luxembourg
E-mail: ilyasazeem@live.com
[b] SnT, University of Luxembourg, Luxembourg
E-mail: sallam.abualhaija@uni.lu
[*] Corresponding Author

on pre-trained BERT and RoBERTa language models. Our analysis further shows that other alternative solutions based on deep learning (e.g., BiLSTM) and few-shot learning (e.g., SetFit) can achieve comparable accuracy, yet are more efficient to develop.

Keywords Requirements Engineering (RE) · The General Data Protection Regulation (GDPR) · Regulatory Compliance · Data Processing Agreements (DPAs) · Artificial Intelligence (AI) · Natural Language Processing (NLP) · Classification · Large Language Models (LLMs) · Few-shot Learning (FSL) · Data Augmentation.

1 Introduction

Legal requirements describe the behavior and functions of a software system pursuant to applicable regulations [48]. Developing compliant software systems requires the elicitation of legal requirements from regulations, an essential activity in the requirements engineering (RE) field. Both manual and automated approaches have been investigated for navigating through the regulations to create machine-analyzable representations and extract compliance-relevant information (e.g., [17,2,67,65,31]). Despite the community efforts, software development practices are still introducing personal data breaches, e.g., unauthorized sharing with third-parties [26]. The complexity of the legal language used in regulations poses a major challenge to software engineers in understanding as well as properly implementing legal requirements in software systems [62,1, 72]. Eliciting legal requirements is time-consuming and error-prone and often requires adequate legal expertise. For example, the General Data Protection Regulation (GDPR), enforced in the European Union (EU) in 2018 [24], contains provisions on data privacy and security with which organizations, inside or outside Europe, must comply as long as they collect and process personal data of people in the EU. The obligations set out in GDPR entail legal, technological, and operational changes. Understanding the requirements for complying with GDPR is a challenging task for organizations, specifically for small and medium-sized enterprises (SMEs) [29,47]. Existing literature discusses the significant impact GDPR has on Internet-of-Things (IoT) based applications as well as cloud-hosted services widely used in modern technologies in various domains like e-commerce, healthcare, and energy consumption [10,9]. Fines are being imposed yearly due to different types of breaches, most of which are caused by non-compliant practices in the software systems deployed by organizations. Statistics show that most violations are related to breaching data processing principles, leading to 418 fines of more than 1,500 billions of euros[1]. Eliciting legal requirements for a specific application context by navigating through the 88 pages of GDPR with its 173 recitals and 99 articles divided into 11 chapters is a daunting task for requirements engineers.

[1] https://www.enforcementtracker.com/?insights

To bridge the legal knowledge gap and optimize the elicitation of legal requirements, we argue in this paper that a *regulated document* (e.g., privacy policy) provides compliance-relevant information which can be easier to navigate than the entire regulation. Our work focuses on data processing agreements (DPAs), which are legal contracts between a *controller* who collects personal data and a *processor* who processes personal data on the controller's behalf [53, 6]. GDPR can be broken down into 1501 sentences containing generic legal requirements [5,1,66], whereas a DPA contains on average 200 sentences and provides a more specific overview tailored towards specific business needs [19]. It is not uncommon in the RE literature to use contracts as a source for extracting legal requirements [38,60].

To be an effective source document for eliciting legal requirements, DPAs must be GDPR-compliant, meaning that the content of a DPA must be *complete* according to the provisions of GDPR. Otherwise, the resulting legal requirements elicited from the DPA will be incomplete. Verifying the completeness of DPAs against GDPR is thus a pre-requisite for leveraging the DPAs as source knowledge for eliciting legal requirements for software systems that process personal data.

Fig. 1 shows an example of eliciting technical requirements from the legal statements in a given DPA. On the left side of the figure, we show the provisions from an excerpt from a GDPR-compliant DPA, and on the right side we show example requirements that can be elicited from the DPA. The DPA is between *@Ur-Home Ltd*, a large e-commerce company selling various products, and *Data2Info Corp*, a company that provides data analytics services. The former acts as the controller and the latter as the processor. The *Data2Info* company uses an electronic management system (EMS) for managing files. To comply with GDPR, the EMS must, among other things, incorporate security mechanisms to protect personal data and further delete all personal data received form the controller upon the termination date of the agreement. The first four requirements (labeled `SEC-1` – `SEC-4` in the figure) correspond to additional security mechanisms that need to be accounted for when storing files on the server. For instance, if a file is not encrypted, the user shall be immediately informed. The last three requirements (labeled `DEL-1` – `DEL-3` in the figure) are about the deletion of the data. Relying on incomplete DPAs would result in missing compliance requirements and developing a software system whose behavior does not adhere to the applicable regulations. For example, had the third provision not been in the DPA, the developed system would transfer personal data without sufficient protection mechanisms, exposing the data thereby to potential breach incidents. Failing to comply can have serious consequences such as large fines[2]. Our work in this paper helps requirements engineers verify through automated means whether a DPA misses provisions (i.e., incomplete) according to GDPR before using the DPA as a source for requirements elicitation. Completeness of software requirements is challenging to verify and cannot be defined in absolute terms [74]. However, existing work

[2] Meta was fined 1.3 billion due to violating GDPR. *[news article]*

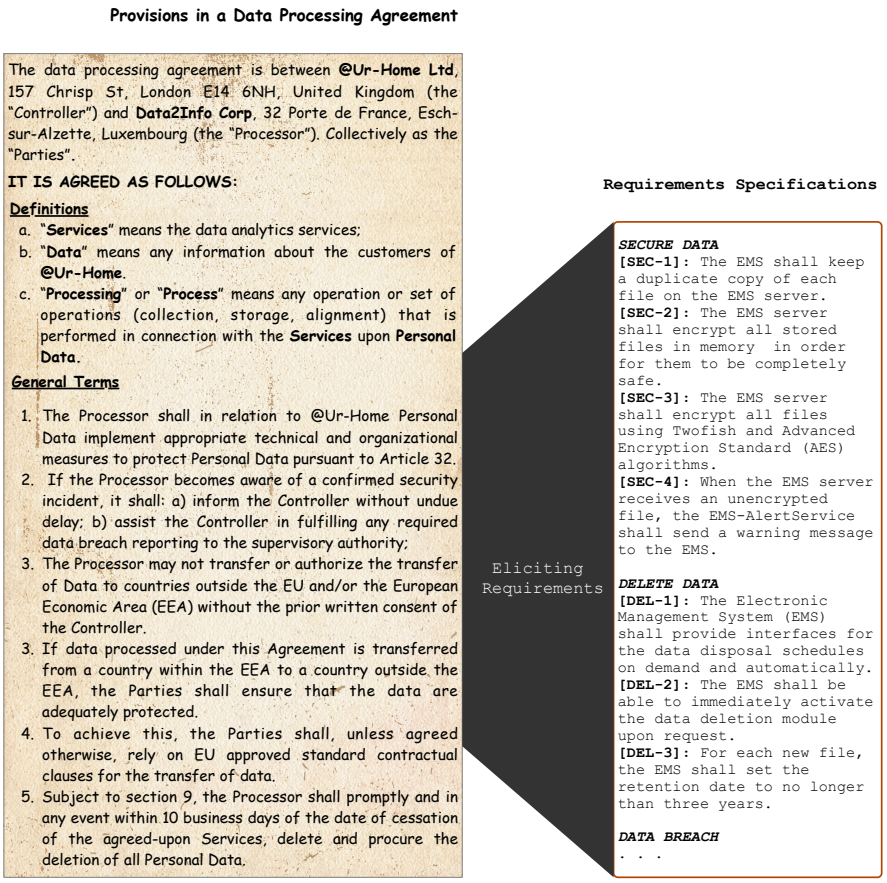

Fig. 1 Example of eliciting technical requirements from a DPA.

in the RE literature often analyze completeness vis-à-vis external knowledge sources [28,22,8,45]. If DPAs are complete, they can serve as such knowledge sources.

Checking the completeness of DPAs against GDPR has been investigated in a recent work by Amaral *et al.* [19]. Amaral *et al.* define, in collaboration with legal experts, a set of 45 GDPR provisions that regulate the content of a DPA. To conclude whether a DPA is complete according to GDPR, the authors develop a rule-based automated approach that verifies the textual content of a given DPA against the GDPR provisions. Their approach utilizes natural language processing (NLP) technologies to analyze the semantic similarity of the textual content in a DPA against GDPR. While their approach has demonstrated its effectiveness on checking the completeness of real DPAs, using rules has several drawbacks. First, regulations are subject to continuous changes which can impact, to a large extent, the compliance checking process [40,18]. Adjusting rules to cover such potential future changes requires

significant effort involving both the engineers who adjust the rules and the legal experts who confirm the changes. Second, there is a need for manually-labeled datasets. While rule-based approaches do not typically require training, creating datasets is still required for validating the rules and drawing conclusions on their generalizability. In the regulatory compliance context, even small datasets are expensive to create since analyzing regulated documents requires legal background. Third, the complexity of the rules grows with the complexity of the legal text in the regulated documents. The DPA text is typically long, convoluted, and attempts to simultaneously address multiple GDPR provisions. Thus, complex rules are necessary for checking the completeness of DPAs against GDPR. In a follow-up work, Amaral et al. [5] attempt to address the above limitations of rule-based techniques by utilizing machine learning (ML) for checking the completeness of DPAs. While using ML indeed alleviates the drawbacks of rules, their ML-based approach (i) does not exploit the potential of recent NLP technologies such as large language models (LLMs), and (i) it further requires large amount of training data to develop an accurate solution.

Drawing on Amaral *et al.*'s work, we propose in this paper ten alternative solutions which are based on different enabling technologies, including traditional ML, deep learning (DL), as well as recent NLP technologies featured by LLMs and few-shot learning (FSL) frameworks. In contrast with Amaral's work, we investigate more recent technologies and examine the performance of the alternative solutions in particular when training data is scarce. Our alternative solutions aim at checking whether the content of a given DPA is complete according to the GDPR provisions. We examine the above technologies side-by-side to assess how they fare in a challenging domain such as the legal domain, focusing on a particular use case; i.e., completeness checking of DPAs against GDPR. Our work is a prerequisite for eliciting software requirements from DPAs. Our comparative analysis aims to reveal the capabilities of the different solutions in analyzing legal text and understanding the domain terminology. Our analysis will further shed the light on data scarcity, an important challenge to consider in general when developing Artificial Intelligence (AI)-enabled automation [62] and in particular in the regulatory compliance context. Our findings provide researchers and practitioners with insights about how technologies compare against one another and help them select the most suitable technology for a new use case.

Contributions. Concretely, our paper makes the following contributions:

(1) We formally define the completeness checking as a text classification problem. Given a set of GDPR provisions and a set of DPA sentences, completeness checking can be achieved by classifying the sentences as relevant or not relevant to each provision. This text classification task can be achieved via binary classification (e.g., "provision x" versus "not provision x") as well as multi-class classification where each provision is considered as one class. We elaborate the problem definition in Section 3.1.

(2) We propose ten alternative solutions for checking the completeness of DPAs against GDPR. Our solutions design is driven by two main objectives. The first objective is to compare the performance of different enabling technologies in checking the completeness of DPAs. We achieve this objective through running extensive experiments in different settings to draw conclusions. The second objective is to experiment with FSL solutions which require a much smaller labeled dataset compared to conventional solutions. To this end, we propose four solutions that utilize recent LLMs, five solutions based on traditional ML, and one that uses FSL framework. We describe our solutions design in Section 3.2.

(3) We empirically evaluate the alternative solutions on 169 real DPAs. To do that, we used the 54 DPAs presented by Amaral *et al.* [19] and further curated a dataset of 113 DPAs covering various sectors including cloud services, data analytics, and accounting audits. The curatation of the dataset was performed by three annotators who had legal background. The annotators collectively analyzed 31,185 sentences, of which only 3,387 sentences were labeled as satisfying at least one provision in GDPR. We provide more details about the curation process in Section 4.3. On a subset of 30 DPAs that we used exclusively for evaluating the alternative solutions, we observe that LLMs-based solutions yielded the best performance with BERT and RoBERTA in the lead. BERT outperformed the rest of the solutions in solving the binary classification problem with a precision and recall of 75.1% and 90.1%, respectively. RoBERTa achieved the best performance in solving the multi-class classification problem with a precision and recall of 69.8% and 96.6%.

Our evaluation further shows that using FSL yields better precision on the cost of a significant loss in recall. Compared to BERT on the binary task, the FSL-based solution has an average gain in precision of ≈3 percentage points (pp) and an average loss in recall of ≈10 pp. Compared to RoBERTa on the multi-class classification task, the FSL-based solution achieves ≈11 pp more precision and ≈17 pp less recall. Given that FSL is developed on a small proportion of the dataset (30% in our case), the above results show that FSL can be useful in some contexts particularly when training data is scarce or when specific hardware (e.g., GPU) is not available to train more complex solutions. We report on our empirical evaluation in Section 4.

(4) We analyze the impact of the dataset imbalance and size, two major characteristics that can significantly affect the performance of the alternative solutions. Specifically, we experiment with various methods, including random sampling and data augmentation to improve the distribution of the underrepresented examples in our dataset. In brief, our experiments indicate that random oversampling performs better than data augmentation techniques with an average gain of about 4 pp in accuracy. We also show that the best strategy to balance the dataset is to use a combination of techniques to increase the minority class and simultaneously decrease the majority class. This strategy has been already applied in the RE literature [58]. Further details are given in Section 4.6.

Structure. The remainder of the paper is structured as follows: Section 2 explains the necessary background for our alternative solutions and further reviews related work. Section 3 defines the completeness checking problem and discusses our solutions design. Section 4 presents the details of our empirical evaluation and reports on the performance of the alternative solutions against one another and further provides insights on the impact of the size and imbalance in the dataset on their performance. Section 6 discusses threats to validity. Finally, Section 7 concludes the paper.

2 Background and Related Work

In this section, we discuss the necessary background and further review the existing literature relevant to DPA completeness checking.

2.1 Background

Below, we briefly present GDPR, and then explain the different enabling technologies underlying our proposed solutions.

GDPR. The GDPR [25] is the European benchmark for data protection and privacy standards. According to GDPR, an organization can be a data controller or data processor. The controller decides the purpose and mechanism of collecting and processing personal data, whereas the processor processes personal data on behalf of the data controller. In our work, we focus on data processing agreements (DPAs). A DPA is a legally-binding contract between a data controller and data processor to ensure the protection of personal data throughout the data processing chain. A DPA sets out the rights and obligations of the controller and processor. To be deemed complete, the DPA's content should comply with the provisions of Article 28 in GDPR.

Large Language Models (LLMs). LLMs, which have dominated the current state-of-the-art in NLP, are deep learning models that are pre-trained with a huge amount of (textual) data with the main objective to predict the next word in a text sequence. The pre-trained models obtain a basis knowledge about the languages to which they are exposed. These models can then be used to solve NLP downstream tasks such as text classification and question answering. There are two strategies for using the pre-trained LLMs, namely *fine-tuning* the pre-trained models with task-specific labeled datasets, and *extracting dense representations* (also called embeddings) that describe text sequences in the downstream task and then use these embeddings as learning features in an ML-based solution. Below, we briefly introduce the LLMs that we apply in our work.

(1) *Bidirectional Encoder Representations from Transformers (BERT) [23]* is one of the early LLMs which is still applied in the NLP literature due to its robust performance. BERT is pre-trained using two tasks: (1) predicting a randomly masked word in a text sequence by learning about its surrounding

context to the left and right, and (2) predicting whether two sentences are consecutive. BERT is built using the Transformer architecture which is composed of a multi-layer encoder-decoder structure, wherein the encoder maps an input text sequence into numerical vector representations and the decoder generates an output sequence. BERT has 110 million parameters, and is pre-trained on 16GB of text corresponding to 3.3 billion words from the BooksCorpus [73] and the entire English Wikipedia.

(2) *A Lite BERT (ALBERT) [42]* is a light variant of BERT. While ALBERT uses the same vocabulary size as BERT, it applies, unlike BERT, reduction techniques to reduce the number of parameters to a total of 12 million (instead of 110 million), increasing thereby the efficiency of the pre-training phase. ALBERT achieves comparable performance to BERT on different NLP benchmarks.

(3) *Legal-BERT [20]* is another variant of BERT that is pre-trained on 12GB of a legal text corpus crawled from publicly available web portals. The corpus contains the EU as well as the United Kingdom legislations, cases from the European Court of Justice and the European Court of Human Rights, and court cases and contracts from the United States (US). Legal-BERT outperforms BERT on two NLP tasks specific to the legal domain, namely multi-label text classification of EU laws and named entity recognition on US contracts.

(4) *Robustly Optimized BERT pre-training Approach (RoBERTa) [44]* is a variant of BERT with the main difference that it is pre-trained with a much larger dataset containing 161GB of text coming from CC-NEWS [51], openWebText [32], and Stories dataset [73]. Additional differences compared to BERT include: (i) RoBERTa is trained only on one task, namely predicting the masked words; (ii) masking the words is done dynamically where different words are masked in different epochs; and (iii) the pre-training is performed for longer time on longer text sequences.

(5) *Sentence-BERT* (SBERT) is a variant of BERT that is trained on large datasets composed of sentences pairs from well-known NL benchmarks [59]. SBERT is trained using a Siamese network architecture which enables the model to derive more meaningful semantic representations of entire sentences instead of merely single tokens. SBERT is the basis for the current state-of-the-art on LLMs for generating sentences embeddings.

In our work, we consider the above-explained strategies for utilizing LLMs. Consequently, we fine-tune the first four LLMs to perform the completeness checking of DPAs, whereas we use SBERT only for extracting the embeddings which then serve as the learning features in our ML-based alternative solutions.

Few-shot Learning (FSL). With the rise of pre-trained LLMs, it became possible to train text classifiers using few examples [57, 61]. FSL leverages the extensive knowledge that these pre-trained models have obtained being exposed to a huge body of unlabeled text. The few labeled examples are then used to teach LLMs about the specific classification task. The idea has been investigated in the RE literature for classifying textual requirements [4]. More recently, Tunstall *et al.* [69] propose SetFit, a framework for few-shot fine-

tuning of sentence transformers. This framework is composed of two phases. In the first phase, it builds on SBERT model to learn sentences representations within the scope of the specific task, and for that it uses a handful of labeled examples. In the second phase, the sentences representations along with their labels are fed in to a classification head, resulting in a text classifier. This framework has shown promising results [11,34,35,3,41]. In our work, we experiment with both ideas: fine-tuning pre-trained LLMs on a small set of labeled data as well as using SetFit framework. Leveraging the massive knowledge obtained by pre-trained LLMs for solving a more domain-specific task is a form of *transfer learning* [39]. Transfer learning is often used when the number of available training examples is not adequate for training classifiers from scratch. In our work, we investigate the capability of automated solutions developed based on LLMs compared to those based on ML which are trained from scratch.

Machine Learning (ML). In our work, we use supervised ML algorithms for performing text classification. Being exposed to manually-labeled datasets of input examples mapped to a set of predefined output labels (or classes), such algorithms can learn to predict the most likely output class for a new, previously unseen input example. For checking the completeness of DPAs (the focus of our paper), an input example represents a text sequence in a given DPA and the output classes are one or more provisions in GDPR with which the DPA text is complying. Positive examples are those text sequences that satisfy the GDPR provisions, and negative examples are those that do not. As a pre-requisite for building ML classifiers, the input text must be converted to numerical representations. Current state-of-the-art represent text using embeddings, which are dense vector representations generated by LLMs. In our work, we represent the text using sentence embeddings from SBERT (described above). For our comparative analysis, we selected three well-known traditional ML algorithms, including *logistic regression (LR)*, *random forest (RF)*, *support vector machine (SVM)*, and two deep learning (DL) algorithms, namely *a multilayer perceptron (MLP)* —a feedforward neural network, and *bidirectional long short-term memory (BiLSTM)* —a recurrent neural network. BiLSTM has been widely applied in the NLP literature prior to LLMs.

Dataset Handling. One of the major challenges encountered when developing automated solutions based on supervised learning, our work not being an exception, is the imbalance of the labeled dataset. In our dataset, the positive examples are significantly under-represented. To improve the distribution of the positive examples, we experiment with data augmentation techniques. To handle data imbalance, we further experiment with random sampling. Below, we introduce the techniques applied in our work.

(1) *Data Augmentation:* Data augmentation is used to automatically generate more training data (positive examples in our case). Data augmentation methods can be broadly categorized into [43]: (i) *paraphrasing-based methods* where the original training examples are reproduced using semantically similar text, and (ii) *noise-based methods* where the original examples are altered

without affecting the meaning. In our evaluation, we use three paraphrasing-based methods, including *back translation (BT)*, *synonym replacement (SR)*, and *embeddings replacement (ER)*. In BT, the original text in the training example is first translated from its language (English in our case) to other languages (e.g., French or German) and is then translated back to the original language, obtaining thereby a paraphrased version of the original text. In SR, one or more words in the training example are replaced by their synonyms using some external knowledge resource such as WordNet [49,27]. In ER, randomly-selected words in the original training example are replaced with the other words which have the most similar embeddings in the vector space as generated by some pre-trained LLMs. In our work, we apply this method using the embeddings of three LLMs, namely ALBERT, BERT, and RoBERTa.

We further use a *noise injection (NI)* method which introduces random noise to the text in four ways [43], including *swapping* which swaps two randomly-selected words in a text sequence, *deletion* which deletes random words in a text sequence, *substitution* which substitutes random words in a text sequence, and *cropping* which cuts off a random consecutive set of words in a text sequence.

(2) *Data Imbalance Handling:* To obtain a more balanced dataset, random sampling can be used [16]. In our evaluation, we apply (i) *random undersampling (RU)* where instances from the majority classes are randomly removed, (ii) *random oversampling (RO)*, where instances for the minority classes are duplicated, and (iii) a mix of RU and RO where both the majority class is reduced and simultaneously the minority class is increased.

2.2 Related Work

Completeness in RE has been extensively studied as part of quality assurance of requirements. A recent mapping study shows that requirements completeness is the second top most investigated quality attribute after ambiguity [50]. Most of the existing approaches rely on external knowledge resources for detecting missing requirements. Ferrari *et al.* [28] uses NLP methods to identify missing concepts and relations in textual requirements according to available documents created during requirements elicitation phase (e.g., customer-meeting transcript). Arora *et al.* [8] verifies the completeness of a set of natural-language requirements against a domain model that is created a priori. More recently, Luitel *et al.* [46] propose leveraging LLMs such as BERT as an alternative external knowledge resource to identify incompleteness violations in requirements. In the context of regulatory compliance, the external resource is the regulation or an abstract representation thereof. Much work has been done so far for eliciting legal requirements, creating abstract representations and facilitating the navigation through regulations mainly to enable developing compliance software systems and/or checking the compliance of software against applicable laws. Existing work relies primarily on modeling the regulation [68,

56, 72, 64, 36, 52]. There are also attempts to retrieve compliance-relevant information by querying the regulation [1, 63]. Dealing with data scarcity is another relevant research topic. Gebauer *et al.* [30] propose integrating ML into the manual annotation process in privacy policies, focusing on data subject rights. The authors employ LLMs to devise a classifier that predicts based on previously seen training examples whether unseen text is about data subject rights. The predictions are leveraged to help human annotators annotate unseen privacy policies more efficiently.

Another research strand in RE focuses on the completeness of regulated documents against applicable regulations. The intuition behind is that regulated documents are often more focused on certain aspects and are thus easier to navigate and elicit legal requirements. For example, DPAs set out the obligations of the processor. Bhatia *et al.* [14] develop an automated approach for detecting the incompleteness violations in privacy policies against the privacy regulations. Their approach applies semantic frames where a sentence is analyzed according to what semantic roles certain verbs expect. For instance, the verb "buy" expects the roles of an actor (i.e., the buyer), something bought, and a price. This decomposition enables detecting incompleteness according to what roles are missing in the sentence. Torre *et al.* [66] and Amaral *et al.* [7] propose using a combination of NLP and ML technologies to detect the incompleteness violations in privacy policies. Specifically, they classify the textual content of a privacy policy according to a comprehensive conceptual model consisting of semantic metadata manually-defined based on relevant GDPR provisions.

The closest to our work is the one by Amaral *et al.* [19] who propose using semantic frames combined with rules to check the completeness of DPAs against GDPR. In collaboration with legal experts, they first define a set of 45 compliance criteria that are derived from GDPR provisions concerning DPAs. These criteria are documented as "shall"-requirements to be better understood by requirements engineers. Out of the total 45 criteria, 26 are mandatory, meaning that the DPA's content must satisfy these criteria, otherwise the DPA is deemed non-compliant. The mandatory criteria are further broken down into three categories, namely metadata, processor obligations, and controller rights. Using rules on top of semantic frames, they develop then an automated approach to verify the content of DPAs according to these criteria. In a follow-up, the authors propose leveraging ML and conceptual modeling for checking the completeness of DPAs.

Existing work in RE has two limitations. First, the research focuses on privacy notices and gives little attention to compliance checking of DPAs. DPAs describe different requirements related to privacy and data protection [5]. Second, the existing work on DPAs does not exploit the recent advancements of NLP, notably LLMs, which have recently shifted the NLP landscape. In our work, we aim to address these limitations.

Similar to the existing literature in RE, our work focuses on verifying the completeness of a regulated document (a DPA, in our case) against the applicable regulations (the provisions of GDPR). In contrast with the existing

literature, we (1) conduct a comparative analysis of ten alternative solutions covering different enabling technologies, and further (2) provide insights about the impact of the size and imbalance of the training data on these solutions. Specifically, our work builds on the legal knowledge concerning the interpretation of GDPR with regard to DPAs completeness as established by Amaral *et al.* [19]. We scope our work to 19 mandatory compliance criteria concerning processor obligation, listed in Table 1. These criteria (labeled **P1** – **P19**) are extracted from the GDPR provisions relevant to DPA compliance and should be considered for creating a GDPR-compliant DPA. The motivation behind our choice is two-fold. First, these criteria are mandatory, i.e., they must be satisfied in the DPA, consequently in the software systems that are employed for data processing. Second, unlike other mandatory criteria which mainly constrain the binding agreement (e.g., its duration), criteria on processor obligation can be translated to actionable points in the software systems which process personal data. For example, the criterion **P6** states that the processor must ensure the security of personal data. This criterion is satisfied in the first provision of our example DPA in Fig. 1. To fulfill this criterion, requirements engineers can use the provision in the DPA specify dedicated functional requirements like the ones labeled `SEC-1` – `SEC4` in Fig. 1.

3 Automated Completeness Checking of DPAs

In this section, we first provide a generic definition of the completeness checking against a regulation as a text classification problem. We then explain the design of our proposed alternative solutions.

3.1 Problem Definition

Definition 1 (Completeness Checking) Let $\mathcal{S} = (s_1, s_2, \ldots, s_n); n \geq 1$ be a list of text span in a given regulated document (denoted as $\mathcal{D}$), where n is the number of sentences. Let $\mathcal{P} = (p_1, p_2, \ldots, p_m); m \geq 1$ be a set of provisions in some regulation (denoted as $\mathcal{R}$) related to the completeness of the regulated document, where m is the number of provisions. Completeness checking is typically a multi-class text classification problem involving classifying the text in a given DPA into the different provisions. Assigning the label p_i to a text span s_j indicates that s_j is relevant to (or satisfies) p_i. This multi-class classification problem can be solved either (1) by building *multiple binary classifiers* (one classifier for each provision) or (2) by building *one multi-class classifier* for all provisions combined. Though one solution proposal would suffice to address the completeness checking problem, our work is intended to provide a comprehensive overview of possible alternatives that one can pursue. We define in the following each solution proposal:

1. *Binary classification:* For each provision $p_i; 1 \leq i \leq m$, we build a corresponding binary classifier $c_i; 1 \leq i \leq m$ which can predict for a given text

Table 1 GDPR Provisions concerning Mandatory Processor's Obligation in DPAs [19].

ID	Provision
P1	The processor shall not engage a sub-processor without a prior specific or general written authorization of the controller.
P2	In case of general written authorization, the processor shall inform the controller of any intended changes concerning the addition or replacement of sub-processors.
P3	The processor shall process personal data only on documented instructions from the controller.
P4	If the processor requires by Union or Member State law to process personal data without instructions and law does not prohibit informing the controller on grounds of public interest, the processor shall inform the controller of that legal requirement before processing.
P5	The processor shall ensure that persons authorized to process personal data have committed themselves to confidentiality or are under an appropriate statutory obligation of confidentiality.
P6	The processor shall take all measures required pursuant to Article 32 or to ensure the security of processing.
P7	The processor shall assist the controller in fulfilling its obligation to respond to requests for exercising the data subject's rights.
P8	The processor shall assist the controller in ensuring the security of processing.
P9	The processor shall assist the controller in notifying a personal data breach to the supervisory authority.
P10	The processor shall assist the controller in communicating a personal data breach to the data subject.
P11	The processor shall assist the controller in ensuring compliance with the obligations pursuant to data protection impact assessment.
P12	The processor shall assist the controller in consulting the supervisory authorities prior to processing where the processing would result in a high risk in the absence of measures taken by the controller to mitigate the risk.
P13	The processor shall return or delete all personal data to the controller after the end of the provision of services relating to processing.
P14	The processor shall immediately inform the controller if an instruction infringes the GDPR or other data protection provisions.
P15	The processor shall make available to the controller information necessary to demonstrate compliance with the obligations Article 28 in GDPR.
P16	The processor shall allow for and contribute to audits, including inspections, conducted by the controller or another auditor mandated by the controller.
P17	The processor shall impose the same obligations on the engaged sub-processors by way of contract or other legal act under Union or Member State law.
P18	The processor shall remain fully liable to the controller for the performance of sub-processor's obligations.
P19	When assessing the level of security, the processor shall take into account the risk of accidental or unlawful destruction, loss, alternation, unauthorized disclosure of or access to the personal data transmitted, stored or processed.

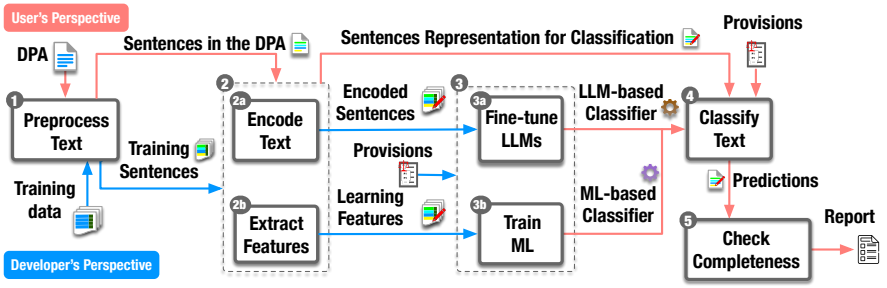

Fig. 2 Overview of our Solution Design.

span $s_j; 1 \leq j \leq n$ whether s_j *satisfies* p_i or *does not satisfy* p_i. Each c_i would predict a single label for the same span. The span that is labeled as *not satisfying* by all classifiers is then deemed not relevant to $\mathcal{P}$. This definition allows multi-labeling, i.e., the same span can satisfy multiple provisions.

2. *Multi-class classification:* For the set of provisions, $\mathcal{P}$, we build one classifier c_p which predicts a label indicating whether a given text span $s_j; 1 \leq j \leq n$ *satisfies* any provision $p_i \in \mathcal{P}$. To enable predicting the cases when s_j is not relevant to $\mathcal{P}$, we add to the set of previsions above the label *other*, i.e., we train the multi-class classifier on $m+1$ labels including $p_1, \ldots, p_m$ and *other*. This definition is restricted to single-labeling, i.e., a span can satisfy exactly one provision (or none) .

Following this, $\mathcal{D}$ is complete according to $\mathcal{R}$ when there is, for each provision p_i, at least one text span s_j in $\mathcal{D}$ which satisfies p_j. Otherwise, $\mathcal{D}$ is considered as incomplete. A provision that is not satisfied in $\mathcal{D}$ leads to an *incompleteness violation*. Recall that all provisions we consider in our work are mandatory, i.e., they must be satisfied by the DPA, irrespective of the application domain of the data controller or processor, to be deemed complete according to GDPR. Additional domain or business relevant text spans not obliged by GDPR can be also present in the DPA. These spans are however not part of our GDPR compliance checking approach.

Applying this definition on our work, $\mathcal{R}$ represents GDPR and $\mathcal{D}$ represents a DPA. We consider 19 provisions from GDPR, i.e., $m = 19$. In our experiments, we build both binary classifiers as well as multi-class classifiers as we discuss later in this section. The overall goal of the alternative solutions in our work is to reveal the incompleteness violations in a given DPA, i.e., the GDPR provisions which the DPA does not satisfy.

3.2 Alternative Solutions Design

In this section we discuss the design of our proposed solutions for checking the completeness of a DPA against GDPR. The solutions can be categorized

into (A) LLM-based which also covers the FSL solution, and (B) ML-based which spans several traditional and recent ML algorithms. The main difference between the two categories is that LLM-based solutions are pre-trained on a massive amount of text, i.e., they have obtained knowledge about the language. We fine-tune the pre-trained LLMs on completeness checking utilizing thereby transfer learning concepts. In contrast, ML-based solutions are directly trained on the specific task at hand without having any prior knowledge. In Section 4, we investigate the benefit of transfer learning. With the intuition to keep our solutions design versatile, i.e., adjustable to more recent alternative algorithms in each category, we describe below the different steps in each category without referring to specific algorithms.

The overview of our solutions is depicted in Fig. 2. Note that the figure distinguishes between the user's and developer's perspectives. The user's perspective assumes that the classifiers used in Step 4 are readily available, whereas the developer's perspective needs to build these classifiers. Step 1 preprocesses the textual content of a single DPA or a training set of several DPAs. Step 2 represents the input text as feature vectors compatible with the intended solution. To do that, Step 2(a) encodes the text, and Step 2(b) extracts a set of feature embeddings following the common practices in the current ML literature. Based on the results of Step 2, Step 3(a) fine-tunes an LLM-based and Step 3(b) trains an ML-based solution over the training dataset. Step 4 predicts whether the DPA's text satisfies the GDPR provisions. Finally, Step 5 post-processes the predictions to conclude the violations in the DPA producing thereby a detailed report as the final output. Below, we elaborate these steps.

3.2.1 Step 1: Preprocess Text

The input to this step can be either a single DPA or a set of several DPAs used for creating the solutions. In both cases, we parse each DPA and preprocess its text using a simple NLP pipeline, composed of the three modules: The first module is a *tokenizer* which splits the text into a set of tokens, e.g., words and punctuation marks. The second module represents a *sentence splitter* which splits the text into sentences according to widely-used endings of a sentence captured in the punctuation marks. Examples of such endings include periods and colons in the English language. We note that the sentence splitter does not necessarily produce grammatically correct sentences. Finally, the third module is a *normalizer* that replaces the specific names and references of the processors and controllers by their generic descriptions, i.e., *PROCESSOR* and *CONTROLLER*. While the first two modules are fully automated, the third module is semi-automated using regular expressions and human in the loop. The reason for semi-automating is due to the multiple ways of referring to the processors and controllers which vary across the DPAs in our dataset. For instance, a processor can be referred to by its name (i.e., ORGANIZATION X) or using different alternative references including *importer*, *service provider*, etc. The intermediary output of this step represents the list of sentences in

the DPA, i.e., $\mathcal{S} = (s_1, s_2, \ldots, s_n); n \geq 1$ where n is the number of sentences in the DPA. This output is passed on to Step 2.

3.2.2 Step 2: Create Feature Vectors

In this step, we loop over the sentences in the input DPA and create for each sentence a corresponding feature vector which will be used to enable building the solutions or classifying the text. Below, we discuss this step considering the two enabling technologies considered in our work.

(a) Encode Text for LLM-based Solutions. LLMs require the input text to be encoded in a specific format. BERT-like models automatically add two special tokens, namely *[CLS]* and *[SEP]*. The [CLS] encapsulates the text representation which is fed into an output layer for classification. The [SEP] token, typically used to split two input text pairs, is not applicable in our case as the input is a single sentence. Given a sentence $s_j \in \mathcal{S}$, where s_j is a sequence of tokens denoted as $t_1, \ldots, t_k; k \geq 1$, this step encodes s_j as $([CLS], t_1, \ldots, t_k[SEP]\phi)$. This encoded text is passed on to Step 3(a) and Step 4 for fine-tuning LLM-based solution or applying readily-available ones to classify the text according to the provisions, respectively.

(b) Extract Features for ML-based Solutions. As discussed in Section 2.1, text classification using ML requires transforming the text into mathematical representations which are then used as learning features. In our work, we use the most recent representation methods to generate sentences embeddings (denoted as e). A sentence s_j will be represented by a sequence of numbers corresponding to its embeddings $e_1, \ldots, e_{768}$. Note that the sentences embeddings are 768-dimensional since we apply SBERT (a variant of BERT) to generate these embeddings. Similar to Step 2(a), the intermediary output of this step is passed on to Step 3(b) to train the ML classifiers or Step 4 to classify text.

3.2.3 Step 3: Build Text Classifiers

This step is applied only in the developer's perspective. The goal is to create different classification models that can be then used for classifying the text of a given DPA according to whether they satisfy the GDPR provisions. For creating the classifiers, we use a training dataset (elaborated in Section 4.3) which is subjected to the first two steps, namely preprocessing and feature extraction. Referring to the problem definition described earlier, we build binary and multi-class classifiers as follows. For a particular provision p_i, a corresponding binary classifier is created by being exposed to positive examples, i.e., all sentences in the training set that satisfy p_i, and negative examples, i.e., all sentences in the training set (or a subset thereof) that do not satisfy p_i. As a result, we created a total of 19 binary classifiers corresponding to the GDPR provisions considered in our work. We further create one multi-class classifier that is exposed to all sentences satisfying any GDPR provision.

3.2.4 Step 4: Classify Text

This step takes as input the provisions of GDPR, a classification model (created in Step 3) and the representation of a sentence in the input DPA. Each model then predicts whether the sentence satisfies a provision or not. In a realistic scenario, the user would use only one solution and one problem definition. For instance, the user will apply one multi-class classifier to predict the provision discussed in the given sentence. To help the user decide what is more suitable for their application context, we experiment in Section 4 with multiple alternative solutions and report on their performance.

3.2.5 Step 5: Check Completeness

Finally, Step 5 takes the predictions made for all the sentences in the input DPA and process them to predict GDPR-related completeness violations as follows: A provision p_i is violated in a given DPA if there are no sentences that are predicted to be on p_i in the DPA. Otherwise, if there is at least one sentence on p_i in the DPA, then p_i is satisfied. Any missing p_i leads to an incomplete DPA according to GDPR. The final output of this step is then a detailed report describing the missing provisions as well as the satisfied ones alongside the sentences that actually satisfy them.

4 Empirical Evaluation

In this section, we evaluate our proposed solutions.

4.1 Research Questions

Our empirical evaluation addresses the following research questions (RQs):

RQ1: Which alternative solution is the most accurate for checking the completeness of DPAs against GDPR?
RQ1 thoroughly compares our proposed solutions aiming at determining the most accurate one for completeness checking of DPAs. We analyze the performance of the solutions in solving the completeness checking both as a binary and multi-class classification problem. The best-performing solution is then used to address the subsequent RQs.

RQ2: Which data imbalance handling method yields the best accuracy for DPA completeness checking against GDPR?
As we discuss in Section 4.3, our dataset is highly skewed towards the negative examples (i.e., sentences that do not satisfy any provision) being significantly more than with the positive examples (i.e., DPA sentences that satisfy any GDPR provision). RQ2 investigates several methods that handle data imbalance, and further analyzes the impact of applying these methods on the results of the alternative solutions.

RQ3: How accurate are FSL solutions in checking the completeness of DPAs against GDPR? Scarcity of labeled datasets is a problem that has been often discussed in the RE literature. In the context of regulatory compliance, building labeled datasets often requires the involvement of legal experts which is expensive and not always possible. To address this concern, we apply an FSL framework which utilizes LLMs, yet requires much less training data. In RQ4, we assess using FSL in several scenarios and discuss its performance compared with the other alternative solutions developed over the entire training dataset.

RQ4: What is the execution time of our proposed solutions? Our solutions are intended to assist requirements engineers in identifying incompleteness violations in DPAs which can significantly impact the compliance requirements specified for data processing activities in software. Our solutions can also assist legal experts in their compliance checking activities. To run any solution in a realistic scenario, it must have practical time. RQ4 investigates the execution time of our proposed solutions.

4.2 Implementation Details and Data Availability

We implemented all alternatives solutions using Python 3.8 [33], in PyCharm IDE [37]. For operationalizing the NLP pipeline applied in Step 1, we use the NLTK 3.7 library [15]. In Step 2(b), we extract the feature embeddings from SBERT (concretely, 'all-MiniLM-L6-v2') which is available in the Transformer 4.18.0 library [71]. We use the same Transformer library and PyTorch 1.12.1 [54] for implementing MLP and BiLSTM in addition to all details concerning the LLM-based solutions, including the pre-trained models, the fine-tuning process, and the text encoding. All LLMs used in our study are the pre-trained *BASE* variants, including 'bert-base-uncased', 'nlpaueb/legal-bert-base-uncased', 'roberta-base', and 'albert-base-v2'. We implement the FSL framework using the SetFit 0.6.0 library [69] and sentence transformers 2.2.2 library [59]. Finally, we use the Scikit-learn 1.0.2 library [55] which provides the basic functions for ML, e.g., training ML classifier, dataset imbalance handling and splitting the dataset. We make all our non-proprietary material used in our empirical evaluation publicly available at this link[3].

4.3 Data Collection Procedure

The purpose of our data collection is to collect a large set of DPAs that are manually analyzed and checked for completeness against GDPR. To do so, we extended the original dataset (consisting of 54 DPAs) that is presented by Amaral *et al.* [19] with another 113 DPAs, of which 17 are from diverse public resources. The DPAs in our dataset are contracts between controllers and processors coming from different sectors, such as tax and cloud services,

[3] `https://zenodo.org/records/11047441`

services related to human resources, data analytics, accounting, audits, and pension services.

To build the ground truth in our work, the DPAs were manually analyzed by three third-party annotators (non-authors). All annotators were in the graduate program at law department. Thus, they had the necessary legal background for our task. The annotators further attended a half-day training about compliance and completeness against GDPR. In this training, we introduced the GDPR provisions (in Table 1) concerning processor's obligations. We prepared the annotation process as follows: We first take out 10% of the DPAs to be used as a shared subset on which the interrater agreement is measured. We then split the DPAs into three subsets maintaining to the possible extent nearly equal number of sentences in each subset to balance the load. We provided clear instructions for analyzing the DPAs, including to examine each sentence in the DPA and decide whether the sentence satisfies one or more GDPR provisions. Each annotator was assigned a subset of DPAs to analyze. To ensure that the annotators obtained mutual understanding of the task, we had few feedback sessions where annotators discussed border cases. On a subset of 10% of overlapping DPAs, where each DPA is independently analyzed by at least two annotators, we computed Cohen's Kappa interrater agreement metric (κ) [21]. The average κ was 0.82, implying "almost perfect agreement" [70]. To solve the disagreements, we presented to each annotator the sentences which are disagreed upon by the other annotator on the same set, and asked them to provide feedback on whether they agree to change their annotation accordingly. All remaining disagreements were discussed in an online session.

To mitigate fatigue, the annotators were recommended to work two hours at a time. They declared a total of ≈57 hours each over a span of three months.

Our data collection procedure resulted in analyzing a total of 169 DPAs consisting of 31,185 sentences. Of these sentences, only a small fraction of ≈12% (about 3,387 sentences) was labeled as satisfying any provision from GDPR. The remaining fraction (≈88%) was labeled as *other* as it contained information not directly relevant to the provisions we consider in this work (see Table 1). Example of *other* sentences include the contact details of the controller and processor or the definitions of certain terms in the agreement. These sentences together with their labels are then used as our ground truth. We split the DPAs in our ground truth into two proportions as follows: 70% of the DPAs is used for developing the alternative solutions and the remaining 30% is used for assessing the performance of the solutions. The training and evaluation sets are thereafter referred to as τ and ϵ, respectively. Table 2 provides the statistics about the total number of sentences as well as the total number of DPAs in each set that satisfy each GDPR provision.

Table 2 Results of our Data Collection Procedure.

Provisions	No. of DPAs			No. of Sentences		
	τ	ϵ	$\sum$	τ	ϵ	$\sum$
P1	120	19	139	234	40	274
P2	68	17	85	83	21	104
P3	113	28	141	168	50	218
P4	58	12	70	73	20	93
P5	110	24	134	155	34	189
P6	76	17	93	627	100	727
P7	115	24	139	244	41	281
P8	28	1	29	30	1	31
P9	46	6	52	57	7	64
P10	16	1	17	18	1	19
P11	68	19	87	88	20	108
P12	28	2	30	30	2	32
P13	127	25	152	202	38	240
P14	79	15	94	89	16	105
P15	86	13	99	165	22	187
P16	98	20	118	192	33	225
P17	113	22	135	205	31	236
P18	97	17	114	127	26	153
P19	64	12	76	84	13	97

4.4 Solutions Tuning

We describe below the details of tuning our solutions. The resulting tuned solutions are used to answer our RQs in Section 4.6.

Tuning LLMs. We fine-tune LLM-based solutions to maximize the F_2 score for completeness checking. All LLMs are fine-tuned on the training dataset τ. Each training example is a sentence in the training set alongside the label(s) assigned to it. The sentence is pre-processed and encoded as explained in Section 3.2. For building the binary classifiers, we fine-tune the LLMs on τ for each provision p_i, where the sentences that satisfy p_i are considered as positive examples and all sentences that do not satisfy p_i are considered as negative examples. Following this, we fine-tune ALBERT for 10 epochs with 2e-5 learning rate and 64 batch size, BERT for 20 epochs with 3e-5 learning rate and 64 batch size, Legal-BERT for 20 epochs with 2e-5 learning rate and 32 batch size, and RoBERTa for 20 epochs with 2e-5 learning rate and 64 batch size.

For building the multi-class classifiers, we use the sentences that satisfy any provision as well as the sentences that are labeled as other to properly predict when none of the provisions is satisfied in any sentence. Following this, we fine-tune ALBERT for 10 epochs with 2e-5 learning rate and 32 batch size, BERT for 20 epochs with 3e-5 learning rate and 64 batch size, Legal-BERT for 20 epochs with 2e-5 learning rate and 32 batch size, and RoBERTa for 10 epochs with 2e-5 learning rate and 64 batch size.

Tuning ML. We train and optimize our ML-based solutions on τ. All training examples are represented using embeddings extracted from SBERT as explained in Section 3.2. In total, we apply five algorithms including LR, RF,

SVM, MLP and BiLSTM (see Section 2.1 for more details). All algorithms are tuned with optimal hyper-parameters that maximize the F_2 score. For tuning, we apply grid search [12].

4.5 Evaluation Procedure

To answer the RQs, we conduct the following experiments (EXPI – EXPIV), explained below.

EXPI. EXPI answers RQ1. Specifically, we assess the performance of our proposed solutions both for performing binary and multi-class text classification to check the completeness of a given DPA against GDPR. The solutions are evaluated exclusively on ϵ (our evaluation set). To evaluate the results, we define for a given provision p_j, *true positives* (TPs) as the cases for which p_j is correctly found as satisfied in the DPA, *false positives* (FPs) as the cases when p_j is wrongly found to be satisfied, *false negatives* (FNs) as the cases when a satisfied p_j is missed by the solution (i.e., found as violated), and *true negatives* (TNs) as the cases when p_j is correctly found as not satisfied (i.e., violated) in the DPA. Following this, we apply the traditional metrics accuracy (A), precision (P) and recall (R), computed as A=(TP+TN)/(TP+TN+FP+FN), P=TP/(TP+FP), and R=TP/(TP+FN).

Precision reflects the errors of falsely predicting a satisfied provision (i.e., FPs). FPs entail that the solution presents actual sentences found in the DPAs (which falsely indicate a given provision). Such sentences, if not too many, can be examined and filtered out by the analyst. Recall, on the other hand, reflects the errors of missing satisfied provisions (i.e., FNs). FNs entail that the solution fails to predict actual sentences. Consequently, the analyst would need to review the entire DPA to identify the missing provisions. Reviewing the DPA is a daunting task for the analyst who is typically a requirements engineer often with a limited budget. The objective of our work is to contribute to eliciting more complete legal requirements. In line with this objective (and also with the RE literature [13]), we favor recall over precision. In practice, the solution should have high recall (with reasonable precision) to provide useful assistance and minimize the manual effort required for reviewing DPAs. In light of the above, we compute in EXPI F_β score for $\beta = 2$, indicating that recall is favored over precision. F_2 is computed as $F_2 = ((2^2 + 1) * P * R)/(2^2 * P + R)$.

EXPII. EXPII answers RQ2. Specifically, we investigate which data imbalance handling strategy yields the most accurate solutions. Recall from Section 3.1 that we train 19 binary classifiers, one classifier for each provision $p_j \in \mathcal{P}$. In the training dataset τ, the *positive examples* include all sentences that satisfy p_j in all DPAs in τ. Conversely, the *negative examples* include the sentences that do not satisfy p_j, i.e., the sentences that satisfy any provision but p_j in addition to the sentences that do not satisfy any provision. In this case, the positive examples represent the minority classes, whereas the negative examples represent the majority. Table 2 reports the total number of sentences in τ satisfying each p_j, i.e., positive examples. We further train one

multi-class classifier over the positive examples that satisfy the 19 provisions. In this case, we treat all sentences in τ that do not satisfy any provision as another class referred to as *other*. The rationale behind including *other* is that the overall accuracy of the multi-class classifier in distinguishing provisions improves due its the exposure to the text segments that do not satisfy any provision. The multiple classes in our work include the 19 provisions (minority classes) and *other* (majority class). In EXPII, we apply several data handling strategies aiming to balance τ through (1) increasing the minority classes, (2) decreasing the majority classes, and (3) a combination of both increasing and decreasing. We create 13 variants of τ using random sampling in combination with data augmentation (see Section 2.1 for more details about these methods). Considering the best performing solutions resulted from EXPI, we redevelop these solutions by exposing the classifier each time to a different variant of τ. We then apply the same metrics A, P, R, and F_2 as defined in EXPI to assess the performance of the resulting solutions over ϵ.

EXPIII. This experiment addresses RQ3. Specifically, we test the performance of the alternative solutions when they are developed on a much smaller proportions of the dataset equivalent to 10%, 20%, and 30%. In EXPIII, we also use SetFit framework (explained in Section 2.1) to perform FSL with the same 30% of the training dataset. All the different solutions are again tested exclusively on ϵ. The purpose of EXPIII is to provide insights about how well the solutions fare when there are only few training examples available. To assess the performance we report on A, P, R, and F_2 as defined in EXPI.

EXPIV. This experiment addresses RQ4. We measure the running time of the different solutions taking into account the two perspectives described in Section 3.2. In our work, we use a work station with AMD 12-Core processor (3.70GHz), INVIDIA GeForce RTX 3090 GPU, and 64GB of RAM for training and developing the classifiers, whereas we apply the developed classifiers to test the alternative solutions on ϵ using a normal laptop with 8-Core Intel processor (2.4GHz), and 32GB of RAM.

4.6 Answers to the RQs

RQ1: Which alternative solution is the most accurate for checking the completeness of DPAs against GDPR?

Table 3 shows the accuracy, precision, recall, and F_2 of our proposed alternative solutions. The metrics are computed exclusively over ϵ to measure the capability of the solutions in identifying the satisfied provisions in the DPAs in ϵ. The table shows the results considering the DPA completeness checking both as a binary classification and multi-class classification problem.

With regard to binary classification, BERT achieves the best overall results in terms of accuracy, recall, and F_2. Compared to other LLMs, BERT achieves an average gain in accuracy of 10.5 percentage points (*pp*). BERT further achieves an average gain in accuracy of 11.6 *pp* over all ML-based solutions except BiLSTM which yields the same accuracy as BERT. BERT achieves the

Table 3 Accuracy Results for DPA Completeness Checking Solutions **(RQ1)**.

	Binary Classification				Multi-class Classification			
	A(%)	P(%)	R(%)	F_2(%)	A(%)	P(%)	R(%)	F_2(%)
① ALBERT	69.1	82.4	51.0	55.2	78.1	79.6	77.2	77.7
② BERT	**80.9**	81.4	**81.6**	**81.6**	83.0	82.7	84.7	84.3
③ Legal-BERT	67.5	76.8	53.1	56.6	81.8	79.7	86.7	85.2
④ RoBERTa	74.7	83.7	63.1	66.4	**83.5**	80.5	**89.8**	**87.8**
⑤ BiLSTM	**80.9**	82.5	80.3	80.7	79.3	79.3	81.0	80.6
⑥ LR	59.6	**93.2**	23.5	27.6	66.3	**94.0**	37.1	42.2
⑦ MLP	77.5	85.8	67.7	70.7	78.8	86.5	69.7	72.5
⑧ RF	65.6	87.1	39.1	44.0	63.9	83.0	39.8	44.4
⑨ SVM	74.4	89.4	57.1	61.6	76.0	91.5	58.8	63.4

† The best values of A, P, R, and F_2 are highlighted in **bold**.

best recall with a significant gain of 25.9 *pp* and 34.7 *pp* over LLM and ML-based solutions, respectively. Again, BiLSTM is an exception which performs on par with BERT and has a loss in recall of 1.3 *pp*. In contrast, we observe that ML-based solutions are performing better or on par with LLM-based solutions (in particular, BERT) in terms of precision. As shown in the table, LR yields the best precision value of 93.2%, indicating a gain of 11.8 *pp* over BERT. This is followed by the second best performing SVM with a precision of 89.4%, RF with 87.1%, and MLP with 85.8%. BiLSTM achieves a comparable precision to the one achieved by LLM-based solutions. ML-based solutions (except BiLSTM) have very low recall ranging from 23.5% for LR to 67.7% for MLP which introduces a particularly notable degradation in the respective F_2 scores. Having a low recall indicates that many provisions were missed by these solutions. For instance, LR compared to BERT has an average loss in recall of ≈58 *pp*, entailing the risk of missing a large proportion of the satisfied provisions in ϵ. On the one hand, the 11.8 *pp* in precision (the advantage of LR over BERT) entails a fraction of falsely satisfied provisions introduced by BERT which can be filtered out by an expert with relative ease. The low recall of LR, on the other hand, entails the risk of missing a lot of satisfied provisions, compromising thereby the completeness of the DPA. Therefore, for binary classification, we select BERT as the best performing solution.

With regard to multi-class classification, RoBERTa yields the best results in terms of accuracy, recall, and F_2. Compared to RoBERTa, legal-BERT and BERT achieve comparable performance in terms of F_2 with a loss of 2.6 *pp* and 3.5 *pp*, respectively. As shown in the table, ML-based solutions demonstrate similar behavior to the one observed for binary classification. That is, they achieve high precision values at the cost of low recall. BiLSTM is still an exception that achieves comparable results to LLMs. For multi-class classification, we select RoBERTa as the best performing solution.

The remarkably lower recall in the case of traditional ML-based solutions confirms the complexity of the text classification problem in the context of completeness checking of DPA. Not having any ground knowledge about the language (in contrast with LLM-based solutions) makes the prediction task for

Table 4 Breakdown of F_2 Score per Provision for LLM-based Solutions.

		Binary Classification				Multi-class Classification			
P	$\|\epsilon\|$	①	②	③	④	①	②	③	④
P1	19	89.5	94.7	**95.7**	90.4	86.0	**88.5**	**88.5**	85.1
P2	17	39.5	**77.4**	0.0	0.0	67.1	68.8	76.5	**78.3**
P3	28	0.0	**93.5**	0.0	88.2	88.2	**97.1**	88.2	94.2
P4	12	82.0	88.7	**89.6**	**89.6**	78.1	85.9	88.7	**95.2**
P5	24	96.6	**98.4**	**98.4**	88.2	**95.8**	95.0	95.0	95.0
P6	17	0.0	**61.8**	0.0	0.0	46.0	59.5	**73.0**	71.4
P7	24	84.0	87.5	**90.9**	87.5	80.5	84.0	**97.6**	94.3
P8	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**55.6**
P9	6	0.0	**46.9**	0.0	32.3	57.1	**62.5**	27.8	46.9
P10	1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P11	19	63.2	**85.1**	0.0	81.5	79.8	76.1	**85.1**	**85.1**
P12	2	26.3	**55.6**	50.0	0.0	50.0	45.5	**55.6**	41.7
P13	25	96.0	96.0	92.7	**99.2**	92.7	**99.2**	**99.2**	**99.2**
P14	15	82.3	90.9	**94.9**	0.0	80.0	74.3	**90.9**	**90.9**
P15	13	0.0	0.0	0.0	68.5	32.8	76.9	68.2	**83.3**
P16	20	49.5	63.2	**90.0**	87.6	68.4	94.1	89.1	**91.3**
P17	22	0.0	**95.5**	0.0	95.5	91.7	95.5	91.7	**99.1**
P18	17	79.3	**93.0**	0.0	0.0	84.3	89.3	**94.1**	**94.1**
P19	12	28.3	67.2	**69.2**	0.0	**75.8**	74.6	68.2	68.2

† P: Provision, $|\epsilon|$: The number of DPAs satisfying P in ϵ.
‡ ① ALBERT, ② BERT, ③ Legal-BERT, ④ RoBERTa.

ML-based solutions more challenging. The reason could be due to the commonalities and textual overlaps between the different text segments in a given DPA which satisfy different provisions. BiLSTM, which was widely applied in the NLP literature prior to LLMs, performs exceptionally well. In addition to the fact that BiLSTM is built using recurrent neural networks, it is also effective since it learns the text in a bidirectional manner (similar to BERT). Bidirectional is essential in learning about the context and words co-occurrences in any language. LLM-based solutions obtained language capabilities during the pre-training phase and hence have an advantage over ML-based solutions (including BiLSTM) to better distinguish the DPA text.

Overall we observe different behavior from the same LLM in binary versus multi-class classification. Recall that binary classifiers are created separately for each provision, e.g., the BERT-based model fine-tuned for predicting P1 is different from the one fine-tuned for P2. We believe that this can be the reason why the same model behaves differently. For instance, RoBERTa learned better in our context when it was exposed to all provisions at the same time (i.e., multi-class classification) whereas it learned poorly when presented with separate provisions against their negative examples.

To better understand the behavior of LLMs on predicting the satisfied provisions in a DPA, we provide a breakdown of the F_2 scores per provision in Table 4. The table also shows the number of DPAs that contain the provision in the evaluation set ($|\epsilon|$). For binary classification, the table shows that BERT outperforms the other LLM-based solutions in nine provisions, namely P2, P3, P5, P6, P9, P11, P12, P17, and P18. However, BERT has zero F_2

scores for identifying P8, P10, and P15. The reason for such low performance for P8 and P10 is two-fold. First, our training set has few positive examples of these provisions (see Table 2). Second, they are satisfied in only one DPA (and in one sentence). Missing those instances can easily lead to zero recall which significantly impact the F_2 scores. P15 is about providing to the controller information necessary to demonstrate compliance with Article 28 in GDPR. The sentences satisfying this provision in our dataset are typically convoluted and refer implicitly to P15. Examples of the text satisfying P15 include *"The Processor will make that policy available to controller, along with other information reasonably requested by controller regarding processor's security practices and policies."*, *"to provide Controller upon request and within reasonable time the information necessary to demonstrate compliance with the obligations laid down in this clause 5;"*, *"and allow the Controller to audit that compliance,"* . We see that the text does not always explicitly refer to compliance with Article 28 or demonstrating compliance. This implicit reference to P15 explains the low performance for identifying P15 by BERT.

The table further shows that Legal-BERT achieves good performance, and outperforms the other solutions in six provisions, namely P1, P4, P7, P14, P16, and P19. Given that we build different classifiers for identifying each provision, it is possible to create a hybrid classifier that combines the best performing classifiers based on BERT and Legal-BERT. The hybrid classifier will use BERT for identifying P2, P3, P5, P6, P9, P11, P12, P17, and P18, and Legal-BERT for identifying the remaining provisions. This hybrid classifier can slightly improve F_2 to 84.1%, a gain of 2.5 *pp* over BERT.

Unlike binary classifiers, the multi-class classifiers are developed on the entire set of provisions, i.e., we have one classifier per LLM-based solution. We observe more variation in the behavior of the best performing solutions than for binary classification. RoBERTa still performs the best in 11 out 19 provisions. RoBERTa scores zero F_2 for identifying P10 for the same reasons highlighted above. Legal-BERT outperforms RoBERTA in three provisions (P1, P6, and P7) and scores equally good for five provisions (P11-P14 and P18). BERT outperforms the rest in two provisions (P3 and P9) and scores equally good in another two provisions (P1 and P13). Despite the overall worse performance, ALBERT still outperforms the rest in two provisions, namely P5 and P19. However, the average gain is not significant compared to the performance of the other models for P5 and BERT for P19. The consistent good performance of RoBERTa confirms our decision to select it as the most accurate solution for multi-class classification.

The answer to RQ1 is LLM-based solutions outperform ML-based solutions in checking the completeness of DPAs against GDPR. When the completeness checking is formulated as a binary text classification problem, the most accurate solution is based on BERT with an average F_2 of 81.6%. Alternatively, RoBERTa is the most accurate solution, yielding an F_2 of 87.8%, when formulating the problem as a multi-class classification.

Table 5 Accuracy Results for Data Handling Strategies **(RQ2)**.

Dataset†	Binary Classification (②)				Multi-class Classification (④)			
	A(%)	P(%)	R(%)	F_2(%)	A(%)	P(%)	R(%)	F_2(%)
τ (RQ1)	**80.9**	**81.4**	81.6	81.6	**83.5**	80.5	89.8	87.8
	Increasing minority classes							
τ+*RO*	79.5	75.1	90.1	**86.7**	79.1	74.2	91.2	87.2
τ+*BT*	79.8	81.0	79.6	79.9	81.4	79.7	85.7	84.5
τ+*ER*	79.6	77.1	86.1	84.1	79.8	79.2	82.7	81.9
τ+*NI*	78.6	74.7	88.4	85.3	80.9	80.9	82.3	82.0
τ+*SR*	80.4	79.7	83.0	82.3	79.1	80.7	78.2	78.7
τ+*All*	79.1	76.8	85.4	83.5	79.6	76.3	87.8	85.2
	Decreasing majority classes							
τ+*RU*	54.7	53.3	**99.7**	84.9	76.7	69.8	**96.6**	**89.7**
	Increasing minority classes and decreasing majority classes							
τ+*RO*+*RU*	63.0	58.4	98.3	86.5	82.5	**85.0**	81.7	82.3
τ+*BT*+*RU*	58.1	55.2	99.3	85.6	77.5	71.3	94.6	88.8
τ+*ER*+*RU*	60.7	56.9	98.3	85.8	79.3	79.7	80.3	80.2
τ+*NI*+*RU*	60.7	56.9	98.3	85.8	78.4	75.7	85.7	83.5
τ+*SR*+*RU*	59.3	56.0	99.0	85.8	80.4	75.4	91.8	88.0
τ+*All*+*RU*	61.6	57.4	99.0	86.5	78.4	74.9	87.4	84.6

† τ: Original training dataset, *RO*: Random oversampling, *RU*: Random undersampling, *BT*: Back translation, *NI*: Noise injection, *ER*: Embeddings replacement, *SR*: Synonym replacement, *All*: All data augmentation methods, namely *BT*, *NI*, *ER*, *SR*.

RQ2: Which data imbalance handling method yields the best accuracy for DPA completeness checking against GDPR?

Table 5 compares the accuracy results of the best performing solutions from RQ1, i.e., ② BERT for binary classification and ④ RoBERTA for multi-class classification, when fine-tuned on 13 variants of our original training dataset. As explained in Section 4.5, these variants are generated through a combination of random sampling (*RO* and *RU*) and data augmentation methods (*BT*, *ER*, *NI*, *SR*).

For binary classification, increasing the minority classes through random oversampling significantly improves the recall with a gain of 8.5 *pp*, consequently F_2 with an gain of 5.1 *pp*. All data augmentation methods also yield a better recall and F_2 than those achieved on the original dataset. The exception to this is *BT* which applies back translation techniques to triple the minority examples (in this case, the sentences that satisfy any provision). We believe that the low performance of *BT* is because it (unlike other data augmentation methods) adds completely new sentences which might have lost the legal domain specificity due to translation. For instance, sentences that are written using "shall" modal verb to express a processor's obligation might be translated back to sentences without any modal verb, in which case the legal context concerning an obligation is no longer present. While increasing the minority classes did not improve accuracy and precision, the achieved values

compared to the ones on the original dataset come with a loss ranging from -1.4 *pp* to -0.5 *pp* in accuracy and -6.3 *pp* to -0.4 *pp* in precision.

Decreasing the majority classes alone or in combination with increasing the minority classes yields a nearly perfect recall for binarcy classification with a notable improvement of ≈18 *pp* over fine-tuning on the original dataset. However, this comes at the cost of huge decrease in precision of about ≈28 *pp*. One reason is that minority classes for some provisions are very few and thus reducing the majority class would result in an overall smaller training dataset that is not informative enough to the binary classifiers. Combining the undersampling with increasing the minority classes does not help much in improving the precision. We conclude that the negative examples are particularly important to teach the classifier more distinguishable patterns to correctly predict the provisions in the DPA. This can be justified through our observation that the DPA text that satisfies any provision is highly similar to the text that does not satisfy any provision. Exposing both texts to the classifier is therefore helpful.

Unlike binary classification, decreasing the majority class for multi-class classification significantly improves recall and F_2 with a gain of ≈7 *pp* and ≈2 *pp*, respectively. Compared to binary classifiers which are fine-tuned for each provision, the multi-class classifier is exposed to all provisions as well as the other sentences that do not satisfy any provision. For this reason, reducing the majority class (*other*, in this case) would have a positive effect. Nonetheless, precision has a loss of ≈12 *pp*. We believe that this loss is still acceptable compared to the benefit of achieving a higher recall. Increasing the minority classes also improves the recall and F_2. In contrary to their behavior for binary classification, data augmentation methods that work at the word level (e.g., SR and ER) performed much worse than BT for the multi-class classification.

We note that in our context data augmentation did not perform exceptionally well compared with random sampling techniques which can be implemented more efficiently. Therefore, we conclude that augmenting textual data is not very well suited for the legal domain due to the complexity of the legal language and the sensitivity of the concepts, terms, and words co-occurrences that can be disrupted through data augmentation methods.

In view of the above analysis, **the answer to RQ2 is** that handling the imbalance of the training dataset using RO yields the best performing binary classifiers with an average F_2 of 86.7% and applying RU yields the best performing multi-class classifier with an F_2 of 89.7%.

RQ3: How accurate are FSL solutions in checking the completeness of DPAs against GDPR? To show the advantage of FSL in our context, we examine the performance of our proposed solutions when fine-tuned (or trained) on a small proportion of our training set (τ). Specifically, we consider the proportions 10%, 20%, and 30% of τ, corresponding to a total of ≈2413 sentences, ≈4828 sentences, and ≈7200 sentences, respectively. We do not experiment with proportions larger than 30% as these would correspond to relatively large datasets (>7200 sentences) and would therefore defeat the

Table 6 Accuracy Results for FSL-based Solutions fine-tuned on a proportion of the training data (%τ) and tested on ϵ **(RQ3)**.

		Binary Classification				Multi-class Classification			
	%τ	A(%)	P(%)	R(%)	F_2(%)	A(%)	P(%)	R(%)	F_2(%)
RQ1	100	80.9	81.4	81.6	81.6	83.5	80.5	89.8	87.8
	FSL scenario 1: Fine-tuning LLMs on %τ, testing on ϵ								
① ALBERT	30	54.4	68.9	21.1	24.5	75.3	82.3	66.3	69.0
	20	61.9	88.9	29.9	34.5	66.5	86.0	41.8	46.6
	10	54.3	91.9	15.3	18.4	62.9	86.0	33.6	38.3
② BERT	30	78.8	78.2	81.6	**80.9**	77.5	77.9	78.9	**78.7**
	20	77.7	82.5	72.1	74.0	77.9	80.7	75.2	76.2
	10	73.4	78.8	68.5	70.3	75.4	82.1	67.0	69.6
③ Legal-BERT	30	67.5	82.6	46.9	51.4	78.6	82.3	74.5	75.9
	20	62.5	74.7	41.2	45.2	79.3	81.2	77.9	78.5
	10	59.1	79.3	31.1	35.4	63.0	77.9	39.5	43.8
④ RoBERTa	30	67.0	79.8	48.3	52.4	79.8	82.5	77.2	78.2
	20	63.7	72.8	47.3	50.8	59.3	77.2	29.9	34.1
	10	55.5	87.5	18.9	22.4	55.6	78.1	19.4	22.8
	FSL scenario 2: training ML on %τ, testing on ϵ								
⑤ BiLSTM	30	78.8	80.1	78.2	78.6	78.2	78.7	68.0	69.9
	20	73.9	74.6	74.8	74.8	73.3	79.3	65.3	67.7
	10	72.5	89.1	53.1	57.7	70.9	79.1	59.2	62.3
⑥ LR	30	48.4	0.0	0.0	0.0	51.8	95.2	6.8	8.4
	20	48.4	0.0	0.0	0.0	49.1	83.3	1.7	2.1
	10	48.4	0.0	0.0	0.0	48.4	0.0	0.0	0.0
⑦ MLP	30	69.1	86.6	52.9	57.4	71.5	89.3	45.6	50.5
	20	67.2	86.9	42.9	47.7	67.9	91.1	41.8	46.9
	10	63.2	92.0	31.3	36.1	52.1	81.8	9.2	11.2
⑧ RF	30	58.2	82.4	28.6	32.9	60.0	80.4	25.2	29.2
	20	59.3	86.0	25.2	29.3	58.8	83.1	25.2	29.2
	10	57.7	88.2	20.7	24.5	57.2	85.7	20.4	24.1
⑨ SVM	30	61.9	91.0	27.6	32.0	61.2	89.7	29.6	34.2
	20	57.0	94.5	17.7	21.1	58.9	91.7	22.4	26.4
	10	53.2	93.5	9.9	12.0	54.0	92.1	11.9	14.4
	FSL scenario 2: Using FSL framework on %τ, testing on ϵ								
⑩ SetFit	30	77.7	77.6	79.9	79.4	79.6	80.5	79.9	80.0
	20	77.7	77.9	79.3	79.0	78.8	81.0	76.9	77.7
	10	76.3	80.9	70.7	72.6	78.1	81.9	73.8	75.3

purpose of using FSL. Considering the different proportions (%τ), we then introduce three FSL scenarios for DPA completeness checking. In the first scenario, we simply fine-tune the LLMs on %τ, in the second scenario, we train the ML classifiers on %τ, and finally in the last scenario, we apply SetFit framework designed specifically for FSL. Table 6 shows the results of the different solutions. For comparison, we provide in the table the results of the best-performing solutions from RQ1, i.e., BERT for binary classification and RoBERTa for multi-class classification.

In the first scenario, the table shows that fine-tuning the LLM-based solutions on on smaller proportions of τ for solving the DPA completeness checking as a binary classification yield much worse performance than fine-tuning them on the entire τ. With 30% of τ, BERT binary classifier manages to achieve an average F_2 of 80.9% (0.7 *pp* less than the F_2 on the entire τ). Compared with binary classification, the table shows that LLM-based multi-class classifiers perform on smaller proportions of τ generally worse than the entire τ. We observe that RoBERTa (the best solution in RQ1) has a loss in F_2 score of ≈9.6 *pp* when fine-tuned only on 30% of τ. This general observation indicate that the multi-class classification task requires considerably larger training sets for the LLMs to effectively learn to distinguish between the classes. BERT performs the best with less data also for multi-class classification, outperforming RoBERTa with 0.5 pp on 30% of τ. We conclude from the above results that BERT has robust performance and generally requires less training data for effective learning compared to other LLMs.

In the second scenario, we notice that, as expected, ML-based classifiers perform bad on smaller proportions of τ. This result is expected since the ML classifiers start learning from scratch compared with the LLMs which are readily pre-trained. BiLSTM is still an exception achieving a comparable results to LLM-based solutions.

In the last scenario, we apply SetFit. As explained in Section 2.1, SetFit leverages LLMs (concretely, sentence transformer) for text classification. The framework attempts first to understand the representations of the training examples (typically, sentences). This is done by fine-tuning the LLM on a few examples from the training data. The fine-tuned model is then used to generate the sentences embeddings that are the main elements for training a classifier tailored to the specific task. Table 6 shows that SetFit provides the best trade-off between the manual effort needed to create labeled examples versus accuracy. Using only 20% of τ, SetFit achieves comparable F_2 values both in binary and multi-class classification compared with fine-tuning BERT on 20% of τ. SetFit achieves an average F_2 of 79% for binary classification (1.9 pp less than that achieved by BERT), and 77.7% for multi-class classification (1 pp less than BERT).

The answer to RQ3 is that BERT demonstrates a robust performance when fine-tuned on a small proportion of the training set, with 30% yielding acceptable F_2. With even less amount of training data, using a dedicated framework like SetFit yields comparable results to BERT.

RQ4: What is the execution time of our proposed solutions? We analyze the running time considering both perspectives highlighted in Fig. 2, namely user's and developer's perspectives. To simply use our proposed alternative solutions, one can apply the readily-developed classifiers and run all the steps depicted in Fig. 2 except Step 3. Following the discussion in the previous RQs, we report below the time required for running four alternative solutions, namely BERT, RoBERTa, SetFit, and BiLSTM. The first two are the best

performing solutions, whereas the latter two demonstrated a comparable performance in terms of accuracy as well as other advantages: SetFit is a perfect alternative when labeled data is scarce while BiLSTM is computationally efficient to develop compared to LLM-based solutions.

Considering the largest DPA in our our evaluation set with about 580 sentences, the first two steps to preprocess text and transform it into an intermediate representation required negligible time for all solutions. Steps 4 and 5 classify the text and check the completeness of the DPA. To do so, BERT and RoBERTa required $\approx$ 5.5 minutes and $\approx$ 19 seconds, respectively. Note that in the case of BERT, we run 19 different classifiers compared to one multi-class classifier in the case of RoBERTa. SetFit required $\approx$ 6 minutes and $\approx$ 19 seconds for respectively solving the binary and multi-class classification problems. Finally, BiLSTM required $\approx$ 34 seconds for solving the binary classification and $\approx$ 1 second for solving the multi-class classification.

The developer must account for the time needed by step 3 to further develop the classifiers. Fine-tuning 19 BERT-based classifiers took about four weeks, whereas fine-tuning RoBERTa required about three days. Note that the fine-tuning time includes the hyper-parameter optimization. SetFit, which is designed to run on a CPU (instead of a GPU), took about 16 hours for developing the 19 binary classifiers and $\approx$ 51 minutes for the multi-class classifiers. In comparison, BiLSTM took about two weeks to train and optimize the the hyper-parameter for binary classification and nearly one day for multi-class classification.

The answer to RQ4 is that the estimated time for analyzing the completeness of a given DPA with 100 sentences ranges between 3 seconds and 1 minute when applying RoBERTa and BERT, respectively. This time is, on the one hand, practical from a user's perspective. Developing similar Solutions, on the other, largely depends on the availability of appropriate hardware (i.e., GPU) and sufficient number of labeled examples. Alternatively, one can develop SetFit, an FSL-based solution, or rely on BiLSTM, compromising thereby the overall accuracy.

5 Discussion

DERECHA is the existing solution proposed by Amaral *et al.* [19] for DPA completeness checking. DERECHA applies manually-crafted rules over the semantic analysis between the text in the DPAs and the provisions in the GDPR. The authors report an overall average precision of 89.1% and recall of 82.4% for detecting the violations of GDPR provisions in DPAs (i.e., detecting missing provisions in DPAs). DERECHA allows for binary and multi-label classification and is thus more comparable to our proposed solution using LLMs as binary classifiers. As mentioned in Section 1, we scope our work to mandatory requirements related to processor's obligations.

For finding the violations of these provisions (P1 – P19, listed in Table 1), DERECHA has an average precision and recall of 81.2% and 70%, respectively. This precision measures the ratio of violations that are falsely introduced by DERECHA due to *missing sentences that actually satisfy the provision*. The recall measures the ratio of missing violations due to *falsely identifying sentences that satisfy the provisions*. In our work, we focus on identifying the sentences that satisfy the provisions (i.e., the present provisions in the DPAs). The best performing model in our results, i.e., BERT, has an average precision and recall of 81.4% and 81.6%, respectively. The precision, in this case, measures ratio of falsely introducing sentences that satisfy a given provision, where as the recall measures the ratio of missing sentences that satisfy a provision. While not directly comparable, we see that BERT has an advantage in identifying sentences that satisfy the provisions.

6 Threats to Validity

This section discusses the validity concerns pertinent to our work.

Internal Validity. The main threat to internal validity concerns bias. To mitigate this threat, we constructed our labeled dataset with the help of three third-party annotators (non-authors). The annotators were not provided any details regarding the development of the different alternative solutions.

External Validity. The main concern in respect of external validity is the generalizability of our proposed solutions. To this end, we note that we conducted our empirical evaluation on a large dataset of real DPAs covering different sectors. We believe that the obtained results reflect real-world conditions since the evaluation set was not exposed during the training or fine-tuning phases. More experimentation and user studies can be however beneficial for improving this validity concern.

7 Conclusion

In this paper, we developed and evaluated ten alternative solutions for checking the completeness of data processing agreements (DPAs) against General Data Protection Regulation (GDPR). The alternative solutions utilize various enabling technologies, including traditional machine learning (ML), deep learning (DL), large-scale language models (LLMs) and few-shot learning (FSL). To evaluate our proposed solutions, we used a dataset of 169 real DPAs curated by three third-party (non-author) annotators. Our results indicate that the LLM-based solutions performed significantly better than ML-based solutions. The best performing solution is BERT with an F_2 score of 86.7%, while enabling multi-label classification, considering that the text in DPAs can fulfill multiple provisions. Our results further highlight the benefit of two alternative solutions, including BiLSTM (a DL algorithm) and SetFit (an FSL framework). BiLSTM performs on par with LLM-based solutions but requires much less

time to develop. SetFit yields less accurate results than LLMs, but it can be developed with much less training data. Overall, the results show that fine-tuning LLMs should be the most immediate option for addressing problems in the legal domain. However, selecting which LLM to choose plays a major role. For instance, though Legal-BERT was pre-trained on legal text, it was out-performed by other LLMS like RoBERTa which was exposed to general yet much larger text body. In the future, we would like to conduct user studies to assess the practical usefulness of these solutions that achieved promising results.

Acknowledgment This paper was supported by Linklaters and Luxembourg's National Research Fund under grant BRIDGES/19/IS/13759068/ARTAGO. This research was further funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant reference NCER22/IS/16570468/NCER-FT. For the purpose of open access, and in fulfillment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission.

Declarations

Data Availability: We make all our non-proprietary material used in our empirical evaluation publicly available at this link

https://figshare.com/s/77338e558ffb6adf6f55.

Conflict of Interest: The authors declared that they have no conflict of interest.

References

1. Abualhaija, S., Arora, C., Sleimi, A., Briand, L.C.: Automated question answering for improved understanding of compliance requirements: A multi-document study. In: 2022 IEEE 30th International Requirements Engineering Conference (RE), pp. 39–50. IEEE (2022)
2. Akhigbe, O., Amyot, D., Richards, G., Lessard, L.: Gorim: a model-driven method for enhancing regulatory intelligence. Software & Systems Modeling **21**(4), 1613–1641 (2022)
3. Al-Kaswan, A., Izadi, M., van Deursen, A.: Stacc: Code comment classification using sentencetransformers. arXiv preprint arXiv:2302.13149 (2023)
4. Alhoshan, W., Ferrari, A., Zhao, L.: Zero-shot learning for requirements classification: An exploratory study. Information and Software Technology **159**, 107202 (2023)
5. Amaral, O., Abualhaija, S., Briand, L.: Ml-based compliance verification of data processing agreements against gdpr. In: 2023 IEEE 31st International Requirements Engineering Conference (RE), pp. 53–64 (2023). DOI 10.1109/RE57278.2023.00015
6. Amaral, O., Abualhaija, S., Sabetzadeh, M., Briand, L.: A model-based conceptualization of requirements for compliance checking of data processing against gdpr. In: 2021 IEEE 29th International Requirements Engineering Conference Workshops. IEEE (2021)
7. Amaral, O., Abualhaija, S., Torre, D., Sabetzadeh, M., Briand, L.: AI-enabled automation for completeness checking of privacy policies. IEEE Transactions on Software Engineering (2021)
8. Arora, C., Sabetzadeh, M., Briand, L.C.: An empirical study on the potential usefulness of domain models for completeness checking of requirements. Empirical Software Engineering **24**, 2509–2539 (2019)

9. Barati, M., Rana, O., Petri, I., Theodorakopoulos, G.: Gdpr compliance verification in internet of things. IEEE access **8**, 119697–119709 (2020)
10. Barati, M., Theodorakopoulos, G., Rana, O.: Automating gdpr compliance verification for cloud-hosted services. In: 2020 International Symposium on Networks, Computers and Communications (ISNCC), pp. 1–6. IEEE (2020)
11. Bashir, S., Abbas, M., Saadatmand, M., Enoiu, E.P., Bohlin, M., Lindberg, P.: Requirement or not, that is the question: A case from the railway industry. In: Requirements Engineering: Foundation for Software Quality: 29th International Working Conference, REFSQ 2023, Barcelona, Spain, April 17–20, 2023, Proceedings, pp. 105–121. Springer (2023)
12. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(1) (2012)
13. Berry, D.M.: Evaluation of tools for hairy requirements and software engineering tasks. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 284–291 (2017). DOI 10.1109/REW.2017.25
14. Bhatia, J., Evans, M.C., Breaux, T.D.: Identifying incompleteness in privacy policy goals using semantic frames. Requirements Engineering **24**(3) (2019)
15. Bird, S.: Nltk: the natural language toolkit. In: Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, pp. 69–72 (2006)
16. Branco, P., Torgo, L., Ribeiro, R.P.: A survey of predictive modeling on imbalanced domains. ACM Computing Surveys (CSUR) **49**(2), 1–50 (2016)
17. Breaux, T.D., Norton, T.: Legal accountability as software quality: A us data processing perspective. In: 30th IEEE International Requirements Engineering Conference (2022)
18. Breitbarth, P.: The impact of gdpr one year on. Network Security **2019**(7), 11–13 (2019)
19. Cejas, O.A., Azeem, M.I., Abualhaija, S., Briand, L.C.: NLP-based automated compliance checking of data processing agreements against gdpr. IEEE Transactions on Software Engineering **49**(9), 4282–4303 (2023)
20. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., Androutsopoulos, I.: Legal-bert: The muppets straight out of law school. ArXiv **abs/2010.02559** (2020)
21. Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement **20**(1), 37–46 (1960)
22. Dalpiaz, F., Van der Schalk, I., Lucassen, G.: Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and nlp. In: Requirements Engineering: Foundation for Software Quality: 24th International Working Conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings 24, pp. 119–135. Springer (2018)
23. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
24. European Union: The GDPR: New opportunities, new obligations. Justice and Consumers (2018)
25. European Union: General data protection regulation. Official Journal of the European Union (2018)
26. Feal, A., Gamba, J., Tapiador, J., Wijesekera, P., Reardon, J., Egelman, S., Vallina-Rodriguez, N.: Don't accept candy from strangers: An analysis of third-party mobile sdks. Data Protection and Privacy, Volume 13: Data Protection and Artificial Intelligence **13**, 1 (2021)
27. Fellbaum, C.: WordNet: An Electronic Lexical Database, 1st edn. The MIT Press (1998)
28. Ferrari, A., dell'Orletta, F., Spagnolo, G.O., Gnesi, S.: Measuring and improving the completeness of natural language requirements. In: Requirements Engineering: Foundation for Software Quality: 20th International Working Conference, REFSQ 2014, Essen, Germany, April 7-10, 2014. Proceedings 20, pp. 23–38. Springer (2014)
29. Freitas, M.d.C., Mira da Silva, M.: Gdpr compliance in smes: There is much to be done. Journal of Information Systems Engineering & Management **3**(4), 30 (2018)
30. Gebauer, M., Mashhur, F., Leschke, N., Grünewald, E., Pallas, F.: A human-in-the-loop approach for information extraction from privacy policies under data scarcity. arXiv preprint arXiv:2305.15006 (2023)
31. Ghanavati, S., Rifaut, A., Dubois, E., Amyot, D.: Goal-oriented compliance with multiple regulations. In: Proceedings of 22nd IEEE International Conference on Requirements Engineering (2014)

32. Gokaslan, A., Cohen, V., Pavlick, E., Tellex, S.: Openwebtext corpus (2019). URL http://Skylion007. github. io/OpenWebTextCorpus (2019)
33. Guido, V.R., Drake Jr, F.: Python 3 reference manual. Scotts Valley: CreateSpace (2009)
34. Halterman, A.: Synthetically generated text for supervised text analysis. arXiv preprint arXiv:2303.16028 (2023)
35. Halterman, A., Schrodt, P.A., Beger, A., Bagozzi, B.E., Scarborough, G.I.: Creating custom event data without dictionaries: A bag-of-tricks. arXiv preprint arXiv:2304.01331 (2023)
36. Ingolfo, S., Siena, A., Mylopoulos, J.: Nòmos 3: Reasoning about regulatory compliance of requirements. In: Proceedings of 22nd IEEE International Requirements Engineering Conference (2014)
37. Islam, Q.N.: Mastering PyCharm. Packt Publishing Ltd (2015)
38. Jain, C., Anish, P.R., Ghaisas, S.: Automated identification of security and privacy requirements from software engineering contracts. In: 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW), pp. 234–238. IEEE (2023)
39. Ji, Y.S., Chen, J.J., Niu, G., Shang, L., Dai, X.Y.: Transfer learning via multi-view principal component analysis. Journal of Computer Science and Technology **26**(1), 81–98 (2011)
40. Johansson, E., Sutinen, K., Lassila, J., Lang, V., Martikainen, M., Lehner, O.M.: Regtech – a necessary tool to keep up with compliance and regulatory changes. ACRN Journal of Finance and Risk Perspectives, Special Issue Digital Accounting **8**, 71–85 (2019)
41. Kashyap, A.R., Nguyen, T.T., Schlegel, V., Winkler, S., Ng, S.K., Poria, S.: Beyond words: A comprehensive survey of sentence representations. arXiv preprint arXiv:2305.12641 (2023)
42. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
43. Li, B., Hou, Y., Che, W.: Data augmentation approaches in natural language processing: A survey. AI Open **3**, 71–90 (2022)
44. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. ArXiv **abs/1907.11692** (2019)
45. Luitel, D., Hassani, S., Sabetzadeh, M.: Using language models for enhancing the completeness of natural-language requirements. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 87–104. Springer (2023)
46. Luitel, D., Hassani, S., Sabetzadeh, M.: Using language models for enhancing the completeness of natural-language requirements. In: Requirements Engineering: Foundation for Software Quality: 29th International Working Conference, REFSQ 2023, Barcelona, Spain, April 17–20, 2023, Proceedings, pp. 87–104. Springer (2023)
47. Matulevičius, R., Tom, J., Kala, K., Sing, E.: A method for managing gdpr compliance in business processes. In: Advanced Information Systems Engineering: CAiSE Forum 2020, Grenoble, France, June 8–12, 2020, Proceedings 32, pp. 100–112. Springer (2020)
48. Maxwell, J.C., Antón, A.I., Swire, P., Riaz, M., McCraw, C.M.: A legal cross-references taxonomy for reasoning about compliance requirements. Requirements Engineering **17**, 99–115 (2012)
49. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)
50. Montgomery, L., Fucci, D., Bouraffa, A., Scholz, L., Maalej, W.: Empirical research on requirements quality: a systematic mapping study. Requirements Engineering **27**(2), 183–209 (2022)
51. Nagel, S.: Cc-news. URL: http://web. archive. org/save/http://commoncrawl. org/2016/10/newsdatasetavailable (2016)
52. Otto, P.N., Anton, A.I.: Addressing legal requirements in requirements engineering. In: 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 5–14 (2007)

53. Pantlin, N., Wiseman, C., Everett, M.: Supply chain arrangements: The abc to gdpr compliance—a spotlight on emerging market practice in supplier contracts in light of the gdpr. Computer law & Security review **34**(4), 881–885 (2018)
54. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
55. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
56. Pullonen, P., Tom, J., Matulevicius, R., Toots, A.: Privacy-enhanced BPMN: Enabling data privacy analysis in business processes models. Software & Systems Modeling **18**(6) (2019)
57. Pushp, P.K., Srivastava, M.M.: Train once, test anywhere: Zero-shot learning for text classification. arXiv preprint arXiv:1712.05972 (2017)
58. Rasiman, R., Dalpiaz, F., España, S.: How effective is automated trace link recovery in model-driven development? In: Requirements Engineering: Foundation for Software Quality, pp. 35–51. Springer International Publishing (2022)
59. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (2019)
60. Sainani, A., Anish, P.R., Joshi, V., Ghaisas, S.: Extracting and classifying requirements from software engineering contracts. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 147–157. IEEE (2020)
61. Schick, T., Schütze, H.: True few-shot learning with prompts—a real-world perspective. Transactions of the Association for Computational Linguistics **10**, 716–731 (2022)
62. Singhal, A., Anish, P.R., Sonar, P., Ghaisas, S.S.: Data is about detail: an empirical investigation for software systems with nlp at core. In: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI, pp. 145–156 (2022)
63. Sleimi, A., Ceci, M., Sannier, N., Sabetzadeh, M., Briand, L., Dann, J.: A query system for extracting requirements-related information from legal texts. In: 2019 IEEE 27th international requirements engineering conference (RE), pp. 319–329. IEEE (2019)
64. Soltana, G., Fourneret, E., Adedjouma, M., Sabetzadeh, M., Briand, L.: Using UML for modeling procedural legal rules: Approach and a study of luxembourg's tax law. In: International Conference on Model Driven Engineering Languages and Systems, pp. 450–466. Springer (2014)
65. Soltana, G., Sannier, N., Sabetzadeh, M., Briand, L.C.: Model-based simulation of legal policies: Framework, tool support, and validation. Software & Systems Modeling **17**(3) (2018)
66. Torre, D., Abualhaija, S., Sabetzadeh, M., Briand, L.C., Baetens, K., Goes, P., Forastier, S.: An ai-assisted approach for checking the completeness of privacy policies against GDPR. In: 28th IEEE International Requirements Engineering Conference (2020)
67. Torre, D., Alferez, M., Soltana, G., Sabetzadeh, M., Briand, L.: Modeling data protection and privacy: application and experience with gdpr. Software & Systems Modeling **20**(6), 2071–2087 (2021)
68. Torre, D., Soltana, G., Sabetzadeh, M., Briand, L.C., Auffinger, Y., Goes, P.: Using models to enable compliance checking against the GDPR: an experience report. In: 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, (2019)
69. Tunstall, L., Reimers, N., Jo, U.E.S., Bates, L., Korat, D., Wasserblat, M., Pereg, O.: Efficient few-shot learning without prompts. In: Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022) (2022)
70. Viera, A.J., Garrett, J.M., et al.: Understanding interobserver agreement: the kappa statistic. Fam med **37**(5), 360–363 (2005)
71. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference

on Empirical Methods in Natural Language Processing: System Demonstrations. Association for Computational Linguistics (2020)
72. Zeni, N., Kiyavitskaya, N., Mich, L., Cordy, J.R., Mylopoulos, J.: GaiusT: Supporting the extraction of rights and obligations for regulatory compliance. Requirements Engineering **20**(1) (2015)
73. Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S.: Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: Proceedings of the IEEE international conference on computer vision, pp. 19–27 (2015)
74. Zowghi, D., Gervasi, V.: The three cs of requirements: consistency, completeness, and correctness. In: International Workshop on Requirements Engineering: Foundations for Software Quality, Essen, Germany: Essener Informatik Beitiage, pp. 155–164 (2002)