

# Comparing general purpose pre-trained Word and Sentence embeddings for Requirements Classification

Federico Cruciani<sup>a</sup>, Samuel Moore<sup>a</sup> and Chris Nugent<sup>a</sup>

<sup>a</sup>*School of Computing, Ulster University, 2-24 York Street, Belfast, BT15 1AP, United Kingdom*

## Abstract

The recent evolution of NLP has enriched the set of DL-based approaches to include a number of general-purpose Large Language Models (LLMs). Whereas new models have been proven useful for generic text handling, their applicability to domain-specific NLP tasks still remains doubtful, particularly because of the limited amount of dataset available in certain domains, such as Requirements Engineering. In this study, different pre-trained embeddings were tested in three requirements classification tasks, in search of a tradeoff between accuracy and computational complexity. The best F1-score results were obtained with BERT (90.36% and 84.23%), with DistilBERT identified as optimal tradeoff (90.28% and 82.61%).

## Keywords

Requirements Engineering, NLP, Large Language Models

## 1. Introduction

Natural Language Processing (NLP) is an area of Machine Learning (ML) which aims to learn, understand and generate human language content. More specifically, NLP is a set of techniques which are capable of representing written text at several levels of linguistic analysis with the goal of achieving near human-like levels of language processing for a given task or application [1]. The maturity level of Large Language Models (LLMs) reached in the past five years is having an enormous impact on Deep Learning (DL) based approaches for NLP. While, on the one hand, these models have made it possible to address previously unattainable NLP tasks, the ability of such general-purpose models on domain-specific contexts still poses some major challenges [1]. In particular, when applying NLP to domain-specific tasks, the amount of available text is usually extremely limited, and the semantic representation of words when used in a different context might be misleading [1]. Consequently, the research community looked at finetuning pre-trained LLM [2]. Whereas finetuning is a valid method, cases with limited amount of data hinder this approach.

Requirements Engineering (RE) is one such area where NLP approaches can help to improve processes. Requirements, within software development, are largely expressed in natural language [1]. The correct and accurate statement of requirements is essential for the development

---


*In: A. Ferrari, B. Penzenstadler, I. Hadar, S. Oyediji, S. Abualhaija, A. Vogelsang, G. Deshpande, A. Rachmann, J. Gulden, A. Wohlgemuth, A. Hess, S. Fricker, R. Guizzardi, J. Horkoff, A. Perini, A. Susi, O. Karras, A. Moreira, F. Dalpiaz, P. Spoletini, D. Amyot. Joint Proceedings of REFSQ-2023 Workshops, Doctoral Symposium, Posters & Tools Track, and Journal Early Feedback Track. Co-located with REFSQ 2023. Barcelona, Catalunya, Spain, April 17, 2023.*

✉ f.cruciani@ulster.ac.uk (F. Cruciani); s.moore2@ulster.ac.uk (S. Moore); cd.nugent@ulster.ac.uk (C. Nugent)

🆔 0000-0002-1870-0203 (F. Cruciani); 0000-0003-3205-3310 (S. Moore); 0000-0003-0882-7902 (C. Nugent)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

of high-quality software which meets the expectations of customers and end-users. Given the importance of this part of the software development lifecycle, it is necessary to ensure that requirements are stated clearly, adhere to quality criteria, are appropriately classified, and are free from errors. While it is possible, and often the norm, to carry out the requirements engineering processes manually, automated NLP approaches stand to offer significant improvement to the process [1]. Within requirements engineering, there are several areas where NLP approaches may be employed, including; elicitation, quality analysis, error detection, category classification, and traceability [1]. Although LLMs offer a general purpose approach to language modelling, they were not trained on the task of classifying requirements. In order to classify requirements into a given set of categories, a model must be trained to detect these categories, which is not typically the case for LLMs. As such, in order to develop an NLP solution for requirements classification, it is necessary to expose a model to a range of requirements and their associated categories during training, thereby requiring the development of a task-specific language model. This can be achieved either by fine-tuning the LLM on the specific task of requirements classification, as done in [2], or by using the LLM to provide a semantic representation of the requirement, and combining it with more traditional classifiers [3]. The first solution is more resource-consuming, while the second one is more efficient. To our knowledge, the literature does not provide a systematic comparison between different LLMs in this second scenario.

This paper seeks to assess the effectiveness of LLMs in accurately classifying requirements into their respective categories. In doing so, this paper considered pre-trained language models and evaluated their ability to create *semantic representations* from requirements specifications.

The contribution of this work can be summarized as follows:

- comparison of the semantic representational power from available pre-trained LLM with application to RE.
- an explorative study trying to optimize the tradeoff between computational resources and accuracy

The remainder of this paper is organized as follows. Section 2 summarizes the state of the art and related work in NLP tasks for RE. Section 3 describes the experiment design, the research questions, and the evaluation methodology. Results and discussion are reported in Sections 4 and 5 respectively. Finally, conclusions are drawn in Section 6.

## 2. Related Work

Among NLP tasks in RE, *requirement classification* is one of the most common [1]. In most cases, classification is applied to the binary case discriminating between functional (F) and non-functional requirements (NF) [4]. Other studies have also considered specific classes of NF requirements (e.g., usability, security)[5, 2, 6, 7]. In the earliest examples of requirements classification [5], sets of keywords obtained from manually labeled requirements were used to classify unseen data. More recently, studies started to explore the use of ML and DL approaches. In [6], BERT [8] was used in combination with a Graph Attention Network (GAT) and an Multilayer perceptron (MLP) classifier. The method was compared with other ML approaches, (including Naive Bayes, Random Forest (RF)) in two classification tasks: (i) the binary case F vs

**Table 1**  
PROMISE-NFR Dataset

Class	Requirements	#Sentences	#Words*
<b>Functional (F)<sup>+</sup></b>	<b>255</b>	<b>272</b>	<b>4996</b>
Availability (A)	21	29	432
Fault Tolerance (FT)	10	11	176
Legal (L)	13	15	215
<b>Look &amp; Feel (LF)<sup>+</sup></b>	<b>38</b>	<b>51</b>	<b>749</b>
Maintainability (M)	17	25	476
<b>Operational (O)<sup>+</sup></b>	<b>62</b>	<b>89</b>	<b>1231</b>
<b>Performance (PE)<sup>+</sup></b>	<b>54</b>	<b>69</b>	<b>1207</b>
Portability (PO)	1	2	25
Scalability (SC)	21	27	402
<b>Security (SE)<sup>+</sup></b>	<b>66</b>	<b>86</b>	<b>1265</b>
<b>Usability (U)<sup>+</sup></b>	<b>67</b>	<b>96</b>	<b>1508</b>
<b>Total</b>	<b>625</b>	<b>772</b>	<b>12682</b>

\* approximate number of words, <sup>+</sup> Used as Most frequent class

NF requirements, and (ii) for detecting four types of NF requirements. For an in-depth literature review on ML for requirement classification readers can refer to [1], which provides a holistic overview of the progress of NLP for performing RE tasks. On the other hand, this paper is more focused on identifying optimum representational power and analysing the computing resources required to achieve the task effectively.

Similar to [2], in this work we evaluated the use of BERT [8] for requirement classification extending the comparison to include other embeddings, such as GloVE [9], DIstilBERT [10], SBERT [11] and Universal Sentence Encoder (USE) [12]. It should be noted that, unlike other studies like [2], we did not retrain or fine-tuned the models used for embeddings, but simply aimed at comparing them in some classification tasks.

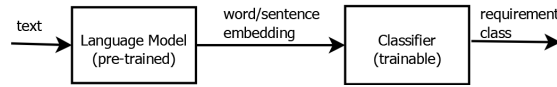
### 3. The experiment

The experiment aimed at comparing different pre-trained models for word and sentence embeddings to verify their suitability for requirements classification. Evaluation was conducted on the PROMISE-NFR dataset [13]. The dataset includes a set of 625 functional and non-functional requirements. The non-functional requirements set includes 11 different subclasses. Table 1 summarizes all the 12 classes, the number of requirements, sentences and words available per class. Despite being fairly balanced for the binary case F/NF requirements, the dataset presents a great challenge in terms of class imbalance when including all 12 classes (some consisting only of 1-20 requirements).

The word embeddings used in the experiments were GloVE [9], BERT [8], DIstilBERT [10], with dimension of 300, 768 and 768 respectively. The sentence embeddings were SBERT [11] and Universal Sentence Encoder (USE) [12] with size 384<sup>1</sup> and 512 respectively.

<sup>1</sup>SBERT was used with the all-MiniLM-L6-v2 model. See [https://www.sbert.net/docs/pretrained\\_models.html](https://www.sbert.net/docs/pretrained_models.html)

As illustrated in Fig. 1, the aim was to train a small size classifier (<10M parameters) on a domain-specific context with limited amount of data, relying on pre-trained language models for semantic representation of words/sentences.



**Figure 1:** In the experiment, different pre-trained models were used for extracting word/sentence embeddings without fine-tuning. Obtained embedding were used to train an ad-hoc classifier model.

All embeddings were tested under the same conditions, using two MLP structures ( $\approx 100k$  parameters and  $2M$  parameters)<sup>2</sup>. The MLP models were implemented using dense layers, with ReLU activation function and Stochastic Gradient Descent (SGD) optimizer. Learning rate values of 0.1 and 0.01 (default) were used. The embeddings and the MLP structures were evaluated in three different tasks:

**Task 1** Binary classification Functional / Non Functional Requirements

**Task 2** Classification of most frequent classes

**Task 3** Classification of all classes (except portability)

Task 1 compares these general-purpose embeddings without an extreme imbalance. Tasks 2 and 3 allow the evaluation to be made on the impact of class imbalance in the more complex cases distinguishing 6 and 12 classes. The 6 classes of Task 2 were chosen as the most frequent classes (comprised of at least 50 sentences and 500 words) as indicated in Table 1. The experiment aimed at answering the following research questions:

**RQ 1** Which embedding provides better accuracy in requirement classification?

**RQ 2** Which embedding provides the best trade-off between accuracy and model's complexity?

### 3.1. Evaluation Methodology

The dataset was split into train and test set, 70% and 30% respectively. With the training set further split into train and validation (10%). The splits were done using the stratify options, i.e. preserving class imbalance, and preventing from separating sentences appearing in the same requirement. The evaluation was done using a 5-fold procedure comparing the embeddings with the two MLP structures on the three tasks. Models were trained with early stopping using a patience of 25 epochs and saving only the models with highest accuracy in the validation set. The data imbalance was handled using weighted loss. Finally, a k-Nearest Neighbors (kNN) classifier was used as a baseline. Since kNN makes direct use of embeddings for classification, and because of its non-parametric nature, it is therefore a suitable approach to measure how well the embeddings obtained from pre-trained models could be used directly to classify requirements.

---

<sup>2</sup>The complete source code is available at: <https://github.com/fcruciani/reqclass>

**Table 2**

Classification Report (Task 1 &amp; 2) using the large and small MLP respectively.

	Precision*		Recall*		F1-score*		F1-Score <sup>+</sup>	
	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2
Glove	<b>0.9110</b>	0.7552	0.8520	0.7351	0.8709	0.7441	0.8840	0.7793
BERT	0.9010	<b>0.8373</b>	<b>0.8900</b>	<b>0.7743</b>	<b>0.8949</b>	<b>0.7994</b>	<b>0.9036</b>	<b>0.8423</b>
DistilBERT	0.9076	0.8043	0.8836	0.7645	0.8934	0.7807	0.9028	0.8261
SBERT	0.8709	0.7455	0.8797	0.7682	0.8748	0.7549	0.8837	0.7833
USE	0.8606	0.7484	0.8841	0.7666	0.8669	0.7550	0.8743	0.7948

\* Macro-average, <sup>+</sup>Weighted**Table 3**

Results with the small MLP architecture trained using BERT (uncased) embeddings (Task 2).

Class	Precision	Recall	F1-Score
Functional (F)	0.8490	0.9449	0.8944
Look & Feel (LF)	0.7273	0.5818	0.6465
Operational (O)	0.8500	0.8500	0.8500
Performance (PE)	0.9138	0.6543	0.7626
Security (SE)	0.8318	0.8318	0.8318
Usability (US)	0.8525	0.8062	0.8287
macro avg	0.8374	0.7782	0.8023
weighted avg	0.8456	0.8454	0.8416

## 4. Results

In the experiment, results covering all combinations between the large and small MLP classifiers were calculated on the three tasks. For the sake of conciseness only some combinations are reported. Additional results including confusion matrices are available in the published repository. Table 2 reports results obtained using the large MLP structure on Task 1 and the small MLP architecture on Task 2.

Additional combinations were tested considering the cased and uncased versions of pre-trained BERT and DistilBERT models. Table 3 report results obtained with the best performing model on Task 2, the small MLP architecture using the uncased version of BERT. Fig. 2 illustrates the confusion matrix and the normalized confusion matrix obtained in Task 2 using BERT uncased and the small MLP.

Table 4 summarizes results obtained in Task 3, including the baseline results using kNN as a classifier. Table 5 reports precision, recall and f-score values for all classes obtained with the best performing combination on Task 3. Fig. 3 illustrates the normalized confusion matrices obtained with the different embeddings on Task 3. Finally, Fig. 4 summarizes the macro-average F1 score obtained with all the embeddings on Task 2 and Task 3.

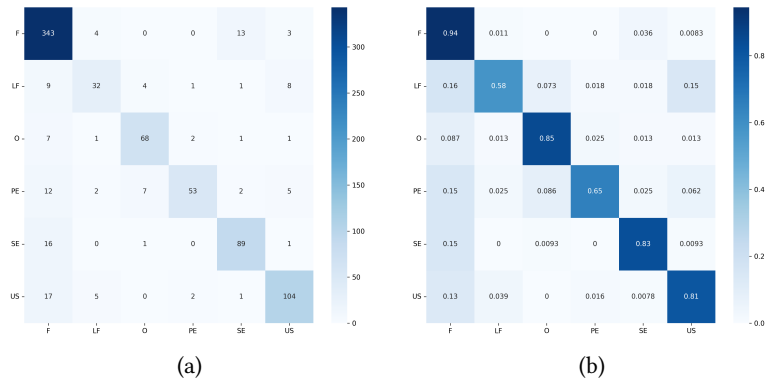


Figure 2: Confusion matrices obtained with BERT uncased (a) and the normalized version (b).

Table 4

Classification Report all classes (Task 3 using Large MLP)

	Precision*		Recall*		F1-score*		F1-Score <sup>+</sup>	
	MLP	kNN	MLP	kNN	MLP	kNN	MLP	kNN
Glove	0.5941	(0.5760)	0.5781	(0.3128)	0.5751	(0.3583)	0.6553	(0.4851)
BERT	<b>0.7585</b>	(0.7640)	0.5920	(0.5065)	0.6148	(0.5470)	0.7363	(0.6798)
DistilBERT	0.7417	(0.7347)	<b>0.6065</b>	(0.5027)	<b>0.6401</b>	(0.5544)	<b>0.7415</b>	(0.6760)
SBERT	0.6220	(0.6492)	0.5661	(0.5301)	0.5777	(0.5462)	0.6660	(0.6572)
USE	0.6221	(0.5890)	0.5472	(0.5113)	0.5615	(0.5216)	0.6860	(0.6583)

\* Macro-average <sup>+</sup>Weighted - in () are the baseline values obtained using kNN

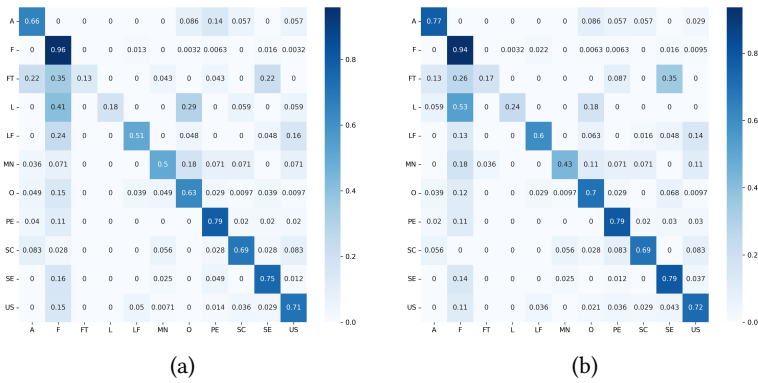


Figure 3: Confusion matrices obtained with BERT (a), and DistilBERT (b) on task 3.

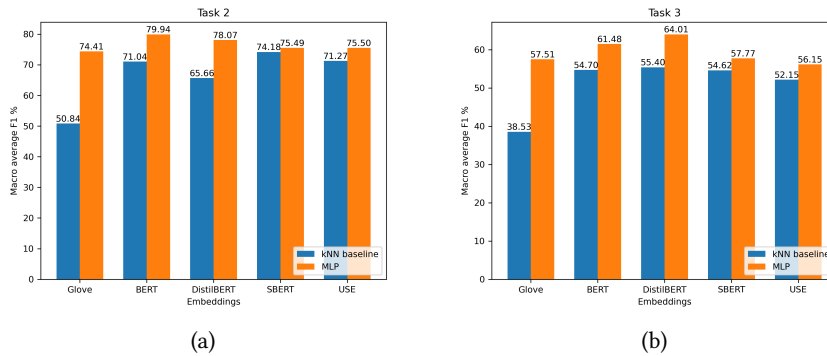
## 5. Discussion

Results on Task 1 highlight BERT and DistilBERT as the best performing embeddings (RQ1), however results obtained with the other embeddings are comparable and might be considered for resource constrained cases. In particular, SBERT is the fastest model (except for GloVe) for

**Table 5**

Results with the MLP architecture trained using BERT uncased embeddings.

Class	Precision	Recall	F1-Score
Availability (A)	0.6486	0.6857	0.6667
Functional (F)	0.7914	0.9397	0.8592
Fault Tolerance (FT)	0.8000	0.1739	0.2857
Legal (L)	0.8000	0.2353	0.3636
Look & Feel (LF)	0.6852	0.5873	0.6325
Maintainability (MN)	0.4800	0.4286	0.4528
Operational (O)	0.7363	0.6505	0.6907
Performance (PE)	0.8404	0.7980	0.8187
Scalability (SC)	0.7143	0.6944	0.7042
Security (SE)	0.6947	0.8148	0.7500
Usability (US)	0.7840	0.7000	0.7396
macro avg	0.7250	0.6098	0.6331

**Figure 4:** Macro F1 for all embeddings including the baseline kNN and the large MLP on Task 2 (a) and Task 3 (b).

generating vectors of 384 dimensions, which also reduces the complexity of the final classifier. Similarly, GloVe embeddings are obtained by simple lookup on a dictionary data structure and are 300 dimensions embeddings. GloVe, however, is exposed to out-of-dictionary (OOD) words limiting its working ability in the presence of OOD words. Similarly, in Task 2 and 3, BERT and DistilBERT were the best performing models, with DistilBERT a good candidate to reduce the computational overhead of BERT without causing detrimental effects on the accuracy performance (RQ2). No major differences were observed when using the small and the large MLP classifiers, possibly due to the limited size of the dataset that does not allow to maximize the benefit of using a classifier with a higher number of trainable parameters. The lack of data is further exacerbated in the case of sentence embeddings with the MLP trained on fewer data points. The comparison with the baseline highlighted how, despite the limited amount of data, training an MLP classifier outperforms the baseline kNN approach of using embeddings to classify new data. The worst performing baseline results were obtained with GloVe, possibly

attributable to out-of-dictionary words. MLP classifiers trained on GloVe vectors, however, appear to reduce the gap, leading to results comparable with SBERT and USE.

## 5.1. Limitations

*Construct Validity* Standard evaluation metrics were used as *macro-averages* to prevent majority classes from masking less represented ones. All mandatory steps of the ECSEER pipeline for evaluating classifiers were performed [14]. Optional steps, e.g. significance tests, were not performed due to the preliminary nature of this work.

*Internal Validity* One major factor affecting internal validity is the correctness of the annotation of the dataset that authors have questioned in the past. Nevertheless, it still represents the most widely used dataset for requirements classification, facilitating the comparison with previous work. Since this type of ML tasks typically include a high degree of randomness, 5-fold cross-validation was used to calculate results.

*External Validity* The dataset includes requirements written by students, which may not be representative of industrial requirements and the evaluation of the language models is limited to the three examined tasks. Different results may be obtained when other classification schemes are used, or other types of requirements-related information (e.g., user stories, or app reviews) are adopted. Concerning the coverage of possible pre-trained embeddings, we have considered a representative set of basic and deep learning-based ones, not only limited to those derived from BERT. Therefore, we argue that our analysis can be considered representative of the usage of different embeddings for requirements classification. As for the classification algorithm, we use two MLP structures and a kNN as a baseline. Different results may be obtained when using other classifiers (e.g., SVM, Naive Bayes).

## 6. Conclusion

This paper reported on the evaluation of pre-trained embeddings for RE. Some of the most common embeddings were tested under the same circumstances on a public dataset. Results obtained identify BERT and its smaller variant DistilBERT as the best performing embeddings, with the latter being an optimal tradeoff between accuracy and model complexity. GloVe and SBERT despite a slightly lower accuracy were found to be the fastest in prediction time and could be suitable for cases in which time represents a key factor or resource constrained environments. Future work will aim to extend the evaluation on additional datasets to verify the validity of these results on different RE tasks and datasets.

## Acknowledgments

This research is supported by the ARC (Advanced Research Engineering Centre) project, funded by PwC<sup>3</sup> and Invest Northern Ireland.

---

<sup>3</sup>PricewaterhouseCoopers LLP a limited liability partnership incorporated in England with its registered office office at 1 Embankment Place, London WC2N 6RH



## References

- [1] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, R. T. Batista-Navarro, Natural language processing for requirements engineering: A systematic mapping study, *ACM Computing Surveys (CSUR)* 54 (2021) 1–41.
- [2] T. Hey, J. Keim, A. Koziolok, W. F. Tichy, Norbert: Transfer learning for requirements classification, in: *2020 IEEE 28th International Requirements Engineering Conference (RE)*, IEEE, 2020, pp. 169–179.
- [3] M. E. Peters, S. Ruder, N. A. Smith, To tune or not to tune? adapting pretrained representations to diverse tasks, in: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, 2019, p. 7–14.
- [4] F.-L. Li, J. Horkoff, J. Mylopoulos, R. S. Guizzardi, G. Guizzardi, A. Borgida, L. Liu, Non-functional requirements as qualities, with a spice of ontology, in: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, IEEE, 2014, pp. 293–302.
- [5] J. Cleland-Huang, R. Settimi, X. Zou, P. Solc, Automated classification of non-functional requirements, *Requirements engineering* 12 (2007) 103–120.
- [6] G. Li, C. Zheng, M. Li, H. Wang, Automatic requirements classification based on graph attention network, *IEEE Access* 10 (2022) 30080–30090.
- [7] O. AlDhafer, I. Ahmad, S. Mahmood, An end-to-end deep learning system for requirements classification using recurrent neural networks, *Information and Software Technology* 147 (2022) 106877.
- [8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).
- [9] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [10] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, *arXiv preprint arXiv:1910.01108* (2019).
- [11] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, *arXiv preprint arXiv:1908.10084* (2019).
- [12] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al., Universal sentence encoder, *arXiv preprint arXiv:1803.11175* (2018).
- [13] H. L. J. Cleland-Huang, S. Mazrouee, D. Port, Promise-nfr dataset, <https://doi.org/10.5281/zenodo.268542>, 1007.
- [14] D. Dell’Anna, F. B. Aydemir, F. Dalpiaz, Evaluating classifiers in se research: the ecser pipeline and two replication studies, *Empirical Software Engineering* 28 (2023) 1–40.