One Size Does Not Fit All: On the Role of Batch Size in Classifying Requirements with LLMs

Ashley T. van Can
Utrecht University
Utrecht, the Netherlands
0009-0001-1190-8327

Fatma Başak Aydemir
Utrecht University
Utrecht, the Netherlands
0000-0003-3833-3997

Fabiano Dalpiaz
Utrecht University
Utrecht, the Netherlands
0000-0003-4480-3887

***Abstract*—Automated requirements classification is a widely explored research topic in requirements engineering. In particular, the distinction between functional and non-functional requirements has received considerable attention. Recently, Large Language Models (LLMs) have demonstrated potential in automating requirement classification tasks. Although existing research emphasizes effective prompting strategies, it provides limited evaluation of how many requirements should be processed within a single prompt sequence as a batch to optimize classifier performance. The batch size is relevant when computational resources are constrained, as minimizing the number of LLM calls becomes essential. Moreover, batching requirements may provide the model with additional contextual information, potentially improving the classification performance. Therefore, this study investigates the impact of batch size on classification performance. We assess how three locally deployable models, Llama3-8B, Gemma3-12B, and DeepSeek-Distill-Qwen 14B, perform in classifying requirements according to their functional and quality aspects. Our findings show that the optimal batch size depends on both the dataset and the model. Selecting a batch size of one by default, which is often used for the classification tasks, does not always yield optimal results. Our findings highlight the importance of selecting a suitable batch size before performing classification tasks.**

***Index Terms*—requirements engineering, requirements classification, quality requirements, large language models**

I. Introduction

Within the field of Requirements Engineering (RE), automating requirements classification has received significant research attention over the years. Studies have focused on identifying requirements from specification documents [1], classifying non-functional requirements [2], [3], and categorizing user reviews into bug reports, feature requests, user experience feedback, and ratings [4].

In recent years, research on requirements classification has increasingly turned to the use of Large Language Models (LLMs). For instance, the research by Ronanki *et al.* [5] evaluated and compared five different prompts, each following a unique pattern, for binary classification of non-functional and functional requirements. Zadenoori *et al.* [6] proposed an approach to optimize the formulation process for distinguishing between functional and quality-related requirements. Furthermore, Motger *et al.* [7] employed encoder-only models to extract features from mobile app reviews.

Decoder-only LLMs, like the well-known OpenAI's GPT models, can be readily used for this task as they are pre-trained. Nonetheless, they are computationally demanding and their use for classifying requirements may lead to potentially high costs and energy footprints due the many LLM calls. In particular, many studies [6], [8], [9] classify a single requirement per prompt to simplify post-processing, which can significantly increase resource consumption. On the other hand, providing multiple requirements in a single prompt may offer additional context to the model, potentially enhancing performance [10]. However, excessive large batch sizes can overwhelm the model's memory capacity, leading to context loss. For example, Lui *et al.* [11] show that in large textual documents or descriptions, performance decreases when the model needs to access information in the middle of the text.

Thus, selecting an appropriate batch size is important for balancing resource efficiency and classification performance. Despite its importance, current research lacks a systematic evaluation of LLM performance across varying batch sizes. To address this gap, we conduct research into the impact of batch size on requirements classification tasks, driven by the following main research question:

MRQ. To what extent does the batch size affect the performance of large language models in a classification task?

In this study, we focus on classifying requirements into functional and quality requirements, considering the revised functional and non-functional classification problem as defined by Dalpiaz *et al.* [12], which, building on the theory by Li *et al.* [13], treats the classification of the functional aspect independent from the identification of a quality aspect.

To answer our main research question, we conduct experiments comparing three popular decoder-only LLMs that are locally deployable: Llama3-8B, Gemma3-12B, and DeepSeek-Distill-Qwen 14B. We make the following contributions:

- We investigate the performance of three well-established LLMs using seven different batch sizes across four distinct datasets.
- We conduct a qualitative analysis to improve our understanding of performance across varying batch sizes.

Our experimentation reveals that the performance of each LLM seem to fluctuate across different batch sizes. This observation appears to be also dependent on the choice of dataset and model. For our tested models, the classification of quality requirements proves to be more challenging than the

classification of functional requirements. We synthesize our results into findings that include recommendations for research and practice.

The remainder of the paper is structured as follows. Section II presents the research problem and refined research questions. Section III describes the selected dataset and models with an explanation of the formulated prompts and metrics used. Section IV presents the quantitative findings of the experiments while Section V interprets the results, summarizes our main findings and covers the results of the qualitative analysis. Section VI discusses the threats to validity and Section VII compares our study to related work. Finally, we conclude this study in Section VIII.

II. Research setup

A. Requirements classification problem

We focus on the classification of functional and quality requirements, considering the revised functional and non-functional classification problem as defined by Dalpiaz *et al.* [12], which is inspired by the ontological analysis of requirements types by Li *et al.* [13].

Dalpiaz *et al.* [12] explored high-level linguistic features (including dependency grammars) when developing machine learning models for classifying functional and quality requirements. Their study treats the decision of whether a requirement is functional independent from the decision on whether the requirement includes a quality aspect. For this reason, they consider these as distinct classification tasks. They collected eight datasets and manually annotated the data with the following labels:

- Functional aspect: the requirement has a functional goal or constraint (class 1) or does not (class 0).
- Quality aspect: the requirement includes a quality goal or constraint (class 1) or does not (class 0).

We build on the same framework, conducting the experiments for each of the two classification tasks independently.

B. Research Questions

We refine our main research question into four distinct research questions, which we aim to address through quantitative and qualitative analysis of the experiments we conducted.

First, we aim to gain insight into the performance of LLMs across different batch size settings:

RQ1. What impact do different batch size settings have on model performance when classifying functional and quality requirements?

To answer RQ1, we experiment with various LLM architectures and evaluate their effectiveness.

Extensive research has been conducted on the use of ensemble learning to improve the performance of classifiers, particularly in machine learning [14]. Ensemble learning uses multiple algorithms to generate predictions and aggregates these predictions based on a voting mechanism. Given that these methods are employed to enhance robustness and stability, we aim to use a similar approach to analyze its effect on different batch sizes and compare the results with the individual models. This leads us to our second research question:

RQ2. How does a LLM ensemble perform compared to individual LLMs across different batch sizes?

To gain a better understanding of the models' performance, we examine whether certain characteristics of the requirements influence classification performance. Previous research into requirements in issue tracking systems has shown that requirements statements may include both functional and non-functional aspects [15]. In these cases, however, the quality aspect is often less explicit. Based on these observations, we anticipate challenges in classifying such requirements, which leads to our third research question:

RQ3. To what extent do requirements that combine functional and quality aspects affect model performance?

Finally, we aim to obtain insight into the impact of dataset selection on the performance of LLMs and how this varies depending on the batch size setting:

RQ4. To what extent is the performance of the models affected by the used dataset?

We answer RQ3 and RQ4 through our qualitative analysis based on the results of the experiments.

III. Experimental settings

In this section, we first describe the datasets (Section III-A) and models (Section III-B) employed in our experiments. Then, we explain the prompt we used (Section III-C), the processing steps (Section III-D), and the selected metrics (Section III-E).

A. Datasets

To investigate the impact of batch size on LLM performance, we conduct experiments on a subset of the data proposed in the online appendices of [12], [16].

a) Original datasets: Dalpiaz *et al.* [12] examined general linguistic features for training machine learning models to classify functional and quality requirements and publicly share five of the annotated datasets. In [16], the same authors proposed a robust pipeline for conducting and reporting empirical studies. Two of their research questions involve the classification of functional and quality requirements, for which they release additional publicly available datasets.

Table I presents four datasets selected for this experiment based on their availability and relatively balanced class distributions. The first three columns show the dataset size and distribution of functional (F) and quality (Q) requirements of the original datasets. The fourth column shows the number of requirements with both a functional and a quality aspect (F+Q). The remaining four columns provide the same information for the sampled version.

Given the multi-label nature of this classification problem, we adopt a strategy to split the problem into two separate binary classification tasks. The first task aims to determine whether a requirement contains a functional aspect, while the second task aims to identify requirements that contain a quality aspect (consistent with the methodology used in [6]).

TABLE I: Descriptive statistics of the datasets and the sample.

	Original				Sample			
Dataset	Size	F	Q	F+Q	Size	F	Q	F+Q
PROMISE	625	310	382	80	192	98	120	31
ERec mgmt	228	163	149	98	192	135	129	83
Leeds library	85	44	61	21	64	36	45	17
OAppT	140	84	53	28	128	83	49	28

b) Sampling and batch setting: We experiment with batch sizes of 1, 2, 4, 8, 16, 32, and 64, warranting that all configurations cover the same set of requirements. To ensure that all data were consistent with multiples of 64, we decided to sample the data. In addition, the requirements were randomly selected to reduce any order-related bias. Consider, for instance, a batch size of 64 on the PROMISE dataset, the LLM is prompted to classify the functional aspect three times: for requirements 1 to 64, 65 to 128, and 129 to 192. When the dataset size permitted, we selected samples of up to 192 requirements (in multiples of 64) to manage the computational costs associated with running experiments across different batch sizes.

B. Models

We conducted our experiments using three widely adopted LLM architectures:

- Gemma 3 12B[1]
- DeepSeek R1 Distill Qwen 14B[2]
- Llama 3.1 8B-instruct[3]

We chose these architectures because of their current reputation as some of the most recognized open-source architectures. Additionally, we selected these specific model sizes to minimize computational resource usage without compromising accuracy. Apart from the selected models, we also experimented with smaller versions (e.g., DeepSeek-Distill-Qwen 7b), but these yielded high variability in the results (e.g., inability to recognize the specified template), making automatic post-processing problematic and impractical.

The models were implemented using the Transformers library in Python[4]. To ensure reproducible results, we set the temperature to 0.01. After an initial experiment, we discovered that performing the experiments in multiple runs yields identical generated output; however, to further rule out potential fluctuations, we also controlled the seed (42).

To answer RQ2, we utilize the predictions of the three independent models and apply majority voting as the voting mechanism for the ensemble model.

We deployed Llama 3.1 on a NVIDIA A100 GPU compute node. Given their larger sizes, we used a NVIDIA H100 GPU compute node for Gemma 3 and DeepSeek-Distill-Qwen.

[1]https://huggingface.co/google/gemma-3-12b-it
[2]https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-14B
[3]https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
[4]https://huggingface.co/docs/transformers/index

C. Prompt Engineering

a) Two distinct classification tasks: The classification problem involves two different tasks: 1) determining whether the requirement concerns a functionality, and 2) identifying whether the requirement addresses a quality aspect. Since we acknowledge that a requirement can reflect both a functionality and a quality aspect (i.e., a requirement does not have to be exclusively functional or quality-related), we have split the experiments into two separate sessions for each requirement. Therefore, we have formulated a distinct prompt for each task. The final prompts can be found in our online appendix [17].

b) Sequence per session: Based on an initial exploratory experiment, we observed that certain models, particularly DeepSeek 14B, generally produce more variable output than other models. Given the importance of obtaining outputs conforming to the specified post-processing template, we included a follow-up prompt after the initial response within each session, in which we instruct the LLM to comply with the template to obtain a final, correctly formatted output.

c) Prompt strategy: When focusing on decoder-only LLMs [18], the performance of these models is highly influenced by the formulation of the corresponding prompt. To assist users in formalizing their prompts, several guidelines have been proposed. One of the most prominent guides is the one by White *et al.* [19]. In addition, organizations developing and introducing new LLMs often include prompt guidelines in their documentation[5][6]. To formalize our prompt, we considered the following well-established prompting patterns to date: persona pattern, template pattern and few-shot prompting:

- *Persona pattern*: this requests the LLM to take a specific perspective, allowing it to determine which output to generate and which details to focus on [19]. Hence, to ensure a consistent viewpoint, each prompt starts with a request to the LLM to act as a requirements analyst.
- *Template pattern*: it enables the user to guide the LLM in producing output in a format that it would not normally use [19]. We request the LLM to generate the output based on the requirement and label numbers, for which we also provide a partial placeholder.
- *Few-shot pattern*: In-context few-shot prompting is widely applied and involves providing a number of demonstration examples in the prompt [20]. Unlike traditional fine-tuning on language models, few-shot prompting does not require updating the model weights. Current research on few-shot prompting aims to develop techniques for selecting an optimal subset of demonstration examples [21]. Here, we employed a manual approach, in which the second and third author collaborated to the creation of five examples per class (i.e., 10 examples per batch) to ensure a diverse range of cases was represented. We opted for this procedure as these authors have substantial experience in requirements classification.

[5]https://platform.openai.com/docs/guides/text?api-mode=responses
[6]https://www.llama.com/docs/how-to-guides/prompting/

D. Post processing

We automated the processing of the output with the help of the defined template. In general, no manual intervention was required, with a few exceptions. On some occasions, Llama deviated slightly from the template, for example, by skipping a requirement number or by repeating the prediction. Due to their limited occurrence (fewer than 10 cases across all experiments) and easy detection, we manually corrected the values based on the conversation history or counted them as incorrect predictions.

E. Metrics

The following metrics are used to gain insight into the performance of the LLMs:

- *F1 score*: the harmonic mean of recall and precision.
- *Recall*: ratio between the number of true positive results and the total number of positive cases.
- *Precision*: ratio between the number of true positive results and the total number of cases predicted as positive.
- *Specificity*: ratio between the number of true negative results and the total number of negative cases.

To present the results, we chose the F1 score as it is a balanced measure of recall and precision, making it easier to identify performance trends across different settings. We examine recall and precision scores to gain a better understanding of overall performance. In addition, we selected specificity to evaluate performance for the negative cases.

IV. Quantitative findings

We present the quantitative results of the experiments. First, we show the performance of each model individually. Subsequently, we present the results of an ensemble model, along with a comparison between all models.

A. Performance of individual LLMs

We conducted our experiments on the four datasets using the three selected LLMs. Table II–IV display the F1 scores for each distinct model. The tables show the values across the different datasets, for each classification problem. The gray-highlighted cells indicate the minimum and maximum F1 score for the specific dataset and classification task.

DeepSeek's performance is shown in Table II. The F1 scores of the functional classification task remain relatively stable across the different datasets and batch sizes. In contrast, the performance of the quality classification task varies depending on the batch size: a batch size of 1 yields the worst results, with the lowest average F1 score and a high variation across datasets. A batch size of 32 achieves the highest average F1 scores and the lowest variation.

Table III shows Llama's performance using different batch sizes. As can be observed in the table, the average performance for both tasks decreases as the batch size increases. For the quality classification task, the variation between different datasets is highest for batch size 1 and 32.

The performance of Gemma is presented in Table IV. Similar to DeepSeek, the F1 score for the functional classification

TABLE II: DeepSeek F1 Scores for functional and quality requirements, with different batch sizes.

(a) Functional Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.835	0.857	0.836	0.867	0.849	0.016
2	0.828	0.833	0.835	0.901	0.849	0.035
4	0.800	0.809	0.836	0.885	0.833	0.038
8	0.809	0.829	0.840	0.886	0.841	0.033
16	0.791	0.835	0.839	0.849	0.828	0.026
32	0.800	0.844	0.838	0.862	0.836	0.026
64	0.809	0.813	0.839	0.808	0.817	0.015
Mean	0.810	0.831	0.838	0.866		
SD	0.016	0.017	0.002	0.031		

(b) Quality Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.615	0.765	0.288	0.111	0.445	0.299
2	0.667	0.837	0.602	0.323	0.607	0.214
4	0.636	0.820	0.558	0.444	0.615	0.158
8	0.657	0.834	0.557	0.478	0.631	0.154
16	0.657	0.833	0.611	0.375	0.619	0.189
32	0.638	0.893	0.623	0.612	0.691	0.135
64	0.613	0.892	0.685	0.513	0.676	0.160
Mean	0.640	0.839	0.560	0.408		
SD	0.021	0.044	0.128	0.161		

TABLE III: Llama F1 Scores for functional and quality requirements, with different batch sizes.

(a) Functional Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.809	0.770	0.828	0.794	0.800	0.025
2	0.756	0.788	0.833	0.806	0.796	0.033
4	0.761	0.780	0.831	0.792	0.791	0.030
8	0.778	0.778	0.827	0.794	0.794	0.023
16	0.759	0.788	0.828	0.802	0.794	0.029
32	0.773	0.743	0.827	0.802	0.786	0.036
64	0.701	0.724	0.828	0.756	0.752	0.055
Mean	0.762	0.767	0.829	0.792		
SD	0.032	0.025	0.002	0.017		

(b) Quality Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.882	0.889	0.796	0.673	0.810	0.101
2	0.828	0.832	0.786	0.684	0.783	0.069
4	0.822	0.837	0.818	0.648	0.781	0.089
8	0.841	0.831	0.812	0.712	0.799	0.060
16	0.846	0.812	0.812	0.631	0.775	0.098
32	0.784	0.807	0.811	0.528	0.732	0.137
64	0.800	0.800	0.763	0.603	0.741	0.094
Mean	0.829	0.830	0.800	0.640		
SD	0.032	0.030	0.020	0.061		

task remains stable across different batch sizes and shows little variation between datasets. The performance of the quality classification task varies significantly depending on the batch size and the dataset. The model performs increasingly better as the batch size scales up to 8, but decreases when the batch size increases further. The best performing batch size for Gemma is eight, as this yields the highest average F1 score.

TABLE IV: Gemma F1 Scores for functional and quality requirements, with different batch sizes.

(a) Functional Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.818	0.810	0.833	0.798	0.815	0.015
2	0.867	0.824	0.831	0.830	0.838	0.020
4	0.847	0.810	0.834	0.841	0.833	0.016
8	0.847	0.815	0.832	0.830	0.831	0.013
16	0.814	0.814	0.831	0.834	0.823	0.011
32	0.828	0.810	0.808	0.838	0.821	0.015
64	0.818	0.814	0.801	0.838	0.818	0.015
Mean	0.834	0.814	0.824	0.830		
SD	0.020	0.005	0.014	0.015		

(b) Quality Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.763	0.887	0.517	0.400	0.642	0.223
2	0.696	0.833	0.584	0.400	0.628	0.183
4	0.701	0.856	0.619	0.553	0.682	0.131
8	0.747	0.833	0.713	0.568	0.715	0.110
16	0.658	0.813	0.670	0.575	0.679	0.099
32	0.623	0.792	0.630	0.642	0.672	0.081
64	0.738	0.737	0.693	0.560	0.682	0.084
Mean	0.704	0.822	0.632	0.528		
SD	0.050	0.048	0.068	0.092		

In Section IV-C, we provide additional details on the performance of each model, accompanied by a comparative analysis.

B. Ensemble performance

As shown by the individual performance of each model, the performance for a single task (functional versus quality) varies depending on the model and batch size setting. Considering the strengths of each model in specific scenarios, a hybrid version of the models may demonstrate superior performance. To answer RQ2, we investigate whether the use of an ensemble models yields the best performance and examine the impact over the batch size settings. This ensemble approach utilizes the prediction of the three independent models and employs majority voting to generate a final prediction.

Table V shows the performance of the ensemble of the models. The functional identification task yields a stable F1 score for all settings, as expected based on the individual performance of Gemma and DeepSeek. The average performance for quality classification increases as the batch size increases, up to batch size 8, after which it decreases only slightly. The configurations with the largest (64) and smallest (1, 2, and 4) batch sizes show the highest variance.

TABLE V: LLM ensemble F1 Scores for functional and quality requirements, with different batch sizes.

(a) Functional Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.837	0.810	0.833	0.813	0.823	0.014
2	0.837	0.821	0.836	0.836	0.832	0.008
4	0.809	0.805	0.835	0.848	0.824	0.020
8	0.818	0.814	0.836	0.824	0.823	0.010
16	0.787	0.830	0.833	0.832	0.820	0.023
32	0.828	0.803	0.833	0.840	0.826	0.016
64	0.809	0.805	0.828	0.822	0.816	0.011
Mean	0.818	0.813	0.833	0.831		
SD	0.018	0.010	0.003	0.012		

(b) Quality Requirements

Batch Size	Dataset				Mean	SD
	Leeds	PM	ER	OAppT		
1	0.810	0.907	0.525	0.400	0.660	0.237
2	0.740	0.858	0.677	0.459	0.684	0.167
4	0.765	0.889	0.689	0.585	0.732	0.128
8	0.787	0.876	0.745	0.658	0.766	0.091
16	0.709	0.855	0.739	0.608	0.728	0.102
32	0.641	0.861	0.696	0.659	0.714	0.100
64	0.773	0.864	0.733	0.554	0.731	0.130
Mean	0.746	0.873	0.686	0.561		
SD	0.057	0.019	0.076	0.098		

C. Model comparison for identifying qualities

As shown in Tables II–V, the differences between the models are most pronounced in the quality classification task. Therefore, we provide additional performance details on the quality classification task, which is the class that originated the research on requirements classification [2]. The same analysis for the functional classification task is in our online appendix.

Table VI shows the mean precision scores of each model for the four datasets, along with the corresponding standard deviation. The gray cells indicate the two best performances per batch size, and the underlined cells represent the optimal batch size for each model in terms of precision. As shown, DeepSeek has one of the highest precision scores for all settings, with the ensemble model achieving a well-balanced performance across all models. In contrast, Llama showcases the lowest precision scores for all settings.

Table VII shows the mean recall scores of the models on the four datasets, as well as the standard deviation. The table clearly demonstrates how the relatively high precision results for DeepSeek and Gemma are contrasted by a limited recall. Also, there is a visible variation in recall across the batch size settings, a behavior which is less prominent for Llama.

Lastly, the specificity scores are shown in Table VIII. The table reveals the low specificity scores of the Llama model and the relatively high scores for the other models. Furthermore,

TABLE VI: Model precision for quality requirements.

Batch Size	DeepSeek		Gemma		Llama		Ensemble	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.88	0.19	0.88	0.14	0.75	0.09	0.88	0.14
2	0.93	0.10	0.93	0.08	0.72	0.09	0.89	0.14
4	0.89	0.14	0.86	0.07	0.71	0.09	0.86	0.09
8	0.90	0.06	0.86	0.04	0.73	0.06	0.88	0.07
16	0.90	0.10	0.84	0.08	0.71	0.11	0.85	0.06
32	0.85	0.11	0.85	0.08	0.66	0.16	0.81	0.09
64	0.78	0.13	0.83	0.07	0.66	0.12	0.78	0.12
SD	0.05		0.03		0.03		0.04	

TABLE VII: Model recall for quality requirements.

Batch Size	DeepSeek		Gemma		Llama		Ensemble	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.33	0.26	0.53	0.25	0.89	0.13	0.55	0.27
2	0.47	0.22	0.49	0.20	0.87	0.05	0.56	0.18
4	0.48	0.16	0.57	0.16	0.87	0.11	0.64	0.15
8	0.50	0.17	0.62	0.15	0.88	0.11	0.70	0.14
16	0.49	0.20	0.57	0.11	0.86	0.13	0.64	0.12
32	0.59	0.16	0.56	0.09	0.83	0.13	0.64	0.11
64	0.60	0.18	0.59	0.11	0.85	0.05	0.70	0.15
SD	0.09		0.04		0.02		0.06	

specificity is consistently and substantially lowest when using batch size 64 for DeepSeek, Llama, and the ensemble model.

TABLE VIII: Model specificity for quality requirements.

Batch Size	DeepSeek		Gemma		Llama		Ensemble	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.98	0.01	0.93	0.03	0.51	0.28	0.93	0.03
2	0.97	0.02	0.96	0.03	0.46	0.23	0.94	0.03
4	0.94	0.06	0.86	0.09	0.44	0.30	0.85	0.08
8	0.91	0.07	0.82	0.14	0.45	0.33	0.79	0.20
16	0.94	0.09	0.83	0.10	0.44	0.30	0.82	0.12
32	0.86	0.09	0.83	0.17	0.36	0.26	0.77	0.16
64	0.73	0.24	0.78	0.18	0.36	0.18	0.67	0.28
SD	0.09		0.06		0.06		0.10	

V. Discussion

We first use the findings from Section IV to answer RQ1 and RQ2, and attempt to provide a plausible explanation for the observed performance. Subsequently, we report the findings from our qualitative analysis of the experiment, thereby answering RQ3 and RQ4.

A. Discussion Quantitative findings

Tables II–V in Section IV illustrate the performance trends for the seven batch size settings. These results highlight not only the variations in average performance, but also in the variance between datasets. While a high average F1 score is desirable when selecting an optimal batch size, it is equally important to ensure this performance is robust across datasets.

For Llama, for example, a batch size of 1 achieves the highest mean F1 score; however, performance for the quality classification task fluctuates substantially across the datasets. Even though the F1 score for batch size of 8 is slightly lower, the variance is less pronounced compared to a batch size of 1, resulting in a more favorable setup.

For Gemma, batch sizes of 8 or 64 are preferable, as these yield the highest average F1 and recall score for the quality classification task (and the standard deviations are among the lowest). Although a batch size of 8 yields satisfactory results for Llama and Gemma, this is not the case for DeepSeek, which benefits from a larger batch size. Furthermore, the trends visible in the F1 scores vary extensively between the three different models. Interestingly, for all models, a batch size of 1 produces the largest standard deviation, suggesting that this setting may not be the most suitable choice, even though it is a commonly used default setting by researchers. These observations lead to our first key finding.

Finding 1 (RQ1). Each large language model follows a unique performance trend across varying batch size settings, ranging from consistently increasing or decreasing performance over larger batch sizes to patterns where performance initially improves and subsequently declines after a specific threshold. Therefore, it is advisable to select[7] the batch size based on the specific characteristics of the target dataset and model in mind.

In Section IV-C, we paid special attention to the classification of quality requirements, given the wide variability between models and the generally poorer results compared to the functional classification problem (Section IV-A). As shown in Tables VI VII, Gemma and DeepSeek exhibit a large gap between recall and precision, while Llama offers a better balanced recall and precision, even though its precision is the lowest for all batch sizes. In fact, it demonstrates the least variation between batch size settings in terms of recall, as shown by the total standard deviation. However, the specificity score indicates that Llama struggles to accurately identify the negative cases (i.e., non-functional and non-quality). Although the positive class are of most relevance to our classification problem, a low specificity score is suboptimal.

The ensemble model yields a more balanced trade-off between recall, precision, and specificity, as reflected in the F1 score (Table V). However, there are still significant variations between batch sizes, demonstrating a need for adjusting this setting to the specific use case. For example, selecting a batch size of 1 limits recall to 0.55, while a batch size of 8 increases recall to 0.70 (with a precision of 0.88 and a specificity of 0.79). This leads to our second finding:

[7]Selecting a suitable batch size choice involves systematically adjusting its values to achieve better performance. We recommend using a validation dataset (separate from your final test dataset), as is common practice in the ML and DL domain.

TABLE IX: Ratio of misclassified requirements that posses both characteristics over all misclassified requirements, normalized by their frequency.

	DeepSeek								Gemma								Llama							
Size	Functional				Quality				Functional				Quality				Functional				Quality			
	LD	OA	PM	ER	LD	OA	PM	ER	LD	OA	PM	ER	LD	OA	PM	ER	LD	OA	PM	ER	LD	OA	PM	ER
1	0.29	0.40	1.03	0.09	1.96	2.67	1.75	1.66	0.00	0.00	0.28	0.00	2.09	2.29	0.99	1.64	0.00	0.00	0.11	0.00	0.00	1.39	0.44	0.27
2	0.00	0.00	0.67	0.04	1.80	2.72	2.19	1.66	0.00	0.00	0.30	0.09	2.33	2.58	1.38	1.71	0.17	0.00	0.24	0.00	0.66	1.14	1.03	0.26
4	0.00	0.00	0.74	0.09	2.04	2.63	2.01	1.51	0.00	0.15	0.28	0.04	1.96	2.42	1.16	1.54	0.00	0.11	0.48	0.00	0.20	1.32	0.76	0.36
8	0.00	0.00	0.49	0.00	1.73	2.87	1.77	1.43	0.00	0.00	0.43	0.04	1.61	2.57	1.51	1.38	0.00	0.00	0.23	0.00	0.00	1.37	1.06	0.21
16	0.00	0.68	0.86	0.00	2.29	2.86	2.12	1.35	0.00	0.00	0.43	0.09	1.88	2.02	2.01	1.37	0.18	0.00	0.38	0.00	0.00	1.00	1.27	0.00
32	0.00	0.37	1.50	0.00	1.66	1.94	2.06	1.27	0.00	0.00	0.28	0.15	1.69	1.89	2.30	1.47	0.22	0.00	0.20	0.00	0.18	0.54	0.79	0.00
64	0.00	0.12	1.39	0.09	1.04	2.05	2.58	0.88	0.00	0.00	0.59	0.31	1.71	1.66	2.38	1.09	0.33	0.19	0.87	0.04	0.20	0.64	0.23	0.26
mean	0.04	0.22	0.95	0.04	1.79	2.53	2.07	1.39	0.00	0.02	0.37	0.10	1.90	2.20	1.68	1.46	0.13	0.04	0.36	0.01	0.18	1.06	0.80	0.19
SD	0.11	0.26	0.38	0.05	0.39	0.38	0.28	0.27	0.00	0.06	0.12	0.10	0.25	0.35	0.56	0.21	0.13	0.08	0.26	0.02	0.24	0.35	0.36	0.14

Finding 2 (RQ2). An ensemble model could offer a balanced trade-off between recall, precision, and specificity. Still, performance discrepancies may persist among varying batch sizes. Therefore, carefully adjusting the batch size remains essential even when applying an ensemble architecture.

B. Qualitative findings

To better understand the errors of the models, we study how having both a functional and quality aspect in a requirement influences the performance of the models. Table IX shows the ratio of incorrectly classified requirements by the model that contain both a functional and a quality aspect, normalized by the ratio of requirements with this property to the entire dataset. A value above 1 indicates that the model experiences more difficulty with these types of requirements, while a value below 1 suggests that the model can handle such requirements well. We show these ratios per batch size and per dataset. The ratios with a value higher than 1 are highlighted in gray.

As evident from the average values, having both a functional and a quality aspect in a requirement appears to influence negatively the classification of the quality aspect. For Gemma and DeepSeek, a substantial proportion of the incorrectly classified quality requirements involved requirements possessing both a functionality and a quality. This suggests that the models struggle to identify a quality when the requirement also includes a functional aspect. Notably, Llama behaves differently from DeepSeek and Gemma in this regard, as it appears to have less difficulty classifying these cases. Furthermore, the batch size settings that result in the most complications in predicting these cases vary depending on the context.

Conversely, recognizing the functional aspect does not pose a problem for the models, regardless of whether it also includes a quality aspect (as shown by the averages).

In short, these observations demonstrate the challenge of classifying quality requirements using LLMs. This finding is in line with prior research in which we examined the requirements within an issue tracking system and discovered that identifying non-functional requirements is more complex than classifying user-oriented functional requirements [22]. Our current findings show that this complexity is particularly apparent when a requirement relates to both a function and a quality, as the LLM tends to recognize the function, while failing to detect the quality. One potential solution would be to include more few-shot examples with requirements that contain both characteristics, enabling the model to recognize this pattern in new cases:

Finding 3 (RQ3). When classifying quality requirements, LLMs struggle to identify quality aspects when the requirements in question also include a functional aspect. It is therefore advisable to pay special attention to few-shot cases with both characteristics, providing the LLM with more context to recognize these patterns in new examples.

Table X shows the accuracy split per dataset-model pair, reporting both the macro-average and the range (i.e., difference between the minimum and maximum value) across the different batch sizes. The gray cells show the dataset with the highest accuracy score achieved by each model.

TABLE X: Macro-average of accuracy and standard deviation for each dataset.

(a) Functional Requirements

Model		Dataset			
		LD	OA	PM	ER
DeepSeek	Mean	0.74	0.81	0.81	0.73
	Range	0.10	0.15	0.06	0.00
Gemma	Mean	0.78	0.73	0.78	0.71
	Range	0.08	0.09	0.02	0.03
Llama	Mean	0.68	0.67	0.71	0.71
	Range	0.09	0.07	0.07	0.01

(b) Quality Requirements

Model		Dataset			
		LD	OA	PM	ER
DeepSeek	Mean	0.61	0.69	0.83	0.56
	Range	0.09	0.12	0.12	0.18
Gemma	Mean	0.64	0.73	0.8	0.62
	Range	0.17	0.10	0.14	0.13
Llama	Mean	0.73	0.68	0.78	0.69
	Range	0.14	0.23	0.13	0.10

The PROMISE dataset shows the highest accuracy for classifying quality and functional requirements consistent across models. For DeepSeek, the difference with the second highest

accuracy (from dataset OAppT) is noticeable, with an average gap of 0.14 (shown in Table X-b). Although the accuracy difference for classifying functional requirements is less substantial, the quality classification problem is considered the most challenging by all models and is therefore the most interesting in this respect.

Finding 4 (RQ4). The requirements in the PROMISE dataset are consistently considered the easiest to classify by all models. Despite its popularity in the RE community, this result emphasizes the importance of incorporating more diverse datasets for a comprehensive evaluation.

The range represents the variability between batch sizes. Although the PROMISE dataset may be easier to classify, the performance variability between batch sizes does not reduce. This demonstrates the importance of considering the batch size setting, as even in a simpler dataset this parameter can still have a significant influence. This leads to our final finding:

Finding 5 (RQ4). Although the particular dataset may influence the average performance per model, the dataset does not have a positive or negative effect on the variation between different batch size settings. This finding further emphasizes the importance of investigating the batch size setting in classification problems when using decoder-only models.

VI. Threats to Validity

We adopt the categories defined by Wohlin *et al.* [23] to organize the threats to the validity of this study.

Conclusion validity: We compared our models using widely accepted evaluation metrics in software engineering research [16]. However, due to the limited number of datasets in our experiments, we did not conduct statistical significance tests, as the small sample size would undermine their reliability. Future studies should incorporate a higher number of datasets to enable statistical analysis of model performance.

External validity: Given the wide variety of LLMs currently available, there is a risk of sample bias. To limit this bias, we examined LLMs which have been used extensively over the past few years. In the experiments, we used three LLMs: Gemma, DeepSeek-Qwen, and Llama. We initially conducted experiments with Mistral; however, despite exploring larger models, the generated responses failed to comply with the template structure, making automation impractical. As a result, we removed Mistral from the experiments.

Another potential threat to validity could be related to the selected datasets. It is possible that the results would have been different if we had selected other datasets. To mitigate this risk, we used multiple datasets, one of which is widely used for classifying functional and quality requirements (PROMISE). In addition, we used a sample from each dataset, which could pose a threat. We decided to limit the sample size to 192 to reduce resource consumption, although we acknowledge that this may be a limitation.

Internal validity: In our experiments, we selected specific model sizes for each LLM architecture based on practical constraints. We acknowledge that model size is a potential confounding factor, as larger variants might exhibit different performance characteristics. However, our objective was to evaluate models that are feasible to deploy under limited resource conditions, making smaller variants more relevant for our study. Future work could explore whether the findings generalize across different model sizes within the same architecture.

Furthermore, as discussed by Dong *et al.* [21], the choice of few-shot examples can have a significant impact on the performance of LLMs. We opted for a manual approach to ensure that a wide range of scenarios was represented. However, it is plausible that a different selection of few-shot examples would have led to different performance results.

As mentioned in [14], when using an ensemble learning method, the aim is to select so-called 'weak classifiers' so that each model focuses on explaining a different part of the data. The LLMs in our selection cannot be considered weak classifiers. We could have opted for a smaller model, but this would have limited the automation of post-processing, as these models were unable to follow a predefined template. It is possible that smaller models may lead to a better performing ensemble model. In addition, we selected majority voting as the voting mechanism. We acknowledge that a more sophisticated ensemble strategy could have led to better results.

Moreover, we did not tune the temperature values (or other hyperparameters) of the LLMs, while a different temperature setting could have influenced the performance of the models. However, for the sake of reproducibility, we opted for a temperature of 0.01.

Construct validity: A key threat to construct validity is whether the chosen evaluation metrics truly reflect model performance. To address this, we used precision, recall, and F1 score—metrics widely adopted in software engineering research for assessing automated classifiers [16]. We also included specificity to account for the model's ability to correctly identify negative cases, offering a more balanced view of performance.

VII. Related Work

Datasets: PROMISE NFR [24] is a dataset of 15 projects aiming to demonstrate various categories of non-functional requirements. The original dataset has been received attention from the community as a benchmark for classifying requirements [25], [26], [27]. Dalpiaz *et al.* [12] re-annotate the original PROMISE dataset based on the theory by Li *et al.* [13]. Hey *et al.* [28] use both the original and re-annotated versions of the dataset for different classification tasks, and further annotate the functional requirements identified by Dalpiaz *et al.* [12] into function, data, and behavior labels. In addition to the re-annotation of PROMISE NFR, Dalpiaz *et al.* annotate four publicly available datasets [29], [30], [31], [32] and three private datasets of functional and non-functional requirements.

Dell'Anna *et al.* [16] introduces six new annotated datasets in addition to those provided by Dalpiaz *et al.* [12].

Shallow machine learning techniques: Cleland-Huang *et al.* [33], [2] use word frequency to classify requirements sentences into functional and non-functional requirements, and then into specific qualities such as security. Hussain *et al.* [25] employ decision trees for binary classification of functional and non-functional requirements. Kurtanovic and Maalej [26] rely on support vector machines as the classification method and n-grams for the features. Abad *et al.* [27] also use decision tree algorithm to classify requirements, building upon features identified by Hussain *et al.* [25]. All these studies only use the PROMISE NFR dataset and features tailored for it. Identifying this gap, Dalpiaz *et al.* classify requirements with support vector machines with linguistic features to move away from word-level features and thus increase the generality of the trained models. They also experiment with multiple datasets. Later, Dell'Anna *et al.* [16] adopts a similar approach but expands the datasets used for the experiments. They also publish guidelines for comparing classifier models, encouraging project-fold validation, reporting statistical significance and effect size, and checking generality and degradation of the trained models.

Transformer Models: Hey *et al.* [28] tackles various classification tasks including functional vs non-functional requirements with BERT-based transformer models. Although the authors report that their approach dominates the state-of-the-art, this claim is not supported with statistical tests. Later, Dell'Anna *et al.* [16] demonstrates that this is not the case for all experimental settings and classification tasks with their replication case study and detailed statistical analyses.

Large Language Models: Ronanki *et al.* [34] study the performance of various prompting patterns on requirements classification and traceability problems. They experiment with the PROMISE NFR dataset and GPT-3.5 model. Zadenoori *et al.* [6] propose a method to optimize prompts and demonstrate their approach on the requirements classification task, using Meta-Llama-3-8B-Instruct[8] on the re-annotated PROMISE NFR dataset by Dalpiaz *et al.* [12]. Karlsson *et al.* [35] compare the reliability of GPT-4o and LLAMA3.3-70B using a zero-shot learning approach on the original PROMISE NFR dataset and find LLAMA3.3-70B more consistent.

VIII. Conclusions

In this study, we assessed the impact of the selected batch size on the performance of a LLM prompted to classify requirements. We define batch size as the number of requirements to be classified in a prompt sequence. To answer our research questions, we conducted experiments with three state-of-the-art LLMs: Gemma 3 12B, DeepSeek-R1 Distill-Qwen 14B, Llama 3.1 8B-Instruct. We performed the experiments on samples from four existing datasets [12], [16].

Our findings show that each large language model (Finding 1) and the ensemble version (Finding 2) follow a unique performance trend across different batch size settings. Therefore, we recommend adjusting the batch size whenever the dataset and model are changed, in order to minimize the negative effect of the batch size setting.

Furthermore, in line with previous work [16], [22], classifying quality requirements with LLMs appears to be more challenging than classifying functional requirements. The models struggled particularly with identifying a quality aspect when the requirement also describes a functionality (Finding 3).

Finally, out of all the datasets, the PROMISE dataset stood out as the easiest to classify, suggesting that this benchmark dataset may not be representative of real-world requirements (Finding 4). This emphasizes the need for further research in the field of RE to focus on additional datasets in order to properly evaluate classifiers.

Future research should focus on expanding our analysis by integrating additional datasets and models to verify the generalizability of our study. In addition, it would be relevant to extend the research to more classification problems, such as classifying quality requirements into specific types.

[8]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct.

Acknowledgment

This research is partially funded by the Dutch Research Council (NWO) through the Open Technology Programme 2021-II TTW, project AUTOLINK (19521). The execution of the experiments was facilitated by a Small Compute grant from NWO (EINF-12939).

References

[1] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, and M. Traynor, "Automated demarcation of requirements in textual specifications: a machine learning-based approach," *Empirical Software Engineering*, vol. 25, pp. 5454–5497, 2020.

[2] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements Engineering*, vol. 12, pp. 103–120, 2007.

[3] M. Binkhonain and L. Zhao, "A review of machine learning algorithms for identification and classification of non-functional requirements," *Expert Systems with Applications: X*, vol. 1, p. 100001, 2019.

[4] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *IEEE International Requirements Engineering Conference (RE)*. IEEE, 2015, pp. 116–125.

[5] K. Ronanki, B. Cabrero-Daniel, J. Horkoff, and C. Berger, "Requirements engineering using Generative AI: Prompts and prompting patterns," in *Generative AI for Effective Software Development*, A. Nguyen-Duc, P. Abrahamsson, and F. Khomh, Eds. Cham: Springer Nature Switzerland, 2024, pp. 109–127.

[6] M. A. Zadenoori, L. Zhao, W. Alhoshan, and A. Ferrari, "Automatic prompt engineering: The case of requirements classification," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2025, pp. 217–225.

[7] Q. Motger, A. Miaschi, F. Dell'Orletta, X. Franch, and J. Marco, "Leveraging encoder-only large language models for mobile app review feature extraction," *Empirical Software Engineering*, vol. 30, no. 3, p. 104, 2025.

[8] A. El-Hajjami, N. Fafin, and C. Salinesi, "Which AI technique is better to classify requirements? an experiment with SVM, LSTM and ChatGPT," in *Joint Proceedings of REFSQ Workshops, Doctoral Symposium, Posters Tools Track, and Education and Training Track*, 2024.

[9] W. Alhoshan, A. Ferrari, and L. Zhao, "How effective are generative large language models in performing requirements classification?" *arXiv preprint arXiv:2504.16768*, 2025.

[10] C. Ling, X. Zhao, J. Lu, C. Deng, C. Zheng, J. Wang *et al.*, "Beyond one-model-fits-all: A survey of domain specialization for large language models," *arXiv preprint arXiv*, vol. 2305, 2023.

[11] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *arXiv preprint arXiv:2307.03172*, 2023.

[12] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *IEEE International Requirements Engineering Conference*. IEEE, 2019, pp. 142–152.

[13] F.-L. Li, J. Horkoff, J. Mylopoulos, R. S. Guizzardi, G. Guizzardi, A. Borgida, and L. Liu, "Non-functional requirements as qualities, with a spice of ontology," in *IEEE International Requirements Engineering Conference*, 2014, pp. 293–302.

[14] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[15] A. T. van Can and F. Dalpiaz, "Requirements information in backlog items: Content analysis," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2024, pp. 305–321.

[16] D. Dell'Anna, F. B. Aydemir, and F. Dalpiaz, "Evaluating classifiers in SE research: the ECSER pipeline and two replication studies," *Empirical Software Engineering*, vol. 28, no. 1, p. 3, 2023.

[17] A. T. van Can, F. B. Aydemir, and F. Dalpiaz, "Online appendix of the paper: 'One Size Does Not Fit All: On the Role of Batch Size in Classifying Requirements with LLMs'," 2025. [Online]. Available: https://doi.org/10.5281/zenodo.15982515

[18] A. Vogelsang and J. Fischbach, "Using large language models for natural language processing tasks in requirements engineering: A systematic guideline," in *Handbook on Natural Language Processing for Requirements Engineering*. Springer, 2025, pp. 435–456.

[19] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with ChatGPT," *arXiv preprint arXiv:2302.11382*, 2023.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," in *International Conference on Neural Information Processing Systems*, 2020, pp. 1877–1901.

[21] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu *et al.*, "A survey on in-context learning," in *Conference on Empirical Methods in Natural Language Processing*, 2024.

[22] A. T. van Can and F. Dalpiaz, "Locating requirements in backlog items: Content analysis and experiments with large language models," *Information and Software Technology*, vol. 179, p. 107644, 2025.

[23] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén *et al.*, *Experimentation in Software Engineering*. Springer, 2012.

[24] The PROMISE NFR repository of empirical software engineering data. [accessed 9-June-2025]. [Online]. Available: https://zenodo.org/records/268542

[25] I. Hussain, L. Kosseim, and O. Ormandjieva, "Using linguistic knowledge to classify non-functional requirements in SRS documents," in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2008, pp. 287–298.

[26] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *IEEE International Requirements Engineering Conference*, 2017, pp. 490–495.

[27] Z. S. H. Abad, O. Karras, P. Ghazi, M. Glinz, G. Ruhe, and K. Schneider, "What works better? a study of classifying requirements," in *IEEE International Requirements Engineering Conference*. IEEE, 2017, pp. 496–501.

[28] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *IEEE International Requirements Engineering Conference*. IEEE, 2020, pp. 169–179.

[29] ReqView Example Requirements. https://www.reqview.com/doc/example-requirements-documents.html. [Online; accessed 8-April-2019].

[30] Leeds University Library Requirements. https://leedsunilibrary.files.wordpress.com/2013/06/repositoryfunctionalrequirementsv1-1_web_1_.xlsx. [Online; accessed 8-April-2019].

[31] Web Architectures for Services Platforms (WASP) Requirements. https://www.zenodo.org/record/581655. [Online; accessed 8-April-2019].

[32] J. Cleland-Huang, M. Vierhauser, and S. Bayley, "Dronology: An incubator for cyber-physical systems research," in *International Conference on Software Engineering: NIER Track*, 2018, pp. 109–112.

[33] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *IEEE International Requirements Engineering Conference*. IEEE, 2006, pp. 39–48.

[34] K. Ronanki, C. Berger, and J. Horkoff, "Investigating chatgpt's potential to assist in requirements elicitation processes," in *Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2023, pp. 354–361.

[35] F. Karlsson, P. Chatzipetrou, S. Gao, and T. E. Havstorm, "How reliable are GPT-4o and LLAMA3.3-70B in classifying natural language requirements?" *IEEE Software*, 2025.