

A Classification Framework for Scientific Documents to Support Knowledge Graph Population






Angelika Kaplan ¹, Jan Keim ¹, Lukas Greiner ¹, Anne Kozirolek ¹, and Ralf Reussner ¹

Abstract: Research papers are a central communication medium to share new scientific insights and progress and are, nowadays, stored as PDF files. However, little effort is spent on reorganizing information with effective knowledge classification and comprehensive representation during the publication process. In terms of software engineering (SE), those papers are also aligned to software artifacts and research data. Aggregating knowledge and empirical evidence is done with systematic literature studies that tend to be very time-consuming and require a manual inspection of the respective research artifacts (paper- and data-wise). Research knowledge graphs like the Open Research Knowledge Graph (ORKG) aim to contribute to and rethink scholarly communication by providing formats while easing the processing of semantic information. Therefore, ORKG offers templates to summarize and structure a research artifact's content, providing metadata as well. Based on this, researchers can connect similar papers, reuse replication artifacts, and generate literature studies more easily. However, adding papers to ORKG is still a tedious manual process. Moreover, selecting suitable template formats is challenging and can be highly domain-specific. To support the manual process, we aim to provide an automated classification framework for scientific papers, supporting the knowledge graph population. This framework intends to be flexible, allowing various input data and schemas, so it can be applied and trained in a multitude of research fields in SE. In this paper, we present the concept of the framework's implementation, and an excerpt of the evaluation result in one of the research subfields in SE, namely software architecture and design.

Keywords: Automated Text Classification, Natural Language Processing, Scientific Documents in Software Engineering, Knowledge Graph Population, Semantic Aspects

1 Introduction

Research in software engineering (SE) is a key pillar for progress in science, practice, and society. Scientific papers are the communication medium for research results and insights that enhance collective knowledge and advance state-of-the-art [Be18]. These digital documents, mostly PDF files, are hardly machine-readable and -processable. As a result, it is difficult to access and automatically process scientific knowledge communicated in this form. This document-based process does not exploit the full potential of digitalization [Ja19].

¹ Karlsruhe Institute of Technology (KIT), KASTEL – Institute of Information Security and Dependability, Am Fasanengarten 5, 76131 Karlsruhe, Germany,
angelika.kaplan@kit.edu,  <https://orcid.org/0009-0009-9101-5833>;
jan.keim@kit.edu,  <https://orcid.org/0000-0002-8899-7081>;
lukas.greiner@alumni.kit.edu,  <https://orcid.org/0009-0004-5236-8226>;
kozirolek@kit.edu,  <https://orcid.org/0000-0002-1593-3394>;
reussner@kit.edu,  <https://orcid.org/0000-0002-9308-6290>

Consequently, it is labor-intensive and cumbersome for researchers and practitioners to aggregate valuable knowledge and empirical evidence just based on documents accessed via academic search engines (i.e., digital libraries and indexers). This has to be done in time-consuming (systematic) literature studies [Oe20; ZB11] that require enhanced information literacy skills [Pi21] and manual inspection of the papers and corresponding replication artifacts in SE. Moreover, academic search engines do not seem to effectively support such reviews and ease knowledge access, as they are struggling with bugs (i.e., faults in the search engines) and usability issues [LR22]. As a result, valuable knowledge in science might be missed [Pi21]. However, research knowledge graphs like the Open Research Knowledge Graph (ORKG) offer great potential in this regard as they are structured representations of knowledge, enabling a semantic description by linking existing meta-data and content-data of scientific research artifacts, supporting the FAIR-principles [Oe20]. However, adding papers to ORKG, i.e., conducting knowledge graph population, is still a tedious manual process. Researchers first need to classify the knowledge into a template that ORKG then uses to populate the knowledge graph. To support the first step, we provide a (semi-) automated classification framework for scientific papers, using and benefiting from recent advancements of Large Language Models (LLMs). In this paper, we mainly address the research question *RQ: How to construct a (semi-) automated classification framework for classifying scientific papers (in software architecture (SWA) research)?*

Consequently, our contributions are:

- C1 An overview of our research project and implementation that supports automated classification approaches for research artifacts (e.g., scientific papers) using LLMs, enabling knowledge graph population.
- C2 The framework's evaluation via use case in SWA research using a gold standard dataset [Ko22b], presenting an excerpt of the classification results and showcasing the framework's maturity. This includes a template for experimental setups and evaluations, including performance and sustainability metrics.
- C3 An open access repository [Re23] comprising artifacts like the source code of the framework and the results of our experiments (cf. Section 5) to foster collaborations in the research community to reuse and further improve the framework.

2 Foundations

There are various approaches for classification, with most approaches based on machine learning. Classic machine learning approaches like SVM, Naïve-Bayes, or Linear Regression need labeled data to learn from this training data. LLMs, however, can either be *fine-tuned* or used via *prompt engineering*. What is best, is usually based on the amount of training data [Tu22], as Figure 1 depicts.

Fine-Tuning means that data is used to continue training a pre-trained LLM to improve its performance for a given task. There are two main ways for fine-tuning. First, additional

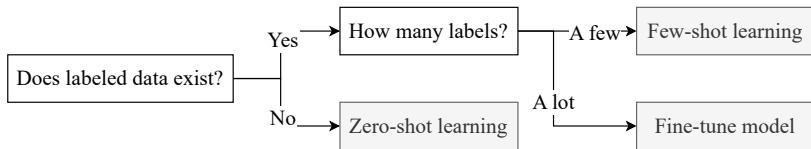


Fig. 1: Overview of suitable classification approaches based on the amounts of labeled data (adapted from Tunstall et al. [Tu22])

data, usually task- or domain-specific, can be used to further train the model’s parameters. Second, a task-specific layer is added to the model using the output of the original model. The labeled data is then used to train this new layer for the specific task.

Prompt Engineering describes the process of developing and optimizing prompts for LLMs to improve the model’s output. In this context, a specified prompt is a natural language text describing the task, or query, that an LLM should perform. There are different types of prompts like zero-shot prompts and few-shot prompts. Zero-shot prompting means no examples about the specific task are given to the LLM. In few-shot prompting, some examples are provided. Besides these, there are other prompting techniques and tricks that influence an LLM’s response. Further, researchers can set and vary the temperature ($\text{temp} \in [0,1]$) of these models to influence the randomness and, thus, the creativity of the responses.

3 Use Case Example

In this section, we present our classification use case to demonstrate the maturity of our developed framework (cf. Section 4). Furthermore, we derive challenges for an automated classification approach based on our use case and general considerations. We analyze and address these challenges to foster our framework’s maturity, reusability, and extensibility.

3.1 Data Schema

To show the feasibility of our classification framework, we used the gold standard dataset created in the literature review process [Ko22b]. In this literature study, we specified a data schema for software architecture (SWA) research objects to provide an overview of current research practices in SWA research while establishing a community standard for semantically describing papers in this research domain. As depicted in Figure 2, the data schema describes metadata and content data of a research artifact (i.e., paper and corresponding replication artifact) in SWA research. Metadata (i.e., data about research artifacts) of papers encompasses, e.g., bibliographic information. Content data focuses on how research enhances the current body of research knowledge. This is done by describing *findings* and the corresponding *evidence*. These kinds of considerations are quite generic.

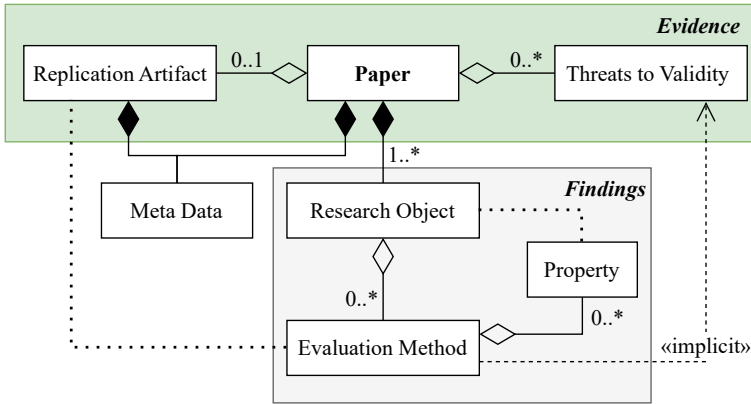


Fig. 2: Data schema and model for software architecture research (adapted and extended from [Ko22a])

The characterization of *findings* is represented with the triple: *Research Object*, *Property*, and *Evaluation Method*. *Research Object* is the object under study and has 13 sub-classes. *Property* is the property evaluated using the evaluation method with 3 direct sub-categories (i.e., *Product Quality* with 8 sub-classes, *Quality in Use* with 5 sub-classes, and *Quality of Method* with 9 sub-classes). *Evaluation Method* is the chosen research method for evaluation w.r.t. to a property with 13 sub-classes, based on the ACM Empirical Standards². The characterization of *evidence* is represented by *Threats to Validity* (i.e., categories of threats that refer to the validity of findings/statements that are mainly reflected in the method design of the evaluation) and the *Replication Artifact* (e.g., information about tool support, input data, code to replicate the research results and corresponding findings).

3.2 Dataset Analysis

For the dataset analysis in this section, we exemplarily regard the content data w.r.t. to the *Property* characterization. The dataset is based on full papers (technical papers) that were published at the International Conference on Software Architecture (ICSA) and the European Conference on Software Architecture (ECSA) between 2017 and 2021. The dataset consists of 153 classified papers according to the data schema introduced in Section 3.1. At least two reviewers classified each paper to derive a gold standard dataset [Ko22b]. Figure 3 depicts the distribution of the labels. The left-hand side indicates that *Property* has missing representatives in the three main categories *Product Quality*, *Quality in Use*, and *Quality of Method*. The aggregated presentation on the right-hand side of the figure emphasizes the highly imbalanced distribution of these categories.

² ACM SIGSOFT Empirical Standards Released, Paul Ralph: <https://doi.org/10.1145/3437479.3437483>

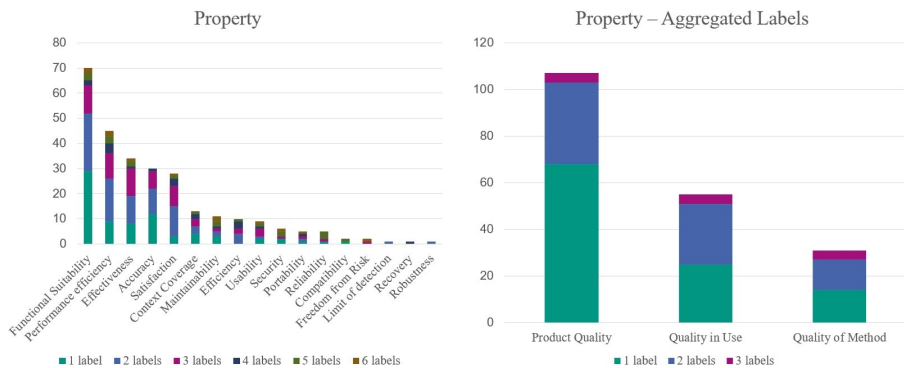


Fig. 3: Occurrences of the *Property* labels in the gold standard dataset [Ko22b]

3.3 Derived Challenges

Based on the preceding data analysis of the gold standard dataset w.r.t. *Property*, general implications of multi-label classification tasks, and our research objective, i.e., enabling knowledge graph population in a (semi-) automated way for various data schema in SE using ORKG, we derive the following challenges:

Degree of complexity. Describing research artifacts w.r.t. metadata and content data is a complex task (cf. Section 3.2). We have the highest complexity when facing a multi-label classification problem, and standard libraries rarely consider multi-label problems.

Missing representatives in the dataset. Missing representations in a dataset (cf. dataset analysis of *Property*) is a big issue when constructing an automated classification approach. We plan to support both classification approaches (cf. Section 2), in particular prompting for missing representatives. To support fine-tuning, we consider classifying data items at a higher abstraction level within the classification hierarchy (cf. *Property* in data schema) to increase the number of representatives per class.

Highly imbalanced dataset. To support variations in the experimental setup, esp. for small or imbalanced datasets (for fine-tuning), the implementation should support data augmentation like (over-) sampling and perturbation strategies. In addition, different splitting strategies should also be supported. For example, in our framework, we support a data splitting mechanism based on metadata of papers, i.e., based on authors (one author appears either in the training, validation, or test split) to support the robustness of the derived classifiers.

Evolution scenarios w.r.t. data schema. As research fields evolve, the knowledge description and data schemas of the research artifacts will need to evolve. Consequently, our framework should support different labeled datasets.

Generalizability. Different research fields in SE require different datasets and data schemas. Thus, our framework should be open to different schemas and datasets.

Suitable classification approach. Based on the different characteristics of the dataset and analysis, the framework should support both classification approaches (cf. Section 2) and

tracking of results for benchmarking. This implies tracking various metrics for performance, time (latency), and sustainability aspects. While metrics for performance and time are commonly used, sustainability metrics are rarely considered. Yet, similarly performing classifiers might differ significantly in sustainability. At scale, this can make a huge difference and, as such, these metrics are central to our framework. This way, ecological factors can be considered when choosing a classifier.

Open science principles. We value open science principles; therefore, our framework should support different variants of open-source and open weights models like Llama [Du24], besides comparing and benchmarking them to models from, e.g., OpenAI (GPT).

4 Classification Framework

In this section, we present our framework for automated classification approaches based on scientific documents using LLMs and answer our research question from Section 1: *How to construct a (semi-) automated classification framework for classifying scientific papers?*

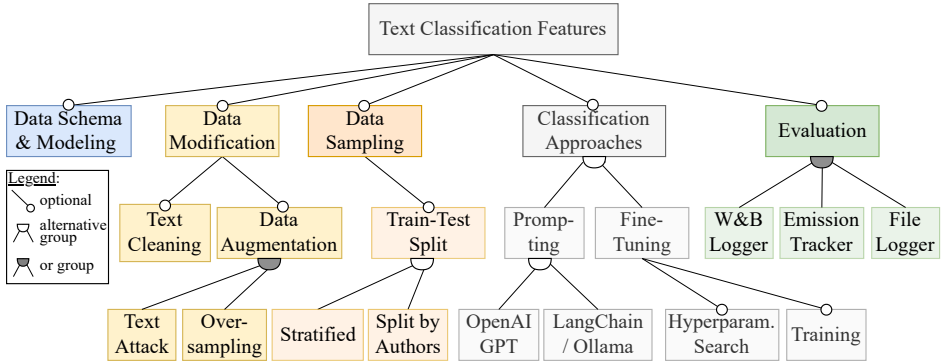


Fig. 4: Feature diagram of the current state of the text classification framework and its variation point for different experimental setups. The feature *Data Schema & Modeling* is universally applicable if the corresponding classification task adheres to our specified input format (see example in List. 1).

Architecture. Following, we present the basic architecture of our classification framework and highlight its core features (cf. Figure 4). As discussed in Section 3.3, the framework should support (1) different types and extensions of datasets (i.e., different research fields/data schemas, supporting data schema evolution), (2) different classification approaches and different versions and variants of LLMs, and (3) various experiment settings and evaluation strategies (e.g., data augmentation and sampling).

Therefore, we used the concept of a *Blackboard Pattern* (cf. [Sc96]) that consists of (i) a blackboard, where all data is written on, (ii) a controller, that runs the agents, and (iii) agents that manipulate the data on the blackboard. This pattern offers the possibility of having different knowledge sources that can add data to the blackboard without sticking to a strict

data representation schema. This pattern also allows the connection of different types of agents and the provision of different variation points (cf. Figure 4) in the experimental setup. All runs of the experiments are logged to the tracking platform Weights & Biases³ (W&B) w.r.t. visualization techniques and metrics (performance, time, and sustainability).

The components of the framework are divided into: *Data*, *Classifier*, and *Utilities*. Each block has a distinct responsibility with clearly defined interfaces; the dependencies between the blocks are minimized. Each of the components can be exchanged or adapted when requirements are changing. This way, we can realize the various features depicted in Figure 4. The component *Classifier* contains all relevant agents to run the classification approaches, i.e., for prediction, training, hyperparameter optimization (with Optuna [Ak19], including the specification of the search space and number of trials) and for prompting, using LLMs provided by the *Hugging Face Transformers*⁴, *Ollama*⁵ and *OpenAI*⁶ library. The visualization of the hyperparameter search is tracked with W&B Sweeps. The component *Data* contains all agents for the data creation, the augmentation (i.e., different data augmentation techniques like text perturbation via TextAttack⁷ and multi-label-oversampling strategies), the data handling, and the splitting (e.g., author split) modules for manipulating the datasets before handing them over to the *Classifier*. The component *Utilities* contains, among other, the data schema and description handling, and the logging module.

Defining classification schemas. One input of our approach is a definition of a classification (taxonomy) schema. As depicted in List. 1, our input format is specified via JavaScript Object Notation (JSON). With this format, we can read in any classification schema that follows a tree-like structure: Each element can have an arbitrary number of children elements, that themselves again can have children. For each element, a description provides information about the class, its meaning, and its purpose. This knowledge can inform the LLMs.

```

1  { "Category 1": {
2    "description": "description of Category 1",
3    "sub_elements": {
4      "Label 1.1": {
5        "description": "description of Label 1.1",
6        "sub_elements": {
7          "Label 1.1.1": {
8            "description": "description of Label 1.1.1": { ... }
9          }
10         }},
11     ...
12 }

```

List. 1: Example excerpt of the input (taxonomy) file to define classification schemas

³ Weights & Biases (W&B) Docs: <https://docs.wandb.ai/guides/>

⁴ Transformers - Hugging Face: <https://huggingface.co/docs/transformers/index>

⁵ Ollama Docs: <https://github.com/ollama/ollama/tree/main/docs>

⁶ OpenAI developer platform: <https://platform.openai.com/docs/overview>

⁷ TextAttack 0.3.9 Documentation: <https://textattack.readthedocs.io/en/latest/apidoc/textattack.augmentation.html>

5 Evaluation

Following, we present an excerpt of the classification results w.r.t. *Property* (3 high-level classes), where we sum up the labels to higher representatives, supported by our framework (cf. List. 1). Please refer to our open-access repository for more details about the experiment and the results. We run our experiments on a server node with 252 GB RAM, 2x Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz (112 parallel threads), and a Tesla V100S-32GB. We formulate the following evaluation questions (EQs):

EQ 1: Which automated classification paradigm achieves the best performance results?

EQ 2: What are suitable configurations for the classification task, especially regarding performance and energy efficiency?

There are various metrics that are crucial for evaluation, as discussed in Section 3.2. Therefore, our framework supports tracking of (i) metrics like accuracy, precision, recall (i.e., sensitivity), specificity, and (macro averaged) f_1 -score as well as Jaccard score and hamming loss (ii) time measurements (iii) sustainability aspects such as carbon emission (CO₂) and energy consumption (kWh). For *fine-tuning*, we experiment with our basic dataset and the augmented version. We consider the learning rate, the number of training epochs, the training batch size, and the weight decay for the AdamW optimizer as hyperparameters for the training, optimizing w.r.t. the macro f_1 -score with 200 Optuna trials. Users can easily adapt these settings within our framework. For the basic dataset, BERT (110M; 10/18) achieved the best results with 59.4% f_1 -score (see Figure 5). For the augmented version, we observe an improvement of the f_1 -score; the best model is DeBERTa (304M; 03/23) with 71.08% and an acceptable CO₂ emission relative to the worst performing classifier (SciBERT) and the classifier with the lowest emission (DistilBERT), see Table 1.

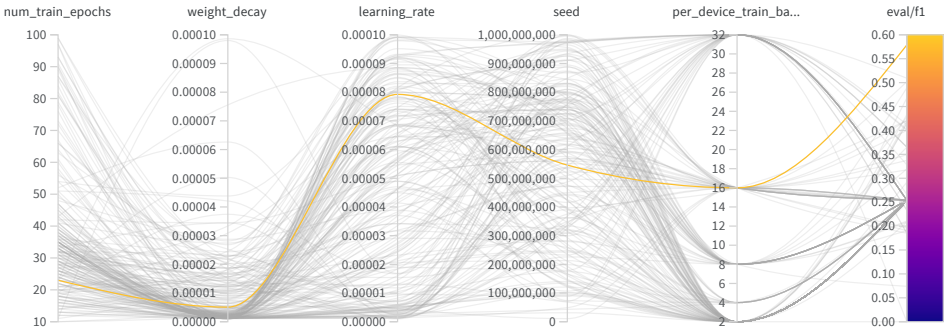


Fig. 5: BERT hyperparam. search *Property* (basic dataset). The yellow line shows the best parameters.

We experiment with a zero-shot approach for *prompting*, setting the temperature to zero. For the category *Property*, Llama2:7b achieved the best results (f_1 -score 56.75%). The low specificity indicates that the model prioritizes high recall. The hamming loss also shows that the predicted label array at the Llama2:7b model differs most from the other LLMs.

LM (#Param.; Month/Year)	F1 (augm.)	CO2 (kg)	Δ F1-score	Δ CO2	Δ CO2/ Δ F1
SciBERT (110M; 03/19)	0.4935	0.006063	–	–	–
DistilBERT (66M; 10/19)	0.6269	0.001908	0.1335	-0.0042	-0.0311
XLNet (92M; 06/19)	0.6463	0.013277	0.1528	0.0072	0.0472
BERT (110M; 10/18)	0.6836	0.002281	0.1901	-0.0038	-0.0199
DeBERTa (304M; 03/23)	0.7108	0.005076	0.2173	-0.0010	-0.0045

Tab. 1: Results of trained models with augmented test data for the category *Property*, including an overview of how much the emissions change relative to the worst-performing classifier.

LM	F1	Precision	Recall	Jaccard	Hamming loss	Specificity
GPT-3.5-turbo-0125	0.5163	0.4556	0.6991	0.3714	0.4728	0.3553
GPT-4-0125-preview	0.4380	0.4286	0.5948	0.3045	0.4989	0.4696
Llama2:70b	0.5487	0.4342	0.7851	0.4089	0.4488	0.2372
Llama2:13b	0.5672	0.4334	0.9526	0.4196	0.5468	0.0804
Llama2:7b	0.5675	0.4263	0.9861	0.4244	0.5708	0.0090
Mistral:7b	0.4992	0.4321	0.6120	0.3544	0.4444	0.4156
Mixtral:8x7B	0.5145	0.4516	0.6709	0.3862	0.3900	0.3476

Tab. 2: Results of the zero-shot approach for the category *Property*, macro averaged

In conclusion, fine-tuning the LLMs with an augmented dataset and an acceptable CO2 emission leads to the best-performing classifier in the experimental setup.

6 Conclusion

In this paper, we introduced a classification framework for scientific documents using LLMs, supporting multi-labeling problems. We presented an overview of our implementation and discussed preliminaries w.r.t. challenges. Further, we argued about the generalizability and flexibility of our framework and demonstrated its feasibility with a use case example using a gold standard dataset in software architecture research (see [Ko22b]) while presenting an excerpt of the results for the classification task. In future work, we plan to apply our framework to different data schemas in various sub-research fields of software engineering and investigate how well the corresponding communities can benefit from our framework.

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

- [Ak19] Akiba, T.; Sano, S., et al.: Optuna: A Next-Generation Hyperparameter Optimization Framework. In: The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Pp. 2623–2631, 2019.
- [Be18] Bertolino, A.; Calabrò, A., et al.: A categorization scheme for software engineering conference papers and its application. *Journal of Systems and Software* 137, pp. 114–129, 2018, doi: 10.1016/J.JSS.2017.11.048.
- [Du24] Dubey, A.; Jauhri, A., et al.: The Llama 3 Herd of Models, 2024, arXiv: 2407.21783 [cs.AI].
- [Ja19] Jaradeh, M. Y.; Oelen, A., et al.: Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge. In: Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19–21, 2019. ACM, pp. 243–246, 2019, doi: 10.1145/3360901.3364435.
- [Ko22a] Konersmann, M.; Kaplan, A., 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, pp. 157–168, 2022, doi: 10.1109/ICSA53651.2022.00023.
- [Ko22b] Konersmann, M.; Kaplan, A., 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, p. 58, 2022, doi: 10.1109/ICSA-C54293.2022.00021.
- [LR22] Li, Z.; Rainer, A.: Academic search engines: constraints, bugs, and recommendations. In: Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation, A-TEST 2022, Singapore, Singapore, November 17–18, 2022. ACM, pp. 25–32, 2022, doi: 10.1145/3548659.3561310.
- [Oe20] Oelen, A.; Jaradeh, M. Y., et al.: Generate FAIR Literature Surveys with Scholarly Knowledge Graphs. In: JCDL '20: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, Virtual Event, China, August 1–5, 2020. ACM, pp. 97–106, 2020, doi: 10.1145/3383583.3398520.
- [Pi21] Pizard, S.; Acerenza, F., et al.: Training students in evidence-based software engineering and systematic reviews: a systematic review and empirical study. *Empir. Softw. Eng.* 26 (3), p. 50, 2021, doi: 10.1007/S10664-021-09953-9.
- [Re23] ResearchClassificationFramework: Research Classification Framework-dev, 2023, URL: <https://gitlab.com/software-engineering-meta-research/karagen/research-classification-framework/classificationframework-dev>.
- [Sc96] Schmidt, D.; Stal, M., et al.: Pattern-oriented software architecture, volume 1: a system of patterns, 1996.
- [Tu22] Tunstall, L.; von Werra, L., et al.: Natural Language Processing with Transformers: Building Language Applications with Hugging Face. O'Reilly, Sebastopol, 2022, ISBN: 978-1-09-813679-6.
- [ZB11] Zhang, H.; Babar, M. A.: An Empirical Investigation of Systematic Reviews in Software Engineering. In: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22–23, 2011. IEEE Computer Society, pp. 87–96, 2011, doi: 10.1109/ESEM.2011.17.