Position Papers of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 61–68
DOI: 10.15439/2025F7772
ISSN 2300-5963 ACSIS, Vol. 44

Utilization of Large Language Models for conformity assessment: Chances, Threats, and Mitigations

János Litzinger, Daniel Peters, Florian Thiel
8.52 Embedded Systems
Physikalisch-Technische Bundesanstalt
Berlin, Germany
{janos.litzinger,daniel.peters,florian.thiel}@ptb.de

Florian Tschorsch
Chair of Privacy and Security
Technische Universität Dresden
Dresden, Germany
florian.tschorsch@tu-dresden.de

***Abstract*—Assessing the conformity of software in measurement instruments is a laborious process and a major bottleneck in the process of developing new devices. Large Language Models have been shown to effectively handle complex tasks and have the ability to surpass humans with regard to speed and accuracy. However, integrating them into the technology stack can bring major security and privacy risks. This position paper performs a threat modeling in this context. By addressing the discovered confidentiality risks the paper draws a way for safely implementing Large Language Models as an essential tool in the process of conformity assessment.**

I. Introduction

In the European Union, measurement instruments such as electricity meters, taximeters or automatic weighing instruments are regulated regarding their specific metrological properties. In order to be sold on the European market, these instruments need to pass a conformity assessment that verifies whether the instrument complies to regulatory requirements. Most modern measurement instruments have a software component that handles, among other things, the storage and transmission of measurement data. Thus, this software is also subject to regulation and therefore requires a conformity assessment. The assessment involves searching for relevant information in software documentation provided by the manufacturer and the decision whether it conforms with the requirements defined for the measurement instrument. This process depends on manual labor and is very time consuming.

Large Language Models (LLMs) have been applied to nearly any field in natural language processing (NLP)—from text classification, question answering or information retrieval to named entity recognition. Those models are trained on enormous data sets with trillions of tokens [1], consisting of newspaper articles, websites, books, and social media entries. However, the training data mainly holds publicly available text [1]. The Physikalisch-Technische Bundesanstalt (PTB)[1], Germany's national metrology institute, envisions to leverage its vast amount of textual data to augment existing models with metrological expertise. Especially, highly contextualized tasks such as conformity assessment of software documentation can benefit from models that are adjusted for the metrological domain.

In this position paper, we outline a path forward to streamline conformity assessment by integrating LLMs into the assessment pipeline. Through threat modeling, we identify potential risks associated with deploying LLMs in risk-sensitive environments. Our analysis highlights confidentiality and integrity as the main security objectives in this context. To better understand the current state of research, we review relevant literature on information leakage in LLMs and discuss how it may help to protect LLM-assisted conformity assessment.

The following section provides an overview of the conformity assessment process and the use of NLP methods. Section III performs threat modeling following the established PASTA method [2], while Section IV discusses related research. Section V outlines directions for future work, and Section VI concludes the paper.

II. Conformity assessment

Measurement instruments that are used in commercial or administrative contexts need to deliver reliable, deterministic measurements. Most users of measurement devices or persons affected by them are not always able to verify these measurements and therefore rely on trusting that the instruments function as intended and output correct measurements. Legal metrology ensures this trust by formulating regulations for measurement instruments in those contexts. These regulation not only dictate the hardware but also the software side of these devices. The EU Directive 2014/32/EU [3], better known as Measurement Instruments Directive (MID), harmonizes the national regulations and enables manufactures of measurement instruments to produce for the entire market of the European Union. In order to receive a MID approval, manufacturers need to prove that their product conforms with the requirement of the MID. In practice, this is achieved by providing a Notified Body with the product and the appropriate hardware and software documentation. In Germany, the PTB functions as such a Notified Body and assesses the hardware's and

[1]https://www.ptb.de/cms/en.html

software's conformity with the requirements defined in the MID, whereas for most devices an assessment is only carried out on document basis.

The software is typically assessed along the lines of the WELMEC Guide Software "7.2" [4]. The WELMEC Guide differentiates between different classes of instruments which determine the specific requirements for the software. Those are defined in blocks for separation and download of software as well as for the transmission and storage of measurement data. Furthermore, each class of measurement instrument has its own specific requirement, e.g., electricity meters or automatic weighing instruments. Assessing the software requires a search for the relevant information in the provided documentation and the comparison with the requirements in the WELMEC Guide. The difficulty of searching in the documentation lies in the diversity of those documents. Each manufacturer uses their own terminology, document structure, and composition of different documents. Thus, the assessors need to adapt their search queries to the manufacturer's unique language. Due to its manual nature, this processing step has become a major bottle-neck in conformity assessment and hinders a fast time to market.

A. LLMs for conformity assessment

To process the vast amount of documents that are generated throughout a conformity assessment, PTB developed a software, which allows to search the provided documentation in regard to the requirements defined in the WELMEC Guide. However, it fails to extract most of the information needed for assessing the software. Therefore, a lot of manual labor remains. Nevertheless, the approach showed major advantages to a purely manual procedure and stresses the need for a more automated approach to conformity assessment.

The task of conformity assessment involves two major fields of NLP, namely classification and Information Retrieval (IR). Traditional methods such as tf-idf [5], while being strong baselines for classification and IR, they fail when being exposed to out-of-distribution data. LLMs on the other hand are able to abstract from their training data since they embed words or tokens in a semantically clustered vector space. LLMs use these embeddings and efficiently model word semantics up to sentences, paragraphs and entire documents. Especially the transformer architecture [6] has been shown to deliver state-of-the-art results in various language understanding tasks [7]. Transformers can be trained in parallel on large data sets and thus have been scaled up in recent years to large language models with billions of parameters, trained on trillions of tokens [1], [8]. Due to the huge computational resources needed to create such models, training a large language model from scratch for a specific use case or domain is impracticable. Therefore, the main application of language models shifted to the paradigm of adjusting pre-trained language models to a specific task or domain, better known as *fine-tuning*. This paradigm gave rise of so-called *foundation models* that are trained without a specific task and are later further adjusted. While some models (e.g., [9], [10]) hide their models behind free or paid APIs, other models are published for local use (e.g., [8]). These models are also known as *open weight* models since their weights are freely available for researchers, developers, and the end user.

The approach of adjusting pre-trained models to downstream NLP-tasks has shown impressive results on various benchmark test sets[2] and is therefore suitable for the task of conformity assessment. For the practical usage, we propose a system that make use of an embedding model that has been fine-tuned for retrieval of relevant text chunks for a given query. This is done by fine-tuning the model for document embeddings, mapping the input text to an n-dimensional vector. By embedding the document chunks and queries into this vector space, relevant chunks can be retrieved by a neighborhood search. The retrieved chunks are then used as additional context of a generative model, that has been adapted for the task of conformity assessment. This would enable the user to query the model with respect to certain documents asking whether it is in line with the requirements defined in the WELMEC guide. This method is referred as retrieval-augmented generation (RAG) [11]. While it is possible to set up this RAG-pipeline exclusively with pre-trained models, we assume that those models can benefit from the rich training data for conformity assessment inside PTB. Not only could the envisioned system assist the assessors of software, furthermore it could help manufacturers of measurement instruments compiling the documentation and thus reduce the administrative process even more. Due to the entailed security issues of this concept, we see the need for a thorough threat analysis even in this early stage of development.

III. Threat modeling

The documentation provided by the manufactures is of sensitive nature. Not only does it consist of publicly available documentation such as user manuals, it is rather a full documentation of the internal function of the measurement device. From the overall software architecture to fine details such as start parameter for algorithms — the documentation holds enough information to rebuild the measurement instrument and its software from scratch. Due to that sensitive nature of the documents, confidentiality is of hightest concern when designing software to assist in conformity assessment.

Threat modeling is a systematic approach to identifying potential threats, assessing associated risks, and developing appropriate mitigation strategies. While widely used in software development, it applies more broadly to understanding the security and privacy implications of complex systems. Common threat modeling methods include PASTA [2], STRIDE [12], and LINDDUN [13], which have different focuses and follow different approaches. PASTA (Process for Attack Simulation and Threat Analysis) takes a risk-centric view, aligning business objectives with technical requirements to derive risks from threats and known vulnerabilities. STRIDE identifies threats based on a data flow diagram (DFD) and

[2]https://paperswithcode.com/area/natural-language-processing

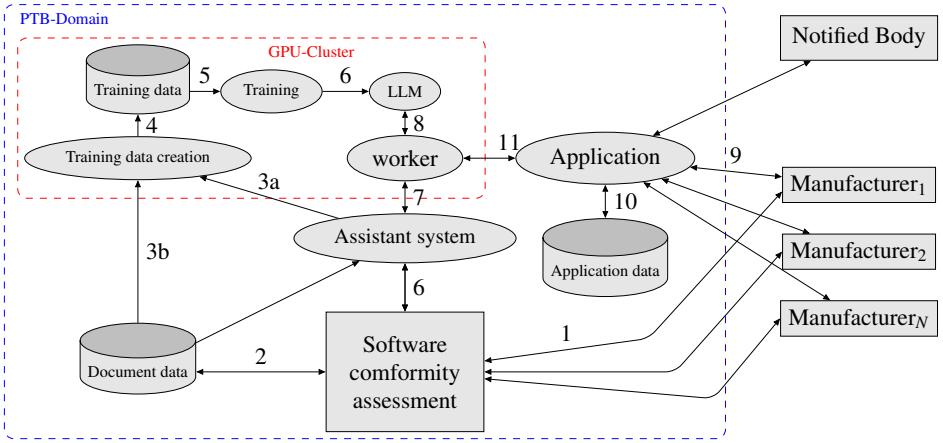

Fig. 1. Data flow diagram of the proposed system. Rectangles reference entities, ellipses symbol process, and cylinders represent data bases. The arrow indicates a flow of data, whereas the direction is indicated by the head. Arrows with heads on either end stand for a bidirectional data flow. The dashed line, here in blue and red, indicates a trust boundary.

categorizes them as spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege. LINDDUN, in turn, defines privacy-focused threat categories such as linkability, identifiability, and non-compliance. Risk assessment methods have also been tailored to measurement instruments [14], however we will hold on to methods known by a broader audience. While a threat-centric approach such as STRIDE might seem suitable for conformity assessment, a risk-driven method like PASTA is more appropriate due to its independence from predefined threat categories. LINDDUN may be well-suited for systems with high privacy requirements; however, practical experience shows that it can become overly detailed. In early design stages, systems often lack the specificity needed for such granular analysis and benefit more from broader, flexible approaches like PASTA.

In the following, we adopt PASTA, which involves seven stages: Stage 1 defines the context, including business objectives. Stage 2 outlines the technology stack and system scope. Stage 3 decomposes the application and identifies its actors. Stage 4 focuses on threat identification, which is linked to known vulnerabilities in Stage 5. Stage 6 simulates potential attacks, and Stage 7 maps the findings back to the original business objectives defined in the initial stage.

Context (Stage 1): The main purpose of the proposed application is to assist the software tester, while assessing the documentation provided by the manufacturer. Secondly, it should help the manufacturer of measurement instruments with compiling the appropriate software documentation needed for the conformity assessment. These documents can hold sensible information such as start parameters, technological innovations or novel solutions to known problems. They might also include typically public information such as User Manuals but since the conformity assessment is done prior to the market launch manufacturer have the interest to keep this information confidential from their competitors. Thus, the documentation documentation can not be made accessible to any 3rd-Party. Furthermore, the regulatory environment of conformity assessment makes it necessary to store these documents up two 10 years. Since the software testers should benefit from the assistant system during their everyday work, the system needs to be designed for high availability.

Technology Stack (Stage 2): The main component of the system is a GPU-Cluster running a free and open-source operating system. Due to the current dependence of the CUDA library[3] while developing LLMs the cluster is equipped with NVIDIA GPUs and thus runs proprietary drivers. The LLMs are pre-trained by 3rd parties such as Meta (Llama), Google (Gemma) or Mistral (Mistral). Usually these bigger models are published as "open-weight"-models, meaning the training algorithm and data is kept secret. Smaller models might be published as "open-source". Most of the models (big or small) are obtained via Huggingface[4] – a platform to publish machine learning models. It also develops a python library for developing and running models that will most likely be used for fine-tuning the models. Two commonly known libraries to work with language models are Ollama[5] and LangChain[6], whereas the former focuses on efficiently running the models, the latter offers an abstraction layer to build applications around language models. Ollama can be run in a docker container or manually installed whereas LangChain can be drawn into the project via the python package manager *pip*.

Application decomposition (Stage 3): The data flow diagram in Fig. 1 shows an abstracted model of the proposed system. The manufacturer provides the documentation to the conformity assessment either via E-Mail, with a link for download or uploads it to a file-sharing system (1). Those documents are usually in the PDF file format but can also consist of text files holding html or in the doc(x) file format. Those documents are stored (2) in a network drive where the software testers can access it. The data can also be accessed

[3]https://docs.nvidia.com/cuda/
[4]https://huggingface.co/
[5]https://ollama.com/
[6]https://www.langchain.com/

(3) by the process that creates the training data for the model. Creating the training data involves transforming the PDF and docx files into a machine-readable format such as Markdown. This can be done by using python libraries such as Nougat[7], Marker[8] or even using a LLM for layout detection. Most importantly the overall structure of the documents need to be kept since the software tester tend to reference the sections in their reports. Hence, the valuable information lies in the connection of assessment report (3a) and documentation (3b). The creation of the training data (4) and its storage are happening on the GPU-cluster (red dashed line), since some pre-processing needs access to the GPUs. The training data is then used to fine-tune existing LLMs for the task of conformity assessment (5). The trained model will be deployed on the same cluster for production (6). The assistant system can then query the model for document embeddings and search or for question-answering. A worker module ingest those queries (7) and feeds them into the model (8) in order to efficiently schedule the tasks. Manufactures or other Notified Bodies have also the possibility to access the model through a dedicated application (9) where they can check their documentation (10) or query the model (11). That way manufacturers are able to compile their documentation prior to handing it in for conformity assessment. Note that the entities *Manufacturer* and *Notified Body* are outside of the PTB trust zone (blue dashed line), whereas the testing persons of the conformity assessment work inside that trust zone.

Threat analysis (Stage 4): There are multiple threats that could affect the application or even the whole service of conformity assessment. Those can be found in Table I. First, there is the danger of data loss. Either the documentation provided by the manufactures or the work done by the software testers could be irretrievably lost (1). That would be costly in financial and reputational terms but would not threat the whole existing application. The same should hold for the data connected to the application used by the manufacturers. The training data (2) itself could also be lost, but since their creation is an automatic process it could be restored by running the process again. There are two major threats: corrupted integrity of data (3), meaning an undetected change of data, intended or unintended. This could affect the correctness of the result of the conformity assessment and is a major threat to the whole service. Connected in some way is the threat that the model itself outputs wrong results (4). This would lead to lower accuracy and effectiveness in the assistant system as well as in the manufacturer facing application. The other threat is violating the confidentiality of the data provided by the manufacturer (5), e.g., sharing the data with an unauthorized third party, as this information could be used to copy products from a competing manufacturer.

Vulnerability analysis (Stage 5): Identifying and addressing system vulnerabilities is essential to gain a clear understanding of the risks. In order to gather helpful insights from the vulnerability analysis, we restrict it to those that directly map to the threats identified in the previous stage. Since the software libraries used in the system are mainly from open-source providers, the system itself is vulnerable to coding and configuration flaws introduces by these libraries. Furthermore, the processes in the system could gain privilege on the server due to wrong configuration or coding imperfections. Additionally, authentication for the manufacturers could be imperfect, such that it would allow a manufacturer to inspect data that does not belong to its account.

TABLE I
THREAT OVERVIEW

	Threat	Description
1	Data loss	Deletion of data needed for conformity assessment
2	Training data loss	Deletion of data used for training the LLM
3	Corrupted data integrity	Undetected change of data needed for conformity assessment and training
4	Wrong model output	Factually incorrect output of the LLM that can lead to wrong decisions in conformity assessment
5	Information leakage	Leakage of information from conformity assessment to an unauthorized third party

However, the usage of LLMs introduces vulnerabilities to the system that are inherent to machine learning models. When used to generate text, LLMs are known to suffer from sometimes misleading or even factual incorrect answers, also known as *hallucination* [15]. They might be queried in such a way that their response is delivering unintended results. This vulnerability is called *prompt injection*. But most importantly to the current use case, multiple research has shown, that machine learning models are prone to reveal information about their training data (e.g., [16], [17], [18]).

Attack Modeling (Stage 6): In the scenario of conformity assessment, the vulnerabilities identified in the previous stage open up an attack surface for an adversary that intends to withdraw stored information. This adversary could be a manufacturer that wants to gain information about its competitors upcoming product. Since this information might be valuable to the malicious manufacturer, it might allocate appropriate resources for such an attack or even hire a contractor. The attacker can use the manufacturer-facing application as an entry point to run an extraction attack against the model. For the application, it would seem like queries to the model and therefore would be undetected. Furthermore, the adversary can use a so called Membership Inference Attack (MIA) either to strengthen the extraction attack or to verify that a certain manufacturer where handing in their documentation for conformity assessment. In a different attack scenario, a manufacturer exploit the usage of an LLM to cheat its way through the conformity assessment. It could hide instructions for the model asking it to only output positive responses. This could be achieved by using text in white color against a white background in the documentation such that a human reviewer

[7]https://github.com/facebookresearch/nougat
[8]https://github.com/VikParuchuri/marker

is unable to see it by only reading the document. Thus, the malicious instructions would be passed further to the model.

Risk and Impact Analysis (Stage 7): Using a prompt injection attack in order to pass the conformity assessment might be viable to some malicious manufacturers but also leave traces in the system. Note that due to the regulatory context the documentation used for the conformity assessment is stored for at least 10 years. Therefore, the risk of being exposed of cheating might be to high. On the other hand, attacking the model to gain information about its training data seems to be a reasonable approach by a manufacturer. The risk of being exposed is relatively low since the attack itself leaves no obvious traces. It might also not trivial to prove that querying the model was an attempt to gain information about the training data. Nevertheless, every prompt to the model and its response should be saved in order to monitor its usage as well as tracking its behavior over time. Therefore, the threat that confidential data is revealed to a third party is an existing risk that should be dealt with when using LLMs in a high risk environment such as in conformity assessment. Maintaining the integrity of the data is a common practice in the conformity assessment in PTB. For every file provided by the manufacture a checksum is calculated and stored. That way even the smallest changes in files can be detected. Therefore, we restrict the further examination of threats to confidentiality of training data in LLMs.

IV. Assessing LLM information leakage

While confidentiality and privacy are distinct concepts, they share common principles. Privacy refers to the rights of individuals or groups to control access to personal data. Confidentiality refers to restricting access to and disclosure of information, including proprietary and personal data—thus overlapping with privacy goals. As such, research on privacy in LLMs offers valuable insights into their behavior when trained on sensitive data, as in the case of conformity assessment.

While it is desired that language models memorize certain training data such as Wikipedia articles in order to return factually correct text, this is not the case for other training data. For example, models trained on clinical notes might reveal sensitive data about the patients health condition violating the patients' privacy ([19], [20]). In addition, research on Google's auto-complete system 'Smart compose' [21] trained on user e-mails showed that such a model memorizes long random numbers, e.g., social security numbers, that can be extracted by prompting the model [17]. This is not a theoretical threat as [18] were able to extract Personal Identifiable Information (PII) such as names, phone numbers and e-mail addresses from the language model GPT-2 [22].

A. Attack Scenarios

The fact that a machine learning model memorizes parts of its training data can be exploited by two major attacks. The most prominent attack is the Membership Inference Attack (MIA), where an adversary tries to infer whether some data point was a member of the training data, hence was used to train the model under attack [16]. While most of the proposed attacks assume access to the output vector of the model (grey-box scenario) [19], some attacks also work on output labels (black-box scenario) [23]. The intuition behind MIA is, that a model is expected to be more "certain" predicting the label or token for data it has seen during training than for unknown data. An adversary can use this information to train a model that predicts the membership status of a given data point. To test and train this model the adversary needs a dataset for which they can be sure that it was part of the training dataset. For proprietary models, the training dataset is usually not available, but as [16] have shown a dataset from a similar distribution is sufficient. Recalling the attack scenario in the previous section a manufacturer that has access to the model would know if its data were used, since their approval is needed when using their data. Furthermore, their dataset is from a similar distribution if they previously handed in documentation for the conformity assessment.

The authors of [16] trained so-called *shadow models* on that data and then queried with unknown data and data it has seen while training. This results in a dataset for classification where a data point consists of the output vector of the shadow models with a binary label indicating whether that data point was part of the training dataset. After being trained on this dataset the attack model is able to predict the membership for a given data point by using the output of the model under attack. This reference based approach has been applied to LLMs by [18] but remains computationally expensive for large models. Other approaches calculate the model's loss over the target sample [24] and extending approaches such as calibrating with zlib entropy [18], and a neighborhood comparison approach [25].

Generative models such as GPT have been shown to be particular vulnerable to attacks that aim to gain PIIs ([18], [26]). For example, *PII reconstruction* aims to reconstruct PII for a given data point. In the case of language models the adversary queries the model with an incomplete sentence or masked item for the to be reconstructed PII, for example "John Doe lives in [MASK], England". The missing item can be filled by a masked language model and the resulting sentence acts as a query to the target model. The perplexity of the target model is then used to infer whether the sentence has been seen during training. Perplexity is the measure for a generative model on how "surprised" by a token the model is. A variant of PII reconstruction is *PII inference* where the adversary wants to infer which item in a set of candidates was part in the training. As for PII reconstruction, the PII candidate is inferred by the lowest perplexity. In the scenario of conformity assessment a adversarial manufacturer could prompt the model for parameters of a known product from a competitor. Both of these attacks assume that the adversary has background knowledge on what PII to extract from the model. However, [26] showed that it is also possible to extract PII by simply generating text. The intuition behind this attack is that the model tends to generate text it has seen during training. The authors generated thousands of sentences and

used a named entity recognition algorithm such as flair[9] or spaCy[10] to find text with PIIs. By cross-checking with the developers of GPT at OpenAI they found that their method was able to extract 23% of PII with a precision of 30%.

B. Quantifying LLM Confidentiality

The leakage of sensitive information through language models is closely tied with memorization of training data ([17], [27], [28]). The intuition behind this observation is that the model first generalizes over data through the first epochs of training and then starts to memorize the data in later epochs. This is due to the fact that it encounters the same text multiple times and adjusts its weights accordingly. In its extreme case, memorization leads to overfitting, where the weights of the model shifted towards the training dataset unable to generalize from it any more. Overfitting is indicated by a bigger difference between the evaluation metric for the training and validation set. While overfitting is seen as a natural marker for memorization and thus information leakage, it has been shown that training data does get memorized without any overfitting of the model ([28], [27]).

Memorization in a model and thus the likelihood to leak its training data is usually measured by the performance of an MIA. While it might be tempting to use accuracy as a metric of performance, it lacks expressiveness when it is applied in the context of confidentiality. While some data might not be extracted and thus the false negative rate rises, false positives directly influence the usability of the attack model by diluting its positive results [29]. Area Under ROC Curve is a slightly more informative measure as it takes different classification thresholds into account. Still, it is an aggregate measure that fails to give a good sense on whether an attack delivers successful results with a low false positive rate. The authors of [29] therefore suggest reporting the true positive rate at extremely low false positive rates, e.g., at 0.1 %.

Another way of evaluating memorization in a model is to measure exposure or extractability. In [17], *canary* text is inserted into the training data holding a "secret" random number. To measure how much a specific canary is memorized by the model they calculate an *exposure* metric using the log-perplexity of the sequence. The authors of [17] report a positive correlation for the number of insertions of a canary and exposure, hence the degree of memorization. They tested, how exposure influences the probability of the canary sequence to be extracted and found that when exposure exceeds a certain threshold, in their case 30, the probability of extraction quickly shifts from near 0 to near 1. This hints that the more a sequence is memorized by a model, the more likely it is to be extracted, by accident or by a malicious actor.

For [18], a string is extractable from an language model (LM) if there exist a prefix or context for which the LM outputs the string. From that they give a definition of memorization where a string is "k-eidetic memorized" if it is extractable from the LM and occurs in k examples of the training data. Hence, if a string is only present in a few documents and can be extracted, it is much worse that if a string occurs all over the training data [18]. Measuring k-eidetic memorization is thus a good method to determine whether a model is vulnerable to disclose parts of its training data.

In [27], a slightly different notion of memorization is used to study the effects of memorization. It defines a string as memorized if there exist a string s and a prompt p such that the output of the LM is equal to s when prompted with p. The authors of [27] found an effect of model size on the speed of memorization. Smaller models need to encounter a training example more often that larger models to fully memorize it. Thus, when training larger models the danger of memorizing sensitive data increases. In [30] also the length of the prompt is taken into account. It reports that larger models generally memorize more of its training data. Not only in overall quantity but also for the particular string. Smaller models tend to output only fractions of a training example or only thematically similar text. In accordance with [18], it found that repetition of examples in the training data increases memorization. Furthermore, if a prompt to a model is longer, then more memorization of the model is discovered. Interestingly, [30] found that some tokens require more context to be extracted from the model.

Furthermore, recent work [31] has tried to predict memorization from smaller models to larger ones in order to give developers a hint, if the model shows unwanted behavior. Even though it was found that small models might not act as a forecast for bigger models, this direction of research is still a challenging path to follow, as different forecasting methods are still left untouched.

C. Mitigations against Privacy Leaks

Information leaks of language models can be mitigated at different levels in the development and deployment of these models. With regard to memorization of PII equivalent material, sanitation of the training data is practical method. This can be done by blacklisting sensitive strings and removing them from the training data. However, [17] notes that this approach, while being best practice, is far from being perfect and can still miss sensitive strings. Moreover, [26] show that while PII scrubbing reduces the extraction rate, it does not protect against membership inference attacks.

As [30] and [17] have discussed, the number of occurrences of a specific training examples increase their chance to be memorized. This observation is in accordance with [26] and [32]. Intuitively, removal of duplicate training examples seems as a promising starting point to reduce memorization in a language model. In [32], the authors showed that by deduplicating the training data, they were able to lower the chance of a membership inference attack. Furthermore, deduplication also benefits the model performance itself, when duplicates are removed between the training and the test set [33].

[9]https://huggingface.co/flair/ner-english-ontonotes-large
[10]https://spacy.io

In addition to methods that apply in the pre-training stage of the language model, which should reduce the overall memorization of the model, privacy preserving training methods such as differentially private stochastic gradient decent (DP-SGD) [34] can be used. In [17], it is shown that by using DP-SGD, the exposure of their inserted canaries dropped significantly. The learning algorithm bases on Differential Privacy (DP) [35] that gives a strong privacy guarantee to individual training examples. While in differential private databases the privacy guarantee is given to individual rows, the situation in huge text corpora is different. It might make sense to apply differential privacy on per document level, yet private and sensitive text might occur across multiple documents. Moreover in the domain of conformity assessment, a manufacturer might use the same components across multiple instruments and thus sensitive information about that component are distributed over multiple documents. Thus, a carefully defined usage of differential private learning algorithms is necessary to protect training data from being leaked. Despite DP-SGD presents itself as an optimal mitigation against MIAs and extraction attacks, it is far from being perfect, since it comes with a utility cost manifesting in increased compute and decrease performance of the model.

Sensitive information can also be protected in a post-training stage. Simply filtering out sensitive information during inference is a naive approach that cannot hold to its promises. The authors [36] reported that a filter-approach can prevent generating verbatim text from the training data. However, the model is still able to produce text holding sensitive information by avoiding verbatim repetition and generating alternative texts with synonyms. Applying DP to inference of the model is another approach of preserving privacy of the training data. In [37], multiple LMs were fine-tuned with disjoint private data. During inference all models are queried and if all models come to a consensus about the predicted token, the generated token is seen as not holding private data. On the other hand, if the models disagree the prediction of a public model is mixed in. While this approach achieves comparable privacy to DP-SGD, storage and computation increase.

As has been shown above, data curation methods such as deduplication can significantly reduce memorization of training data but do not fully prevent a model from leaking private data. Differential Private Learning on the other hand can give such a privacy guarantee, but suffers from increased compute and is prone to ill-defined privacy scopes. It also comes with a utility cost. Filtering a models output only prevent certain text from being generated by do not apply for non-verbatim extraction.

V. Opportunities for further research

In the following, we outline directions for future research aimed at assessing and mitigating information leakage risks in LLMs within the context of conformity assessment.

Building on the attacks outlined in Section IV-A, further research should focus on evaluating their impact on the proposed system, particularly for embedding or classification models, as these are most relevant. In order to evaluate these attacks, a notion of sensitive strings need to be developed. Analogously to PII for the privacy domain, extraction attacks are to be evaluated with regard to how many sensitive strings can be extracted. Furthermore, gradually weighting information leakage can help to grade the severity of a violation of confidentiality. Additionally, broader forms of extraction need to be examined. While parts of the training data may contain secret sequences, it remains unclear whether entire concepts, such as novel solutions developed by the manufacturer, can be extracted. In such cases, no suitable evaluation method exists to quantify the information leakage, as current approaches rely primarily on string comparison. Semantic string comparison may offer a promising starting point for assessing whether entire concepts can be extracted from a model.

Mitigation strategies against information leakage would benefit from this research, as improved semantic string comparison and a notion of sensitive strings could enable effective data sanitation similar to PII scrubbing and support deduplication. In the long term, manufacturers, notified bodies, and conformity assessment could collaboratively define a training dataset that is safe for model training, with minimal risk of memorization. While such a data set would be a desirable solution, its development and coordination are likely to be time-consuming. In the meantime, creating synthetic data sets might be a suitable interim solution.

VI. Conclusion

In this paper, we have shown that LLMs can be utilized in conformity assessment of software in measurement devices. We sketched a system that can benefit software testers, Notified Bodies as well as manufacturers. Due to the sensitive nature of the documents involved in conformity assessment, we conducted a threat modeling using the established PASTA framework. The threat modeling yielded a possible threat of violating confidentiality. A literature review on information leakage by LLMs showed that LLMs tend to memorize parts of their training data, which can be extracted via multiple attack methods. Fortunately, mitigation such as data sanitation and differential privacy in training exist but come with a certain utility cost. Nevertheless, the overall advantages of utilizing LLMs for conformity assessment persist and the path of integrating them into an assistant system for software testers, manufacturers, and other notified bodies should be consistently followed in order to streamline conformity assessment.

References

[1] L. Gao *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," 2020, arXiv:2101.00027.

[2] T. Ucedavélez and M. M. Morana, *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis*. Hobekin: John Wiley & Sons, 2015.

[3] European Parliament and C. of the European Union, "Directive 2014/32/EU of the European Parliament and of the council," 2014. [Online]. Available: https://eur-lex.europa.eu/eli/dir/2014/32/oj

[4] WELMEC Guide 7.2, "Software guide (EU measuring instruments directive 2014/32/EU," 2023. [Online]. Available: https://www.welmec.org/welmec/documents/guides/7.2/2023/WELMEC_Guide_7.2_2023.pdf

[5] K. Spärck Jones, "A statistical interpretation of termspecificity and its application in retrieval," *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972. doi: 10.1108/eb026526

[6] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[7] A. Wang *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf

[8] A. Dubey *et al.*, "The llama 3 herd of models," 2024, arXiv:2407.21783.

[9] OpenAI *et al.*, "Gpt-4 technical report," 2024, arXiv:2303.08774.

[10] Gemini Team *et al.*, "Gemini: A family of highly capable multimodal models," 2024, arXiv:2312.11805.

[11] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021, arXiv:2005.11401.

[12] L. Kohnfelder and P. Garg, "The threats to our products," *Microsoft Interface*, 1999. [Online]. Available: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx

[13] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering*, vol. 16, no. 1, pp. 3–32, 2011. doi: 10.1007/s00766-010-0115-7

[14] M. Esche and F. Thiel, "Software risk assessment for measuring instruments in legal metrology," in *Proceedings of the 2015 Federated Conference on Computer Science and Information Systems*, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, 2015. doi: 10.15439/2015F127 pp. 1113–1123.

[15] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023. doi: 10.1145/3571730

[16] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, " Membership Inference Attacks Against Machine Learning Models ," in *2017 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2017. doi: 10.1109/SP.2017.41. ISSN 2375-1207 pp. 3–18.

[17] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019. ISBN 978-1-939133-06-9 pp. 267–284. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/carlini

[18] N. Carlini *et al.*, "Extracting training data from large language models," 2021, arXiv:2012.07805.

[19] F. Mireshghallah, K. Goyal, A. Uniyal, T. Berg-Kirkpatrick, and R. Shokri, "Quantifying privacy risks of masked language models using membership inference attacks," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022. doi: 10.18653/v1/2022.emnlp-main.570 pp. 8332–8347.

[20] E. Lehman, S. Jain, K. Pichotta, Y. Goldberg, and B. Wallace, "Does BERT pretrained on clinical notes reveal sensitive data?" in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova *et al.*, Eds. Online: Association for Computational Linguistics, Jun. 2021. doi: 10.18653/v1/2021.naacl-main.73 pp. 946–959.

[21] M. X. Chen *et al.*, "Gmail smart compose: Real-time assisted writing," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3292500.3330723. ISBN 9781450362016 p. 2287–2295.

[22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[23] W. Shi *et al.*, "Detecting pretraining data from large language models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=zWqr3MQuNs

[24] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, 2018. doi: 10.1109/CSF.2018.00027 pp. 268–282.

[25] J. Mattern, F. Mireshghallah, Z. Jin, B. Schoelkopf, M. Sachan, and T. Berg-Kirkpatrick, "Membership inference attacks against language models via neighbourhood comparison," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023. doi: 10.18653/v1/2023.findings-acl.719 pp. 11 330–11 343.

[26] N. Lukas, A. Salem, R. Sim, S. Tople, L. Wutschitz, and S. Zanella-Béguelin, "Analyzing leakage of personally identifiable information in language models," in *2023 IEEE Symposium on Security and Privacy (SP)*, 2023. doi: 10.1109/SP46215.2023.10179300 pp. 346–363.

[27] K. Tirumala, A. Markosyan, L. Zettlemoyer, and A. Aghajanyan, "Memorization without overfitting: Analyzing the training dynamics of large language models," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 38 274–38 290. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/fa0509f4dab6807e2cb465715bf2d249-Paper-Conference.pdf

[28] F. Mireshghallah, A. Uniyal, T. Wang, D. Evans, and T. Berg-Kirkpatrick, "Memorization in nlp fine-tuning methods," 2022, arXiv:2205.12506.

[29] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramèr, "Membership inference attacks from first principles," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022. doi: 10.1109/SP46214.2022.9833649 pp. 1897–1914.

[30] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang, "Quantifying memorization across neural language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. [Online]. Available: https://openreview.net/forum?id=TatRHT_1cK

[31] S. Biderman *et al.*, "Emergent and predictable memorization in large language models," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 28 072–28 090. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/59404fb89d6194641c69ae99ecdf8f6d-Paper-Conference.pdf

[32] N. Kandpal, E. Wallace, and C. Raffel, "Deduplicating training data mitigates privacy risks in language models," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 10 697–10 707. [Online]. Available: https://proceedings.mlr.press/v162/kandpal22a.html

[33] K. Lee *et al.*, "Deduplicating training data makes language models better," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022. doi: 10.18653/v1/2022.acl-long.577 pp. 8424–8445.

[34] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2976749.2978318 p. 308–318.

[35] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proceedings of the Third Conference on Theory of Cryptography*, ser. TCC'06. Berlin, Heidelberg: Springer-Verlag, 2006. doi: 10.1007/11681878_14 p. 265–284.

[36] D. Ippolito *et al.*, "Preventing generation of verbatim memorization in language models gives a false sense of privacy," in *Proceedings of the 16th International Natural Language Generation Conference*, C. M. Keet, H.-Y. Lee, and S. Zarrieß, Eds. Prague, Czechia: Association for Computational Linguistics, Sep. 2023. doi: 10.18653/v1/2023.inlg-main.3 pp. 28–53.

[37] A. Ginart, L. van der Maaten, J. Zou, and C. Guo, "Submix: Practical private prediction for large-scale language models," 2022, arXiv:2201.00971.