Responsible and Sustainable AI: Considering Energy Consumption in Automated Text Classification Evaluation Tasks

Angelika Kaplan
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
angelika.kaplan@kit.edu

Jan Keim
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
jan.keim@kit.edu

Lukas Greiner
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
lukas.greiner@alumni.kit.edu

Ralf Sieger
FZI Research Center f. Information Tech.
Karlsruhe, Germany
sieger@fzi.de

Raffaela Mirandola
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
raffaela.mirandola@kit.edu

Ralf Reussner
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
ralf.reussner@kit.edu

***Abstract*—Text classification is one of the typical and fundamental natural language processing tasks. With the advent of large language models (LLMs), text classification has evolved much further. Based on the growing sizes of LLMs and the increased demands for hardware, and especially energy, questions about sustainability and environmental impacts and responsibility also arise. To assess text classification approaches, researchers usually only use common performance metrics like precision, recall, and f_1-score. Green AI, i.e., improving environmental aspects while maintaining performance, is regularly disregarded and not a standard in the evaluation of automated text classification approaches. Yet, minor performance improvements might not justify, e.g., much higher energy consumption. In this paper, we aim to raise awareness for this issue and the corresponding trade-off discussions and decisions. Therefore, we present novel sustainability metrics and provide guidelines for text classification approaches that are suitable for Green AI. In a text classification use case, we showcase the applicability of our proposed metrics and discuss corresponding trade-off decisions.**

***Index Terms*—Green AI, sustainability metrics, energy consumption, natural language processing, text classification.**

I. Introduction

Text classification is a fundamental task in natural language processing (NLP) that involves categorizing text into predefined labels or classes based on its content. This task is essential for many applications, such as spam detection, sentiment analysis, topic labeling, language identification, and document organization. As such, text classification drives advancements in automated information retrieval, content moderation, and user experience personalization. In software engineering, common examples include requirements classification [1]–[4], bug and issue classification [5], [6], and commit message classification [7]. Moreover, in meta-research, text classification is used to classify papers that aim to contribute to research data management and scholarly communication [8]. In industry, this is underlined by the increasing usage, adoption, or exploration of Artificial Intelligence (AI), in particular, the adoption of NLP technologies (cf. IBM Global AI Adoption Index Report 2023 [9]).

Text classification has evolved significantly with the advent of deep learning and transfer learning. Models like BERT [10], RoBERTa [11], and the GPT models [12]–[14] have set new benchmarks by leveraging large-scale pre-training on diverse corpora followed by fine-tuning on specific tasks. These models can capture language patterns and context, enabling more accurate and robust text classification.

What is most of the time omitted in text classification studies is energy efficiency and carbon footprint [15], [16]. Sometimes, a new approach provides minimal classification performance increase, but might require much more energy. For example, an f_1-score increase of 1% might increase energy consumption by 50%, causing much higher financial and environmental costs when applying the approach to a big dataset in large-scale use. An increased carbon emission (CO_2e) of only 10 mg per classification results in 1 kg CO_2e already after 100,000 classifications. Discussion (if any) about energy efficiency and carbon footprint is regarded in isolation, and the classification performance trade-off is usually missing [17], [18]. For a comprehensive assessment of the environmental impact of AI systems (see also Green AI [19]), it is vital to document and report these metrics transparently. By doing so, the research community can work towards reducing the environmental impact of AI and, thus, fostering the development of more sustainable technologies that support core principles in the Karlskrona Manifesto [20]. However, there are no guidelines for suitable metrics that can be used to support such a discussion.

Consequently, we address the following research question: *What are suitable metrics for (automated) text classification tasks, especially regarding performance and energy efficiency?*

Contribution. In this paper, we provide an overview of evaluation metrics for automated text classification approaches.

We consider established performance metrics and propose novel sustainability metrics regarding performance and energy efficiency to enable trade-off discussions and decisions. We apply our proposed metrics in a use case example [21], i.e., by automating a multi-label classification problem of abstracts in software architecture research using 13 dedicated classes for determining the *Evaluation Method(s)* used. In addition, we provide our artifacts in an open-access repository [22].

Outline. Next, in Section II, we provide foundations w.r.t. automated text classification and possibilities of emission tracking. We look into related work in Section III. We introduce performance metrics and propose sustainability metrics for the respective classification experiments in Section IV and discuss the limitations of the proposed metrics. In Section V, we describe best practices for assessing carbon emissions and energy consumption. We present our use case example in Section VI and discuss the results based on our proposed metrics before we conclude in Section VII.

II. Foundations

This section introduces the basic building blocks required for our metrics specification (cf. Section IV) and the operational steps for applying them.

A. Classification Approaches

There are various approaches for text classification, with many approaches based on supervised machine learning (ML) [23]–[25]. These approaches require labeled training data to learn from these data. Classic approaches include Support Vector Machines, Naïve-Bayes, or Linear Regression. Modern approaches are often based on artificial neural networks (ANNs), such as Recurrent Neural Networks (RNNs) and Long-Short-Term Memory (LSTMs). ANN-based approaches are powerful, but require lots of training samples.

B. Large Language Models

Large Language Models (LLMs) are huge ANNs that have revolutionized NLP. LLMs typically use the Transformer architecture by Vaswani et al. [26] that contains a self-attention mechanism. This mechanism allows models to capture long-range dependencies within text efficiently. The Transformer architecture enables LLMs to handle large-scale textual data and perform a wide range of language tasks on a high-performance level, from translation to text generation, including text classification, due to their self-supervised training on huge amounts of data.

For classification tasks, LLMs can either be *fine-tuned* or used via *prompt engineering*. In *fine-tuning*, labeled data is used to continue to train the pre-trained LLM to improve its capabilities for a given task. *Prompt engineering* describes the process of improving prompts for LLMs to improve the output. Typically, there are zero-shot prompts (no example provided) and few-shot prompts (some examples provided). Which method works best is usually based on the amount of training data [27]. If no labeled data exists, prompt engineering with zero-shot prompting is recommended. If only a few labeled data exist, few-shot prompting is preferred. Fine-tuning is best used if lots of labeled data exist. There are also further prompting techniques and tricks that can influence an LLM's response, like asking the LLM to provide a chain of thought.

C. Experiment Management Tools

Experiment Management Tools in ML support researchers and practitioners when evaluating and building AI software systems [28]. These tools are usually available as cloud-based platforms (e.g., Microsoft Azure ML) or standalone software tools (e.g., MLflow). In contrast to traditional software engineering, where assets can be managed via version control systems, the management of ML systems is much more complicated. Such assets in ML systems encompass *resource artifacts* (e.g. datasets), *software artifacts* (e.g., source code and hyperparameters), and *metadata* (e.g., experiment and execution metadata, performance metrics) [28]. Consequently, these tools have to address issues such as versioning, traceability, auditability, explainability, interpretability, collaboration, and reproducibility, which are critical for assessing and maintaining the integrity of ML projects. To inspect the results of the different experiment runs, these tools have to provide an efficient retrieval option as well [28]. An overview of the best experiment tracking tools in 2024 is provided by [29]. However, scientific literature indicates that some factors affect the tools' maturity, such as lack of interoperability across different tools (i.e., bridging the gap between development and production environments) and lack of explicit representation of domain knowledge [30]. For our research objective, this also includes the handling and execution of carbon emission and energy consumption tracking (cf. Section II-D). As a result, researchers (and practitioners) have to determine a suitable tool for their purposes carefully.

D. Tracking of Emission and Energy Consumption

Besides measuring energy consumption directly from the hardware, there are different tools to measure or estimate the energy consumption and CO_2 emissions of text classification approaches. In the following, we provide an excerpt overview of the different tools and their license such as Python libraries and online browser apps while focussing on tools that are freely available and documented in a scientific publication.

Carbontracker (2020, MIT) is an open-source tool for tracking and predicting the energy consumption and carbon footprint of training deep learning models. The package provides an additional proactive approach to reduce carbon emissions through the utilization of predictions. The library supports various environments and platforms such as clusters, desktop computers, and Google Colab notebooks [31].

CodeCarbon (2020, MIT) is a Python package and the successor of ML CO2 Impact [32] for estimating and tracking carbon emissions by considering computing infrastructure, location, hardware, usage, and runtime [33].

Cumulator (2021, MIT) is a Python package that estimates the energy consumption of computation based on runtime, GPU load, and carbon intensity, with a fixed value for the consumption of a typical GPU in academia and healthcare [34].

eco2AI (2022, Apache 2.0) is a Python library considering system processes that are related directly to model training. This avoids overestimation [35].

Green Algorithms (2021, CC-BY-4.0) is a web tool to calculate the energy consumption and carbon footprint based on user-supplied information supplied by the user: runtime, number and types of cores, memory, type of platform used (PC, local server, cloud computing), and location [36].

LLMCarbon (2024, NN) is an end-to-end carbon footprint modeling tool for training, inference, experimentation, and storage processes to enable design space exploration considering the trade-off between carbon footprint and test loss [37].

III. Related Work

We present related work from the following research areas that deal with sustainability, esp. in terms of environmental impact (i.e., energy consumption and carbon emission):

Proposing Combined Metrics. Hasan et al. [38] introduce the Robustness Carbon Trade-off Index (RCTI), a novel metric that quantifies the trade-off between model robustness and carbon emissions in ML experiments. They demonstrate the RCTI through an evasion attack experiment and analyze the ensemble between robustness against attacks and carbon emissions. The authors later extend this metric by adding an economic dimension to the carbon-robustness trade-off, resulting in the so-called Cost Per Unit of Robustness Change (CRC) [39] and considering the triple interplay. However, we see problems with implicit, partly domain-specific assumptions and with potential divisions by 0, making them less applicable.

Positions towards sustainable trade-off discussions. In software engineering, sustainability encompasses the following five key dimensions: environmental, economic, technical, social, and individual [20], [40]. Each dimension can have different metrics and values. From a holistic perspective, we can identify studies that deal with sustainability trade-offs of AI-based software systems [17], [18]. We contribute to this holistic view by focusing on and interrelating two dimensions and proposing concrete guidelines and novel metrics to investigate and assess automated text classification as one of the fundamental tasks in NLP.

Investigating sustainability w.r.t. environmental impact. Several works regard sustainability aspects for NLP tasks, in terms of environmental impact and computational aspects. For example, Luccioni et al. [15] performed a study measuring the amount of energy and inference cost of various NLP tasks, including text classification, using representative benchmark datasets. However, in contrast to our objective, the discussion just focuses on energy consumption in isolation, disregarding performance aspects for each task.

IV. Metrics Specification

Selecting appropriate metrics for a classification task is of significant importance. Researchers have to justify the selection of metrics based on a specific scientific problem or experimental issue. In the case of supervised methodologies, a gold standard dataset (annotated by domain experts) is essential to evaluate the applicability of different classification approaches in a comparative way. Additionally, we recommend conducting several runs of each experiment to ensure the statistical significance of the measurements, particularly if the approach has built-in randomness.

In the following, we first introduce common performance metrics to evaluate classification approaches in a shared and comparable way. Second, we provide and define new formulas w.r.t. performance and energy consumption to enable discussions about trade-off decisions for sustainable and responsible AI while answering our research question (cf. Section I).

A. Performance Metrics

To assess performance, investigators can use the *confusion matrix* [41], which is a 2x2 matrix that contains the classification of results, labelled as True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). For multi-class and multi-label classification, the confusion matrix has consequently a bigger dimension than 2x2. The matrix can be interpreted or mathematically reduced to obtain TP, TN, FP, and FN for a specific class. Further, a multi-label classification problem can be interpreted as a multi-class classification problem during evaluation by creating distinct classes for the label combinations.

Common performance metrics for classification tasks include accuracy, precision, recall, and specificity, as well as the f_β-score (usually with $\beta = 1$):

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{4}$$

$$f_\beta = \left(1 + \beta^2\right) \times \frac{\text{precision} \times \text{recall}}{(\beta^2 \times \text{precision}) + \text{recall}} \tag{5}$$

$$f_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Together, these metrics provide a comprehensive evaluation of a model's performance across different aspects.

When evaluating multi-class or multi-label classifiers, metrics like f_1-score need to be calculated for each class. To combine and summarize the results, researchers usually average the scores. There are various variants for averaging. **Micro averaging** aggregates the contributions of all classes to compute the f_1-score, treating each instance equally, making it useful for evaluating overall performance in imbalanced datasets. **Macro averaging** calculates the f_1-score for each class independently and then takes the average, giving equal weight to all classes regardless of frequency. This can be useful to highlight performance disparities between classes. **Weighted averaging** also computes the f_1-score for each class independently, but then averages them according to the class frequencies. This shows the result that reflects the actual distribution of classes in the dataset. Each method has a purpose, and it is important to distinguish these purposes and to conclude accordingly.

Besides measuring the classifier's performance about its ability to predict classes correctly, researchers can also track the *runtime*. Runtimes are usually measured in (milli-) seconds. We differ between (i) training time, (ii) testing time, and (iii) execution time. Execution time here is the overall runtime of each experiment under the same conditions by adding training time and testing time.

B. Sustainability Metrics

One important aspect of Green AI is to evaluate sustainability, specifically the environmental footprint. The training and inference processes consume significant amounts of electricity, particularly when using LLMs on GPUs.

Energy consumption can be measured in kilowatt-hours (kWh) or megajoules (MJ), where $1\text{kWh} = 3.6\text{MJ}$ (see International System of Units (SI)[1]).

Electricity consumption has an environmental impact, especially when it is not entirely generated from renewable energy sources such as gas, oil, or nuclear power. This impact is referred to as the carbon footprint, encompassing greenhouse gas (GHG) emissions such as carbon dioxide (CO_2), methane (CH_4), nitrous oxide (NO_2), and fluorinated gases. Emissions are typically measured in CO_2 equivalents (kg CO_2e) [42].

Carbon emissions can be quantified [32]. To quantify carbon emissions, researchers need to estimate or measure the energy consumption and convert this into CO_2 equivalents, considering the local energy mix. This location-dependent conversion relies on the proportion of energy derived from, e.g., solar and nuclear power or fossil fuel power plants.

Maintaining a stable setup is essential for the accurate measurement of both energy consumption and carbon emissions. Researchers must use the same machine at the same location for their experiments to ensure comparable values. Different machines can vary in energy efficiency, impacting energy consumption. Similarly, different locations can have varying energy mixes, affecting the carbon footprint [32].

Documenting and reporting these metrics transparently is essential for a comprehensive assessment of an AI system's environmental impact. By doing so, researchers can work towards reducing the environmental impact of AI and fostering the development of more sustainable technologies [20].

Combined Metrics: Following, we define our novel metrics. In the first place, researchers can use the *total carbon emission (CE)* to evaluate the environmental footprint.

To relate the carbon emissions to their classification performance (e.g., the macro averaged f_1-score), Equation 7 defines the metric for *relative carbon emission (CE$_{rel}$)*. It presents the caused carbon emission per achieved f_1-score, with lower values being better.

$$\text{CE}_{\text{rel}} = \frac{\text{CE}}{f_1} \tag{7}$$

Researchers can use a *delta-based version* when comparing different classifiers, as shown in Equation 8. To calculate the delta, researchers need to select the worst-performing classifier based on macro averaged f_1-score as the baseline classifier. Accordingly, $f_{1\text{base}}$ is the worst f_1-score among the classifiers and CE_{base} is its corresponding carbon emission. This metric combines the difference in f_1-score with the relation of the baseline's carbon emission compared to the carbon emission of the observed approach. When interpreting the resulting values, bigger values are better because the value increases with better f_1-score, better carbon emissions, or both.

$$\Delta_{\text{CE}} = (f_1 - f_{1\text{base}}) \times \frac{\text{CE}_{base}}{\text{CE}} \tag{8}$$

In addition to the delta-based version, researchers can use a *normalized version* for assessing different classifiers to simplify the comparison. The *normalized carbon emission (n(CE))* in Equation 9 scales the carbon emission between 0 and 1 based on the lowest ($\text{CE}_{\text{lowest}}$) and highest carbon emission ($\text{CE}_{\text{highest}}$) in the evaluation setting. The lowest emitting approach is assigned a 0, the highest emitting approach a 1, and the other approaches are scaled accordingly.

$$\text{n(CE)} = 1 - \frac{\text{CE} - \text{CE}_{\text{lowest}}}{\text{CE}_{\text{highest}} - \text{CE}_{\text{lowest}}} \tag{9}$$

The same calculation can be done with CE_{rel}:

$$\text{n}(\text{CE}_{\text{rel}}) = 1 - \frac{\text{CE}_{\text{rel}} - \text{CE}_{\text{rel}_{\text{lowest}}}}{\text{CE}_{\text{rel}_{\text{highest}}} - \text{CE}_{\text{rel}_{\text{lowest}}}} \tag{10}$$

In the same fashion, investigators can easily calculate further (normalized) comparison metrics, if necessary.

While the above metrics look at carbon emissions, energy consumption (in kWh or MJ) can be assessed equally. Similarly, these metrics focus on f_1-score, but the classification performance can also be exchanged to another preferred metric like f_2-score, precision, recall, or specificity. Moreover, unlike related work, these metrics are designed to avoid pitfalls that make comparisons impractical, like accidental division by 0.

Evaluation from different perspectives and across phases is essential to comprehensively assess these metrics. Specifically, energy consumption should be captured and evaluated separately for both the *training* and *inference* phases of classifiers.

Training typically involves intensive computational processes, leading to substantial energy consumption The energy consumption profile shifts as the classifier transitions into the inference phase. Over time, and with an increasing number of classifications, inference becomes the dominant factor in overall energy usage. Distinguishing these phases is needed to assess the environmental impact more accurately.

Moreover, new data or insights can necessitate updates to the classifier. This process introduces additional energy

[1] https://www.iec.ch/si, Last acc. 2024-11-11

demands, whether through *retraining* or continual learning. These demands must be accounted for to comprehensively evaluate the energy efficiency and environmental footprint.

In summary, a thorough evaluation that differentiates between training and inference phases and also considers updates is crucial to accurately assess the environmental impact of text classification approaches.

C. Limitations

It is important to note that all metrics can only be compared within the same setting for the selected approaches and their results. Due to the intricacies of measuring energy consumption and carbon emission (see also Section V), different setups will likely result in different (absolute) results. These measurements allow the comparison of results within one setup to select the most suitable approach but only allow limited comparisons between, e.g., different researchers that independently measure with different setups.

V. Best Practices for Assessing Energy Consumption

A certain care is necessary when assessing energy consumption and carbon emission. Just like other performance measurements, the setting needs to be equal for all approaches; otherwise the results cannot be compared. In general, the investigator needs to conduct the same classification task while keeping most parameters stable. This way, the investigator can track these metrics in each configuration run while searching for a suitable automated classification approach considering performance and carbon emissions. In the following, we have several best practices to support investigators:

a) Use the same hardware and platform: To achieve comparable results, it is important to use the exact same hardware and the same platform (e.g., operating system, etc.). Other hardware or platforms might have different properties and perform different energy-wise. More efficient hardware can reduce energy consumption significantly, but we are interested in measuring and comparing our software. This also means that changing programming languages makes the comparison harder, as different programming languages have different efficiency, as Pereira et al. showed [43].

b) Resources should focus only on the task: The hardware should not be concerned with other tasks, i.e., it should be idle except for the classifier that should be investigated and measured. Otherwise, the *noise* might influence the measurements, effectively rendering the results worthless.

c) Perform the experiments at roughly the same time: To avoid changes in the local energy mix or other influencing factors like platform updates, the experiments should be performed at roughly the same time.

d) Use the same task and evaluation setting: To be able to compare two or more approaches, investigators need to evaluate the same task using the same evaluation setting on the same dataset. While this sounds obvious, some parts can be overlooked. This includes using the same splits, e.g., for training and evaluation data to avoid having a split that requires slightly more energy due to, e.g., more contained text. If investigators do k-fold cross-validation, the splits also need to be the same, preferably in the same order to avoid fluctuations.

e) Use the same tooling: Different tooling might influence the evaluation due to different margins of error or the overhead these tools introduce for logging or capturing energy consumption, etc. Different programming languages also usually require different tooling, which is another reason to select only one programming language.

f) Measure and distinguish the different phases: A machine learning-based classifier usually has at least a training and an inference phase. The energy consumption for training might differ a lot compared to the inference. For example, some approaches might perform better in training but worse in inference. While training is often a one-time effort (except in the case of retraining), the inference is constantly needed when deploying the approach. Measuring and reporting energy consumption for both phases is advised to facilitate comparisons and provide a sound basis for decisions.

g) Repeat experiments: We advise performing several runs to reduce influences by variance. This also improves the stability and statistical significance of the measurements.

h) Report the results: Written text (in papers) and appropriate visualization techniques are useful tools, to communicate and interpret research results obtained out of applied metrics. We recommend, e.g., tables to list the resulting values or grid reference systems to better contrast both aspects (i.e, performance and carbon emission by, e.g., illustrating results w.r.t. relative carbon emission ($\mathrm{CE}_{\mathrm{rel}}$)).

VI. Use Case Example

In the following section, we introduce the use case example, including the automated classification approaches to show and underline the application of our proposed metrics in Section IV to enable trade-off discussion and decision w.r.t. performance and energy consumption.

a) Experimental Setup: Our experiments run on a server (Debian GNU/Linux 12) equipped with 2x Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz (112 parallel threads), 252 GB RAM, and a Tesla V100S-32GB. We use CodeCarbon [44] (version 2.3.4) for emission tracking and Weights & Biases (W&B) [45] as experiment management tool.

b) Dataset: For our classification task, we used the gold standard dataset from the literature review of Konersmann et al. [21] that aimed to provide an overview of the current state of practice in evaluating software architecture research [46]. In this review, the authors investigated, among other things, how research objects in 153 papers in software architecture (i.e., full papers published at ICSA (International Conference on Software Architecture) and ECSA (European Conference on Software Architecture) between 2017 and 2021) are investigated. The dataset contains the abstracts of these 153 papers along with the classification, where each paper can have multiple labels of each category (i.e., *Research Object*, *Evaluation Method*, and *Property*).

In our use case, we wanted to classify papers according to the used *Evaluation Methods* (e.g., Case Study or Technical Experiment) in the dataset. The category has 13 disjoint classes [21], and a paper has a maximum of three *Evaluation Methods*. The dataset in this category is highly imbalanced and shows missing representatives for training, as Figure 1 depicts.

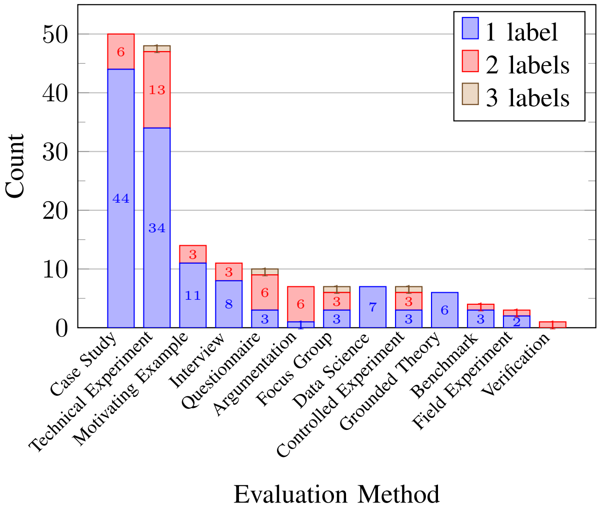

Fig. 1. Stacked bar plot with distribution of the labels in the category *Evaluation Method* in the gold standard dataset [21]. The colors show how often a paper has 1, 2, or 3 ground truth labels and one belongs to the category.

c) Classification Task: For the multi-label classification tasks to classify a paper's abstract according to the used evaluation methods, we selected two kinds of settings, i.e., based on fine-tuning and on zero-shot prompting.

For fine-tuning, we selected five LLMs from the Hugging Face Transformers library, namely BERT (110M; 10/2018), DeBERTa (304M; 03/2023), DistilBERT (66M; 10/2019), SciBERT (110M; 03/2019), and XLNet (92M; 06/2019). To determine the suitable and optimal hyperparameters for the classification task, we performed a hyperparameter search for the seed for network initialization, the learning rate, the number of training epochs, the training batch size, and the weight decay for the AdamW Optimizer. For this, we used Bayesian optimization with the Optuna Framework by defining a dedicated search space for each parameter and optimized towards macro f_1-score. For the data sampling, we used a stratified split. Additionally, we also experiment with different data augmentation strategies, i.e., text perturbation via TextAttack[2] and multi-label-oversampling[3]. In this use case, we are only interested in measuring the carbon emission for the inference phase.

In a second experiment, we also used a prompt-based setting with selected LLMs, applying a zero-shot approach to predict the labels without model training. We followed common and general recommendations to construct the prompt. The prompt first contains basic information, i.e., a description of the labels (i.e., a definition of the respective evaluation methods) and the query to predict labels and to return the answer in JSON, followed by the abstracts' texts. For this family of experiments, we used Llama2 (7B, 13B, 70B; 08/2023), Mistral (7B; 10/2023), and Mixtral (8x7B; 12/2023). We deliberately decided against hosted models like OpenAI's GPT-4 because we cannot determine their energy consumption.

TABLE I
RESULTS OF OUR EXPERIMENTS (INFERENCE). THE FIRST SECTION CONTAINS FINE-TUNING SETTING, THE SECOND SECTION CONTAINS ZERO-SHOT PROMPTING SETTING.

Approach	Prec.	Rec.	f_1	CO_2 (g)	CE_{rel}	$n(CE_{rel})$	Δ_{CE}
BERT	0.082	0.500	0.125	2.638	21.14	1.00	0.0336
XLNet	0.090	0.136	0.106	4.715	44.48	0.00	*base*
SciBERT	0.069	0.538	0.112	4.205	37.68	0.29	0.0063
DeBERTa	0.081	0.731	0.134	3.982	29.72	0.63	0.0332
DistilBERT	0.093	0.462	0.138	5.643	41.01	0.15	0.0264
Llama2:7b	0.090	0.556	0.146	13.034	89.03	0.93	0.0031
Llama2:13b	0.089	0.888	0.144	16.765	116.43	0.89	*base*
Llama2:70b	0.119	0.692	0.187	155.450	831.28	0.00	0.0046
Mistral:7b	0.340	0.487	0.308	8.941	29.00	1.00	0.3081
Mixtral:8x7b	0.284	0.477	0.286	23.715	82.95	0.93	0.1003

A. Results

Table I contains the results for our experiments.

In the first section, the approaches achieve similar f_1-scores; DistilBERT is slightly ahead. BERT has the lowest carbon emission and the best CE_{rel}. For the delta-based carbon emission, we take XLNet as the baseline because it has the worst f_1-score. BERT, SciBERT, and DeBERTA emit less carbon, but do not perform best according to f_1-score. While DistilBERT performs best according to f_1-score, it emits more carbon. With these metrics, a user can decide which classifier to select, factoring in all these metrics. Looking at the scores for Δ_{CE}, BERT is likely the best choice if a slightly lower classification performance is acceptable. Another candidate is DeBERTa that scores slightly worse than BERT; its better f_1-score is slightly offset by the increased carbon emissions. It also has a high recall, which can be good for a semi-automatic approach. DeBERTa still has the second-lowest carbon emission and good performance, but without looking at the carbon emission, the approach could be overshadowed by the better f_1-score of DistilBERT. Comparing DistilBERT and DeBERTa shows exemplarily how the metric can be used to assess the performance regarding Δ_{CE}: Better f_1-score but worse carbon emission results in a lower score for DistilBERT.

For the second section, Mistral achieves the best f_1-score due to its comparably balanced performance. While Llama2:13b has the highest recall, it is among the low-performing classifiers precision-wise. With no surprise, the biggest LLM Llama2:70b emits the most carbon by a big margin. Llama2:70b is by far the worst classifier based on the relative carbon emission due to its lofty carbon emission and its comparably wretched f_1-score. The delta-based carbon emission, confirms this. In this setting, selecting Mistral is the best choice.

[2] https://textattack.readthedocs.io/en/latest/apidoc/textattack.augmentation.html, Last acc. 2024-11-11

[3] https://github.com/phiyodr/multilabel-oversampling, Last acc. 2024-11-11

TABLE II
COMPARISON OF THE RESULTS OF BERT AND XLNET IN THE BASE SETTING AND WITH DATA AUGMENTATION, MEASURING ENERGY ONLY DURING INFERENCE.

Approach	Prec.	Rec.	f_1	CO_2 (g)	CE_{rel}	Δ_{CE}
BERT	0.082	0.500	0.125	2.638	21.14	0.0336
$BERT_{augmented}$	0.092	0.808	0.162	7.080	43.64	0.0375
XLNet	0.090	0.136	0.106	4.715	44.48	*base*
$XLNet_{augmented}$	0.128	0.160	0.139	14.205	102.05	0.0110

Table II compares BERT and XLNet for the base setting and with data augmentation. For BERT, the recall increased a lot, and precision remained stable, resulting in only minor changes for f_1-score. Due to the augmentation, carbon emissions increased more than 2.6x, resulting in a much worse CE_{rel}. Still, Δ_{CE} shows that the increased f_1-score balances out the increased emissions, resulting in slightly better results compared to the non-augmented version. In comparison, the f_1-score of XLNet increased similarly, but the carbon emissions amplified threefold. As a result, the combined metrics identify XLNet as less favorable.

B. Threats to Validity

In the following, we discuss threats to validity based on the categories presented and surveyed in [46]:

Construct Validity. Construct validity refers to the experimental setup to address the research objective. We used common practices (e.g., recommendations for prompt construction) and techniques (e.g., stratified split, hyperparameter optimization via Optuna) for the experimental design. In addition, we used common performance evaluation metrics as well as our novel proposed sustainability metrics (cf. Section IV-B) to assess the derived models, enabling trade-off decisions.

Internal Validity. The dataset used for the experiments is a representative of a gold standard in the considered research domain and publicly available [21]. However, there might be a threat that the abstract for text classification is not sufficient for the information provided. Full-text information could provide different results.

External Validity. External validity refers to the generalizability of our approach and a representative selection of the experimental dataset for our purpose. The labels for *Evaluation Method* are mainly based on the classification and description of the ACM Empirical Standards [47] and are, thus, suitable for software engineering in general.

Repeatability. For reproducibility, we used fixed seeds for the fine-tuning approaches and a low temperature (temp = 0) for the zero-shot approach in each experiment. The dataset and source code are publicly available. Our experimental setup's prerequisites must be fulfilled and aligned with the best practices discussed in Section V.

VII. CONCLUSION

In this paper, we focused on why we should consider sustainability aspects in addition to performance measurements to assess the suitability of different automated text classification approaches and support trade-off decisions. While deep learning techniques, especially those based on transformer architecture, have gained popularity in the last few years, the benefits of these approaches need to be investigated and assessed in terms of their computational costs. As these models should be applied in a large-scale context, a slight performance improvement might not justify the increased power consumption or similar costs, as also stated by Fu et al. [48]. Therefore, we proposed novel sustainability metrics to assess and support trade-off decisions between performance and carbon emission. As new approaches for the same classification task should be baselined against some simpler and faster alternatives [48], we differentiate between metrics that express the *total* carbon emission (unit), the *relative* carbon emission w.r.t. the corresponding f-score, the *delta-based version* w.r.t. the corresponding f-score in relation to the baseline (i.e., worst performing classification approach) and, finally, the *normalized* versions. Besides these metrics, we provided best practices for the assessment. We showcased our approach via a use case example by classifying papers in software architecture research according to the evaluation methods used. In future research, we aim to review and evaluate our proposed metrics in further case studies. Furthermore, based on these insights, we plan to expand the metrics' concepts to, e.g., additional sustainability dimensions, infrastructure settings, or trade-off index for comprehensive and sustainable comparisons.

ACKNOWLEDGEMENTS

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the National Research Data Infrastructure – NFDI 52/1 – project number 501930651, NFDIxCS and supported by funding from the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF).

REFERENCES

[1] Z. Kurtanović and W. Maalej, "Automatically classifying functional and non-functional requirements using supervised machine learning," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 490–495.

[2] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir *et al.*, "Requirements classification with interpretable machine learning and dependency parsing," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 142–152.

[3] T. Hey, J. Keim, A. Koziolek *et al.*, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 2020, pp. 169–179.

[4] T. Hey, J. Keim, and S. Corallo, "Requirements classification for traceability link recovery," in *2024 IEEE 32nd International Requirements Engineering Conference (RE'24)*. Institute of Electrical and Electronics Engineers (IEEE), 2024.

[5] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd International RE Conference*, 2015, pp. 116–125.

[6] R. Kallis, A. Di Sorbo, G. Canfora *et al.*, "Predicting issue types on github," *Science of Computer Programming*, vol. 205, p. 102598, 2021.

[7] O. Meqdadi, N. Alhindawi, J. Alsakran *et al.*, "Mining software repositories for adaptive change commits using machine learning techniques," *Information and Software Technology*, vol. 109, pp. 80–91, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584919300084

[8] A. Kaplan, J. Keim, M. Schneider *et al.*, "Combining knowledge graphs and large language models to ease knowledge access in software architecture research," in *SemTech4STLD 2024, co-located with the ESWC 2024*, ser. CEUR Workshop Proceedings, vol. 3697. CEUR-WS, 2024, pp. 76–82.

[9] IBM (International Business Machines Corporation), "IBM Global AI Adoption Index – Enterprise Report," November 2023, last accessed on 2024-11-11. [Online]. Available: https://filecache.mediaroom.com/mr5mr_ibmspgi/179414/download/IBMGlobalAIAdoptionIndexReportDec.2023.pdf

[10] J. Devlin, M.-W. Chang, K. Lee *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 NACL.* Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[11] Y. Liu, M. Ott, N. Goyal *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[12] A. Radford, K. Narasimhan, T. Salimans *et al.*, "Improving language understanding with unsupervised learning," 2018.

[13] ——, "Improving language understanding by generative pre-training," 2018.

[14] T. Brown, B. Mann, N. Ryder *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[15] S. Luccioni, Y. Jernite, and E. Strubell, "Power hungry processing: Watts driving the cost of AI deployment?" in *ACM Conference on Fairness, Accountability, and Transparency 2024, (FAccT).* ACM, 2024, pp. 85–99. [Online]. Available: https://doi.org/10.1145/3630106.3658542

[16] J. Castaño, S. Martínez-Fernández, X. Franch *et al.*, "Exploring the carbon footprint of hugging face's ML models: A repository mining study," in *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2023.* IEEE, 2023, pp. 1–12. [Online]. Available: https://doi.org/10.1109/ESEM56168.2023.10304801

[17] P. Lago, S. A. Koçak, I. Crnkovic *et al.*, "Framing sustainability as a property of software quality," *Commun. ACM*, vol. 58, no. 10, pp. 70–78, 2015. [Online]. Available: https://doi.org/10.1145/2714560

[18] A. N. P. Kumar, J. Bogner, M. Funke *et al.*, "Balancing progress and responsibility: A synthesis of sustainability trade-offs of ai-based systems," in *21st IEEE International Conference on Software Architecture, ICSA 2024 - Companion, Hyderabad, India, June 4-8, 2024.* IEEE, 2024, pp. 207–214. [Online]. Available: https://doi.org/10.1109/ICSA-C63560.2024.00045

[19] R. Verdecchia, J. Sallou, and L. Cruz, "A systematic review of Green AI," *WIREs Data Mining and Knowledge Discovery*, vol. 13, no. 4, p. e1507, 2023. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1507

[20] C. Becker, R. Chitchyan, L. Duboc *et al.*, "Sustainability design and software: The karlskrona manifesto," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015.* IEEE, 2015, pp. 467–476. [Online]. Available: https://doi.org/10.1109/ICSE.2015.179

[21] M. Konersmann, A. Kaplan, T. Kühn *et al.*, "Replication package of "evaluation methods and replicability of software architecture research objects"," in *IEEE 19th International Conference on Software Architecture Companion, ICSA Companion 2022, Honolulu, HI, USA, March 12-15, 2022.* IEEE, 2022, p. 58. [Online]. Available: https://doi.org/10.1109/ICSA-C54293.2022.00021

[22] ResearchClassificationFramework, "Research classification framework-dev," 2023. [Online]. Available: https://gitlab.com/software-engineering-meta-research/karagen/research-classification-framework/classificationframework-dev

[23] Q. Li, H. Peng, J. Li *et al.*, "A survey on text classification: From traditional to deep learning," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 2, pp. 31:1–31:41, 2022. [Online]. Available: https://doi.org/10.1145/3495162

[24] T. M. Mitchell, *Machine learning.* McGraw-Hill Education, 1997.

[25] Z.-H. Zhou, *Machine learning.* Springer nature, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *NeurIPS 2017*, I. Guyon, U. von Luxburg, S. Bengio *et al.*, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[27] L. Tunstall, L. von Werra, T. Wolf *et al.*, *Natural Language Processing with Transformers: Building Language Applications with Hugging Face.* Sebastopol: O'Reilly, 2022.

[28] S. Idowu, D. Strüber, and T. Berger, "Asset management in machine learning: State-of-research and state-of-practice," *ACM Computing Surveys*, vol. 55, no. 7, pp. 144:1–144:35, 2023. [Online]. Available: https://doi.org/10.1145/3543847

[29] K. Kluge and P. Jenkner, "13 Best Tools for ML Experiment Tracking and Management in 2024," September 2024, last accessed on 2024-11-11. [Online]. Available: https://neptune.ai/blog/best-ml-experiment-tracking-tools

[30] M. M. Cantallops, S. Sánchez-Alonso, E. García-Barriocanal *et al.*, "Traceability for trustworthy AI: A review of models and tools," *Big Data Cogn. Comput.*, vol. 5, no. 2, p. 20, 2021. [Online]. Available: https://doi.org/10.3390/bdcc5020020

[31] L. F. Wolff Anthony, B. Kanding, and R. Selvan, "Carbontracker: tracking and predicting the carbon footprint of training deep learning models," *arXiv e-prints*, pp. arXiv–2007, 2020.

[32] A. Lacoste, A. Luccioni, V. Schmidt *et al.*, "Quantifying the Carbon Emissions of Machine Learning," Oct. 2019.

[33] B. Courty, V. Schmidt, S. Luccioni *et al.*, "mlco2/codecarbon: v2.4.1," May 2024. [Online]. Available: https://doi.org/10.5281/zenodo.11171501

[34] T. Trébaol, "Cumulator—a tool to quantify and report the carbon footprint of machine learning computations and communication in academia and healthcare," 2020.

[35] S. A. Budennyy, V. D. Lazarev, N. Zakharenko *et al.*, "Eco2ai: carbon emissions tracking of machine learning models as the first step towards sustainable AI," *Doklady Mathematics*, vol. 106, 2022. [Online]. Available: https://doi.org/10.1134/S1064562422060230

[36] L. Lannelongue, J. Grealey, and M. Inouye, "Green algorithms: quantifying the carbon footprint of computation," *Advanced science*, vol. 8, no. 12, p. 2100707, 2021.

[37] A. Faiz, S. Kaneda, R. Wang *et al.*, "LLMcarbon: Modeling the end-to-end carbon footprint of large language models," *CoRR*, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2309.14393

[38] S. M. Hasan, A. R. Shahid, and A. Imteaj, "Towards sustainable secureml: Quantifying carbon footprint of adversarial machine learning," in *IEEE International Conference on Communications Workshops, ICC 2024 Workshops, Denver, CO, USA, June 9-13, 2024.* IEEE, 2024, pp. 1359–1364. [Online]. Available: https://doi.org/10.1109/ICCWorkshops59551.2024.10615723

[39] ——, "Evaluating sustainability and social costs of adversarial training in machine learning," *IEEE Consumer Electronics Magazine*, pp. 1–6, 2024. [Online]. Available: https://doi.org/10.1109/MCE.2024.3458350

[40] B. Penzenstadler and H. Femmer, "A generic model for sustainability with process- and product-specific instances," in *Proceedings of the 2013 GIBSE*, S. Malakuti, C. Bockisch, U. Assmann *et al.*, Eds. ACM, 2013, pp. 3–8. [Online]. Available: https://doi.org/10.1145/2451605.2451609

[41] D. Powers, "Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[42] A. S. Luccioni and A. Hernández-García, "Counting carbon: A survey of factors influencing the emissions of machine learning," *CoRR*, vol. abs/2302.08476, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.08476

[43] R. Pereira, M. Couto, F. Ribeiro *et al.*, "Ranking programming languages by energy efficiency," *Science of Computer Programming*, vol. 205, p. 102609, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167642321000022

[44] "CodeCarbon project," last accessed on 2024-11-11. [Online]. Available: https://mlco2.github.io/codecarbon/

[45] "Weights & Biases Documentation," last accessed on 2024-11-11. [Online]. Available: https://docs.wandb.ai/

[46] M. Konersmann, A. Kaplan, T. Kühn *et al.*, "Evaluation methods and replicability of software architecture research objects," in *19th IEEE International Conference on Software Architecture, ICSA 2022, Honolulu, HI, USA, March 12-15, 2022.* IEEE, 2022, pp. 157–168. [Online]. Available: https://doi.org/10.1109/ICSA53651.2022.00023

[47] P. Ralph, S. Baltes, D. Bianculli *et al.*, "ACM SIGSOFT Empirical Standards," *CoRR*, vol. abs/2010.03525, 2020.

[48] W. Fu and T. Menzies, "Easy over hard: a case study on deep learning," in *11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017*, E. Bodden, W. Schäfer, A. van Deursen *et al.*, Eds. ACM, 2017, pp. 49–60. [Online]. Available: https://doi.org/10.1145/3106237.3106256