An Overview of User Feedback Classification Approaches

Rubens Santos, Eduard C. Groen, Karina Villela
Fraunhofer IESE, Kaiserslautern, Germany
rubens.santos@iese-extern.fraunhofer.de;
{eduard.groen; karina.villela}@iese.fraunhofer.de

Abstract

Online user feedback about software products is a promising source of user requirements. To allow scaling analyses to large amounts of user feedback, research on Crowd-based Requirements Engineering (CrowdRE) seeks to tailor natural language processing (NLP) techniques to Requirements Engineering (RE). Various frameworks have been proposed, but it remains largely unclear why particular NLP techniques are better suited for CrowdRE than others, which makes it hard to make a well-founded choice for a technique. We found that CrowdRE research most often uses machine learning (ML) and has so far applied twelve clusters of ML algorithms and seven clusters of ML features. The prevalent algorithm–feature pair is Naïve Bayes with Bag of Words – Term Frequency (BOW-TF), followed by Support Vector Machines (SVM) with BOW-TF. An initial comparison of the reported precision and recall suggests that classifications in RE need further improvement. Our research presents a preliminary overview of the current landscape of automated classification techniques for RE whose results may inspire researchers to apply new strategies to advance research in this field, or to include ML models they had not considered previously in their benchmarks.

1 Introduction

Software products today typically have many geographically dispersed stakeholders who together form a large heterogeneous group of (potential) end-users known as a "crowd" [GSA+17]. Using traditional requirements engineering (RE) approaches, such as interviews, focus groups, questionnaires, and field observations to elicit and validate requirements from a representative sample of such a crowd faces scalability problems. The discipline within RE that investigates solutions for how to derive validated requirements from the crowd by motivating its members to provide feedback, analyzing their feedback and behavior, and validating the outcomes is generally known as Crowd-based RE (CrowdRE) [GSA+17].

One promising approach within CrowdRE is the use of online user feedback about software products for RE. This kind of feedback is widely and often publicly available on review platforms, social media, and tracking systems and forms a valuable resource for useful information from which potential software improvements and changed or new requirements can be derived [GSA+17]. However, because the crowd providing this feedback is large, the amount of information they provide can quickly become too abundant to be analyzed manually [GSK+18], causing similar scalability problems as those observed with traditional RE techniques. This is why

CrowdRE research is particularly interested in applying automation techniques to identify requirements-relevant content in user feedback from the crowd in order to make this activity scalable. To this end, techniques from the domain of natural language processing (NLP) are being considered, especially so-called "classifiers". Classification approaches provide an automated solution for solving a text classification problem (in our case, the classification of user feedback for RE) through *algorithms* that place text snippets into predetermined categories with a certain degree of confidence. The algorithms used most often recently for text classification are supervised machine learning (ML) algorithms, which can identify patterns in labeled data and generalize it to unlabeled data with a certain degree of accuracy. However, because these algorithms were not originally designed for classifying text, they need to be paired with a primary ML *feature* that represents the text as vectors from which the algorithms can deduce patterns. We refer to the combination of an algorithm with a feature as an *ML model*.

Currently, it is unclear what the most suitable approaches are for analyzing online user feedback. For one, most research on classifying user feedback in RE stems from recent years, with many questions still left to be answered. A further challenge is that the user feedback is typically obtained from sources that are not aimed at collecting requirements, such as software distribution platforms, social media, and knowledge-sharing websites, which makes the data inherently ambiguous for RE and the analysis very different from that of other NLP applications in RE, such as the analysis of requirements specifications. Frameworks that have been proposed to date for classifying user feedback for RE differ in both their focus and their choice of technique(s). For example, *AR-Miner* filters out non-informative feedback using Expectation Maximization for Naïve Bayes (NB) [CLH+14] and *ALERTme* models topics through multinomial NB to help select or rank the most informative user feedback [GIG17]; others classify sentiments and extract topics through Latent Dirichlet Markov Allocation to identify common concerns among users [KGM16], or apply Term Frequency – Inverse Document Frequency (TF-IDF) and χ^2 to rank the most problematic software features for usability and user experience analysis [HS13]. The varying performance of these frameworks in different contexts makes it difficult for both researchers and practitioners to choose one that is suitable for their purpose. Moreover, the authors often do not explain why they selected a particular technique.

In this paper, we therefore seek to present a preliminary comparison of the classification techniques used to date in CrowdRE research. We are aware that based on our results, conclusions about the suitability of a technique for different purposes can only be drawn to a limited degree, but we assert that our work can provide some initial insights into the current trends and help us assess how promising these techniques are from an RE perspective. Our aggregated overview of classifiers in CrowdRE is intended as a first step towards understanding why particular combinations of algorithms and features are better suited for RE than others. To achieve this, we seek to answer the following research questions:

RQ1: Which automated techniques have been used in research to classify requirements-relevant content in user feedback?

RQ2: Which classification algorithms and features have been used most often?

RQ3: Which algorithm–feature pairs have yielded the highest precision and recall values?

We first describe how we identified existing literature and assessed their use of NLP techniques in Section 2, and then present the outcomes in Section 3. In Section 4, we discuss these results, concluding in Section 5.

2 Method

Within the scope of a larger benchmarking study, we initiated a systematic literature review (SLR) [KC07] to obtain a comprehensive and broad overview of the literature on classifying user feedback, while excluding any potential selection bias and preventing gaps in our research. As part of this effort, we noticed that our set of systematically obtained literature works proposed and used many disjunct classification structures and categories. This finding led us to investigating through a systematic mapping study [PFMM08] which approaches were used and how effective they were, resulting in the preliminary comparison presented in this work. We defined the following objectives for our mapping study:

- **Overall Objective:** What are the state-of-the-art automated approaches for assisting the task of requirements extraction from user feedback acquired from the crowd, and which NLP techniques and features do they use?
- **Objective 1:** Regarding requirements elicitation from user feedback acquired from the crowd, what are the state-of-art automated approaches for classifying user feedback?
- **Objective 2:** How do such approaches classify user feedback?
  - **Objective 2.1:** What are the different sets of categories in which user feedback is classified?

– **Objective 2.2:** Which automated techniques are used?
– **Objective 2.3:** What are the characteristics of the user feedback these approaches aim to classify?

To perform our search, we composed a search string by defining search terms, many of which are common terms from known literature, and tested these in different combinations. We also used previously identified papers to verify whether the search string would correctly find these publications. The final search string was as follows:

> (("CrowdRE" OR "Crowd RE") OR (("User Review" OR "User Feedback" OR "App Review" OR "Feature Requests" OR "User Opinions" OR "User Requirements")) AND (Classif* OR Framework OR Tool OR "Text Analysis" OR Mining OR "Feature Extraction") AND "Requirements Engineering"))

We selected papers according to the eight exclusion criteria (EC) and two inclusion criteria (IC) listed below. A paper meeting one or more ECs was excluded from the selection, while a paper meeting one or more ICs and no ECs was included in the selection.

- **EC1:** The paper is not written in English.
- **EC2:** The paper was published before 2013.[1]
- **EC3:** The work or study is not published in a peer-reviewed venue.
- **EC4:** The paper is not related to RE for software products and/or the title is clearly not related to the research questions.
- **EC5:** The paper does not address the topic of requirements extraction from user feedback analysis.
- **EC6:** The paper proposes a tool or dataset that does not aim to assist a requirements extraction process from online user reviews, or could not be used in this way; for example, recommender systems, information retrieval for search engines, or approaches that link source code changes to bug fixes.
- **EC7:** The paper proposes an approach or tool that does not process textual user feedback. For example, approaches that analyze implicit feedback, process requirements documents, or merely collect user feedback instead of processing it.
- **EC8:** The paper proposes an approach that does not make use of automation because the user feedback analysis is done entirely manually; for example, crowdsourced requirements elicitation.
- **IC1:** The paper proposes an approach for filtering out irrelevant user feedback from raw data, regardless of whether or not this is done using classification techniques.
- **IC2:** The paper proposes an approach for classifying user feedback into default predetermined categories.

We first applied our search string to search for suitable papers in March 2018, using three prominent scientific databases on software engineering research: ACM, Springer, and IEEE Xplore. Exclusion criteria EC1–EC3 were applied directly through database filters. This query returned a combined result of 1,219 papers. After removing duplicates and screening the title and abstract (to which EC4–EC8 and IC1–IC2 were applied), 146 papers remained. These included papers for which the results of our title and abstract analysis were inconclusive, so this number also included papers where we were uncertain whether they matched our selection criteria. Further scanning of the introduction and conclusion sections, to which EC4–EC8 were re-applied, reduced the number of papers for data extraction to a total of 40. This work was performed by the first author of this paper, and the third author cross-checked a random subset to assure the quality of this work. Any disagreements were discussed and resolved. We repeated the query on 18 December 2018 to include papers that had been added over the course of 2018, which resulted in 14 new papers, of which 3 were relevant to our SLR, for a total number of 43 analyzed papers. Due to space restrictions, we present the complete list of primary papers in a separate document[2], and reference them in this document with the notation Pn.

To satisfy the overall goal of preparing a benchmarking study, we systematically extracted three major groups of data from the selected papers:

- *Dataset-related information*, such as dataset size in number of entries, object granularity (sentence vs. review), source (e.g., app stores, social media), and mean text object size.
- *NLP techniques applied*, such as algorithms, parsers, ML features, and text pre-processing techniques.
- *Classification categories* into which the tool was designed to classify user feedback, along with their definitions, where available. We also paid special attention to any explicit rationales behind design decisions made for a tool to understand for which goal or under which circumstances specific categories are best used.

[1] This year was chosen because prior to the introduction of CrowdRE in 2015 [GDA15], the analysis of online user feedback via NLP for RE was not considered to be a serious source of requirements. We additionally considered six years as a technical obsoleteness threshold to fit our paper selection efforts to our resource constraints.

[2] Bibliography of primary studies: zenodo.org/record/2546422, doi:10.5281/zenodo.2546422

The results from the second group of data, "NLP techniques applied", was a list with many parameters. We sought to structure these findings to find commonalities between the set-ups presented in the papers that can be used as a basis for our future benchmarking study. To assess and map the NLP techniques applied, we based a systematic mapping study on these SLR results. Because nearly all papers appeared to use a set-up using ML algorithms, in the mapping process we categorized the parameters of each set-up according to the following five aspects:

- *ML algorithm*, such as Naïve Bayes, Logistic Regression, Support Vector Machines, or Decision Trees.
- *Primary ML features* that represent the text according to word count or related methods, such as Bag of Words (BOW), Term Frequency - Inverse Document Frequency (TF-IDF), and Bag of Frames (BOF).
- *Secondary ML features* that represent specific aspects of the text or metadata, such as length (found in P14), sentiment score (found in P25), or star rating (found in P34). Secondary ML features yield low scores when used on their own, but can help achieve greater efficiency and quality when used in combination with primary features.
- *Semi-supervised classification algorithms*, such as Expectation-Maximization, Self-Training, or Rasco.
- *Pre-processing techniques*, such as stop words removal, synonym unification, stemming, lemmatization, special characters removal, abbreviation transformation, and negation handling.

The distinction between primary and secondary ML features is not always clear-cut and can sometimes involve a specific (enhanced) variation on a primary ML feature. As a decision rule, we considered variations of particular primary ML features independent of whether these were compared within the same paper. For example, n-Gram, NLP Heuristics, and Augmented User Reviews BOW (AUR-BOW) can be interpreted as combinations of a secondary feature with BOW, and the feature selection technique χ^2 is also applied in combination with BOW in P24.

We subsequently tabulated these results according to the ML algorithm, juxtaposed with the other aspect that is always used in combination with an algorithm: a primary ML feature. In the cells, we then cited the papers using this algorithm–feature pair. To each citation, we added footnotes to indicate additional details of the set-ups from the remaining three categories because they can be used in many different combinations. A set-up can include any number of pre-processing techniques and secondary ML features, or none, and can optionally choose to make use of a semi-supervised classification algorithm. Combined, the aggregation resulted in an intricate matrix that takes all these variations into account. We will leave a discussion of these variations for later stages of our benchmarking study because it would make the presentation of our results too lengthy and complex.

3 Results

In this section, we will begin with a presentation of the classification approaches found in research (Section 3.1), followed by a comparison of the ML models applied to identify "Feature Requests" (Section 3.2).

3.1 Prevalent Classification Approaches

Among the 43 papers that suggest or employ a type of user feedback classification approach, ML clearly forms the predominant group of approaches, being applied in 37 (86%) of the primary studies analyzed. Only six papers, P11, P18, P19, P21, P27, and P33, propose approaches based exclusively on a dictionary, regular expressions, or a parser combined with heuristic rules. Table 1 presents a simplified version of the aggregated table discussed in Section 2, from which details about the use of a semi-supervised classification algorithm, pre-processing techniques, and secondary ML features in the set-up have been omitted, both for space reasons and in order to present a comprehensive overview of results focusing on the key aspects of the ML models themselves. In the table, we have broken the ML models down into their algorithm–feature pairs to better understand which classifiers are used. In the literature, we found a total of 139 such pairs. The table clusters algorithms and features that belong to the same category (e.g., the Decision Tree algorithm C4.5 includes its Java implementation J48; NB includes both its binary and multinomial versions). Most papers included several ML models ($M = 3.8, Min = 1, Max = 14$) because they reported on comparative experiments.

We found seven clusters of ML algorithms being applied: NB (found in 27 publications), Support Vector Machines (SVM; 22), Decision Trees (DT; 17), Logistic Regression (LR; 12), k-Nearest Neighbors (k-NN; 3), Bayesian Network (1), and Neural Networks (1). Decision Trees could be further subdivided into five distinct algorithms, for a total of 12 algorithm clusters. We also found four main ML features among the studies: BOW / Term Frequency (found in 23 publications), TF-IDF (12), NLP Heuristics (8), n-Gram (5), χ^2 (1,

in combination with three algorithms), Bag of Frames (BOF; 1, in combination with two algorithms), Binary BOW (1; in combination with one algorithm), and AUR-BOW (1, in combination with 3 algorithms). Although theoretically any algorithm–feature pair is possible, we found 34 in the literature, 12 of which were used only once. The most frequently found pairs are: NB + BOW (found in 15 publications); SVM + BOW (11); NB + TF-IDF (8), and LR + BOW (7).

Table 1: ML algorithm–technique pairs applied, showing the ML algorithms in the rows, the ML features in the columns, and the pairs in the cells, which reference the papers using that pair.

Algorithm \ **Feature**			BOW TF	BOW Bool	BOF	TF-IDF	χ^2	n-gram	NLP Heuristics	AUR BOW
Naïve Bayes			P1 P3 P8 P9 P10 P13 P20 P22 P24 P25 P28 P31 P32 P35 P36 P41 P42 P43		P20	P5 P6 P14 P15 P16 P24 P26 P40	P24	P28 P37	P7 P9 P10 P28 P35 P36	P24
Bayesian Network			P43							
Logistic Regression			P8 P9 P10 P22 P25 P28 P35 P36 P42			P14 P23		P12 P28	P9 P10 P23 P28 P35 P36	
k-Nearest Neighbors						P6 P5 P40				
Support Vector Machines			P1 P8 P9 P10 P17 P20 P22 P28 P29 P31 P34 P35 P36 P41 P42	P17	P20	P2 P5 P6 P13 P14 P26		P28	P7 P9 P10 P29 P35 P36	
Decision Trees	Single	C4.5	P9 P10 P24 P29 P35 P36			P5 P6 P13 P24 P26 P40	P24		P9 P10 P29 P35 P36	P24
		Unspecified	P25 P28 P30 P42					P28 P30	P28	
	Multiple	Boosted	P9 P10 P35 P36			P4			P9 P10 P35 P36	
		Random Forest	P28 P29 P39 P42					P28 P38	P28 P29	
		Bagging	P24			P24 P40	P24			P24
Neural Networks						P14				

3.2 Classifier Performance

To compare the quality of the user feedback analysis approaches in spite of papers having different classification purposes, we limited our comparison to those that used the most common classification category: "Feature Request". In total, we found 51 reported precision and recall values and 10 F_1-measures on 30 different ML models, reported in 19 papers. Seven of these papers provided precision and recall or F_1-measure values for one ML model: P8, P12, P15, P25, P34, P38, and P39. The other twelve papers, P1, P7, P13, P14, P20, P24, P26, P28, P31, P35, P40, and P41, provided values for two to twelve ML models. For 22 ML models, only one measurement was available; the largest number of measurements was for NB + BOW, with nine measurements. One paper, P24, compared twelve ML models and found AUR-BOW to yield a slightly better performance than other ML features; in P35, BOW with NLP Heuristics was found to perform better than BOW alone across all five algorithms these two features were paired with. Of the six papers that suggested non-ML approaches, two papers also provided precision and recall values for "Feature Request": 0.95/0.87 in P18, and 0.91/0.89 in P19, but these results could not be regarded as part of our comparison of ML models.

Table 2: Evaluation metrics of ML models on "Feature Requests", showing the ML algorithms in the rows, the ML features in the columns, and the precision and recall values in the cells, along with a source reference. Single values represent F_1 values from P1, P12, and P28, for which we could not obtain precision and recall. A paper citation in *italics* represents an averaged evaluation metric across all supported classes within that paper.

Alg. \ **Feat.**	BOW	BOF	TF-IDF	χ^2	n-Gram	NLP Heur.	AUR-BOW
Naïve Bayes	0.77 *P1* 0.72/0.90 P8 0.28/0.62 P20 0.67/0.64 *P24* 0.71/0.79 P25 0.20/1.00 P28 0.89/0.81 P31 0.67/0.58 *P35* 0.77/0.58 P41	0.41/0.73 P20	0.73/0.18 P14 0.74/0.78 P15 0.67/0.65 *P24* 0.32/0.38 *P26* 0.63/0.63 P40	0.67/0.65 *P24*	0.27 P28	0.79/0.77 *P7* 0.69/0.69 *P35*	0.72/0.65 *P24*
Logistic Regression	0.76 P28 0.46/0.46 *P35*		0.68/0.26 P14		0.59 *P12* 0.84 P28	0.49/0.49 *P35*	
k-NN			0.55/0.53 *P40*				
SVM	0.84 *P1* 0.45/0.68 P20 0.51/0.85 P28 0.84/0.79 P31 0.65/0.49 P34 0.59/0.61 *P35* 0.72/0.60 P41	0.62/0.96 P20	0.74/0.76 *P13* 0.69/0.33 P14 0.66/0.65 *P26*		0.70 P28	0.79/0.78 *P7* 0.68/0.68 *P35*	
Decision Trees: Single Tree	0.66/0.67 *P24* 0.57/0.58 *P35*		0.78/0.74 *P13* 0.66/0.68 *P24* 0.53/0.51 *P26* 0.57/0.58 *P40*	0.66/0.68 *P24*	0.06/0.83 P28	0.75/0.74 *P35*	0.72/0.65 *P24*
Decision Trees: Boosted	0.62/0.61 *P35*					0.71/0.71 P35	
Decision Trees: Random Forest	0.67/0.87 P39				0.63/0.84 P38		
Decision Trees: Bagging	0.69/0.69 *P24*		0.69/0.70 *P24* 0.62/0.60 *P40*	0.69/0.70 *P24*			0.63/0.59 P24
Neural Network			0.71/0.39 P14				

Table 3: Evaluation metrics of ML models on "Feature Requests", showing the ML algorithms in the rows, the ML features in the columns, and the F_β-measures with the associated β_T values in the cells, along with a source reference. *Italic values* represent an averaged evaluation metric across all supported classes within a paper.

Alg. \ Feat.	BOW	BOF	TF-IDF	χ^2	n-Gram	NLP Heur.	AUR-BOW
Naïve Bayes	0.89 $_{14.72}$ P8 0.57 $_{3.66}$ P20 0.64 $_{7.14}$ *P24* 0.79 $_{14.72}$ P25 0.86 $_{4.83}$ P28 0.81 $_{5.36}$ P31 0.58 $_{6.52}$ *P35* 0.61 $_{2.10}$ P41	0.69 $_{3.66}$ P20	0.18 $_{11.26}$ P14 0.78 $_{2.38}$ P15 0.65 $_{7.14}$ *P24* 0.37 $_{3.24}$ *P26* 0.59 $_{2.79}$ P40	0.65 $_{7.14}$ *P24*		0.77 $_{15.64}$ *P7* 0.69 $_{6.52}$ *P35*	0.65 $_{7.14}$ *P24*
Logistic Regression	0.46 $_{6.52}$ *P35*		0.26 $_{11.26}$ P14			0.49 $_{6.52}$ *P35*	
k-NN			0.53 $_{2.79}$ *P40*				
SVM	0.66 $_{3.66}$ P20 0.83 $_{4.83}$ P28 0.79 $_{5.36}$ P31 0.49 $_{4.94}$ P34 0.61 $_{6.52}$ *P35* 0.62 $_{2.10}$ P41	0.92 $_{3.66}$ P20	0.76 $_{10.00}$ *P13* 0.33 $_{11.26}$ P14 0.65 $_{3.24}$ *P26*			0.78 $_{15.64}$ *P7* 0.68 $_{6.52}$ *P35*	
Decision Trees: Single Tree	0.67 $_{7.14}$ *P24* 0.58 $_{6.52}$ *P35*		0.74 $_{10.00}$ *P13* 0.68 $_{7.14}$ *P24* 0.51 $_{3.24}$ *P26* 0.58 $_{2.79}$ *P40*	0.68 $_{7.14}$ *P24*	0.54 $_{4.83}$ P28	0.74 $_{6.52}$ *P35*	0.65 $_{7.14}$ *P24*
Decision Trees: Boosted	0.61 $_{6.52}$ *P35*					0.71 $_{6.52}$ P35	
Decision Trees: Random Forest	0.86 $_{5.59}$ P39				0.82 $_{3.93}$ P38		
Decision Trees: Bagging	0.69 $_{7.14}$ *P24*		0.70 $_{7.14}$ *P24* 0.60 $_{2.79}$ *P40*	0.70 $_{7.14}$ *P24*			0.59 $_{7.14}$ P24
Neural Network			0.39 $_{11.26}$ P14				

In order to perform comparative analyses, we needed the precision and recall values for all measurements, so we requested these from the authors of the four papers that reported F_1-measures; the authors of P40 provided these values, the authors of P1 and P28 could not provide them, and the authors of P12 did not respond. The resulting 54 precision–recall pairs and seven F_1-measures are shown in Table 2; the F_1 measures were omitted from further analysis. This especially affected the measurements provided for the ML model n-Gram, with four out of six measurements being excluded. Overall, moderate values were reported on both precision ($M = 0.63$; $SD = 0.15$; $Med = 0.67$) and recall ($M = 0.65$; $SD = 0.16$ $Med = 0.66$). P28 provided a measurement on NB + BOW where both the precision and recall values were outliers ($M \pm 2SD$), as well as an outlier value for precision on DT–Single Tree + n-Gram. Further negative outliers were found for precision on NB + BOW in P20 and for recall on NB + TF-IDF and LR + TF-IDF in P14.

Due to the differences between the studies, we can only draw preliminary conclusions about the suitability of the ML algorithms and primary ML features. In order to use an objective measurement for comparing the performance of the classifiers, we calculated a weighted version of the F-measure, F_β. This is a harmonic average that accounts for the relative importance of recall (R) and precision (P) by using a variable β value that is determined by the contextual characteristics of either the task or the tool (cf [Ber17]). The β value in this formula was determined separately for each paper. Because information about the tool's performance was not available in the papers we assessed, we instead determined the task-based β_T, calculated as $\frac{1}{\lambda}$, where λ is the

fraction of identified "Feature Requests" in the dataset. This assumes that the less frequent the true positives are in a document, the more challenging the task of finding them is for human actors [Ber17]. The average β_T was found to be 6.96 ($SD = 4.57$; $Med = 5.36$; $Min = 2.10$; $Max = 15.64$) across papers, which means that greater emphasis was placed on the reported recall values overall. We then calculated the F_β-measure for each precision–recall pair using the following formula:

$$F_\beta = (1 + \beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R}$$

This resulted in a total of 54 F_β-measures ($M = 0.64$; $SD = 0.15$; $Med = 0.65$). The F_β-measures and the β_T values on which they are based are presented in Table 3. The highest F_β values were 0.92 for SVM + BOF in P20, 0.89 for NB + BOW in P8, and 0.86 for DT–RF + BOW in P28. The lowest were 0.18, 0.26, and 0.33, for various algorithms in combination with TF-IDF reported in P14. Although the two lowest values are outliers, we chose to retain them in our analysis.

The average F_β values per ML algorithm showed that DT–RF ($F_\beta = 0.84$; 2 measurements) and SVM (0.68; 12) scored best, while Neural Networks (0.39; 1) and LR (0.40; 3) scored poorest. The ML features BOF (0.81; 2) and NLP-Heuristics (0.70; 7) had the highest average F_β values, and TF-IDF (0.55; 17) and AUR-BOW (0.63; 3) the lowest. The best and worst average F_β score per ML model (1–8 measurements) were all based on single measurements; SVM + BOF (0.92 in P20), DT–RF + BOW (0.86 in P39), and DT–RF + n-Gram (0.82 in P38) performed best, while LR + TF-IDF (0.26 in P14), NN + TF-IDF (0.39 in P14), and LR + BOW (0.46 in P35) performed worst. This shows that the highest individual measurements found earlier for ML algorithms and ML features were diminished by specific lower reported performances. This finding is corroborated by the finding that the prevalent ML models, those for which we found three or more measurements in the literature, all ranked between 7 and 25 ($F_\beta = 0.51 - 0.72$) due to the large differences in their performance across settings (e.g., NB + BOW: 8 measurements, F_β ranging from 0.57 in P20 to 0.89 in P8, rank 7).

4 Discussion

In recent years, research on CrowdRE has used various approaches to classify user feedback for RE. Some publications suggest the use of dictionary-based approaches, regular expressions or parsing, but the vast majority of approaches use ML models, consisting of ML algorithms paired with ML features **(RQ1)**. Based on this finding, we further investigated the specific classification algorithms and features. The most popular algorithms were found to be NB, SVM, LR, and DT (especially Single Tree), while BOW and TF-IDF were almost always used as ML features. The models NB + BOW, SVM + BOW, and NB + TF-IDF were found most often in research **(RQ2)**. A common denominator of the prevalent ML algorithms is that they provide a relatively large degree of control over the supervised ML, while the ML features BOW and TF-IDF are likely popular because of their versatile nature. They also share a similar degree of tool support, and are often already familiar to the researcher.

To assess the performance, we compared the ML models' F_β-measures. We found that SVM + BOF, NB + BOW, DT-RF + BOW, and NB + BOW each achieved an F_β-measure above 0.85 in a single study **(RQ3)**, but when other measurements were available, their averaged score would be lower than that of other average F_β-measures. This is in line with our finding that overall, the precision, recall, and F_β metrics were surprisingly moderate if we assume that the phenomenon of publication bias usually promotes the reporting of positive results, paired with the best possible outcomes of the analyses performed. All of the popular ML algorithms used can potentially result in good-quality results as well as poorer results. In this preliminary analysis we did not investigate how particular study characteristics might have contributed to better or poorer performance of the ML models, but the strong variance in the performance of the ML models does suggest that other factors have a strong impact on the classification efficiency. These characteristics include the goals and problems the respective study sought to address; the type, quality, and size of the dataset and gold standard; and the choice of classification categories and additional ML techniques (e.g., semi-supervised classification algorithms, pre-processing techniques, and secondary ML features; see Section 2). Moreover, the great diversity in the total number of different algorithm–feature pairs found and the strong variance in study set-ups suggests that the research on user feedback analysis is still exploring appropriate ML models.

We assert that the choice of appropriate ML algorithms and ML features may need to play a larger role in future research on classifying user feedback for RE: Some works made a strong case for NLP Heuristics and n-Grams that can introduce context information into the classification task, which under some circumstances

leads to better results. Moreover, the CrowdRE literature has proposed two adaptations of BOW with promising results: BOF in P20 and AUR-BOW in P24. However, none of these directions have so far been pursued in subsequent research, suggesting that this field could benefit from a more thorough exchange on the technologies applied. Support for these features in ML solutions such as Weka [HFH+09] or Scikit-learn [PVG+11] could facilitate wider use. Similarly, the use of a Bayesian Network as in P43, or the use of a Deep Learning approach such as the Neural Networks in P14, could be replicated in further studies. In addition to ML approaches, the results of several works on non-ML approaches for RE suggest that carefully designed heuristics may in some cases also provide accurate results (i.e., high precision). However, we asserted in P11 that high recall values are harder to achieve because it is challenging to create enough rules.

In terms of threats to validity, the exclusion of other influencing study characteristics in our mapping may have limited our ability to address confounding variables influencing the variance in the reported precision and recall values. By following the procedure of a systematic mapping study [PFMM08], we claim to have achieved strong internal validity and to have covered the publications in this domain well. Conclusion validity may be affected by the heterogeneity of the publications found; by focusing on the precision and recall values for classifications of "Feature Requests" using ML models, we were only able to compare 16 of 43 papers. In terms of external validity, our work might not be generalizable to areas outside of user feedback analysis in RE because we specifically investigated the classification techniques used within CrowdRE.

5 Conclusion & Future Work

In this work, we presented the preliminary results of a systematic mapping study regarding the classification techniques used in CrowdRE studies that sought to classify user feedback for RE using NLP techniques. The field is strongly dominated by ML, for which we identified the twelve clusters of ML algorithms and seven clusters of ML features that have been used to date. A comparison of the reported results, however, shows a varied outcome. It was not possible to draw decisive conclusions from the data about which ML models are most suitable for classifying user feedback, because factors other than the ones we considered in this work appear to have had a strong influence on the performance of the ML models. However, we did find that good results have been attained with the most often used ML algorithms, especially when used in combination with appropriate primary and secondary ML features.

We have seen that the current landscape is still one of exploration into the most suitable techniques. However, progress is hindered by a certain lack of cross-fertilization, where researchers pick up on promising findings in other works to investigate whether these approaches work well in their context. Most notably, this pertains to adaptations of ML features such as BOF and AUR-BOW, which were specifically adapted for CrowdRE research but have not yet found any application in research by others. Similarly, several other ML features and even some ML algorithms have shown promise in merely a single publication. For research on user feedback analysis for RE to progress, it may even be indispensable to evaluate these techniques for different purposes in comparative studies and benchmarks. On the other hand, the potential of non-ML approaches reported in some works should not be ignored either when investigating new solutions for user feedback classification.

This work was descriptive in nature and was limited to a comparison of only the ML algorithms and primary ML features. Future work should assess which other study-specific aspects (e.g., dataset, classification categories, other ML techniques; see Section 4) impact the performance of a classification approach when analyzing user feedback. This could lead to more prescriptive results regarding the most suitable kinds of classification techniques and the rationale for choosing a particular configuration. Although our work focused on the analysis of user feedback, the performance of ML models in other contexts within and outside of RE may also reveal which techniques are most appropriate. We found that most publications do not provide any explanation for their choice of techniques, even though the motives could be an inspiration to others. Moreover, we see the need for an elaborate comparison or benchmarking of classification techniques for the purposes of RE. A thorough comparison of potential ML models is lacking, with only a few papers having compared up to ten ML models for one particular purpose. Although the only study in this domain that we found to apply Deep Learning, P14, did not yield convincing results, this may be a line of research that could be pursued further in coming years.

Acknowledgments

We would like to thank Dr. Daniel M. Berry for his constructive feedback and support on using F_β-measures. We thank Sonnhild Namingha from Fraunhofer IESE for proofreading this article.

References

[Ber17] Daniel Berry. Evaluation of tools for hairy requirements and software engineering tasks. In *Proceedings of the IEEE 25th International Requirements Engineering Conference (RE) Workshops*, pages 284–291, 2017.

[CLH+14] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*, pages 767–778, 2014.

[GDA15] Eduard C. Groen, Joerg Doerr, and Sebastian Adam. Towards crowd-based requirements engineering a research preview. In Samuel A. Fricker and Kurt Schneider, editors, *Requirements Engineering: Foundation for Software Quality*, pages 247–253, Cham, 2015. Springer.

[GIG17] Emitza Guzman, Mohamed Ibrahim, and Martin Glinz. A little bird told me: Mining tweets for requirements and software evolution. In *Proceedings of the IEEE 25th International Requirements Engineering Conference (RE)*, pages 11–20, 2017.

[GSA+17] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, et al. The crowd in requirements engineering: The landscape and challenges. *IEEE Software*, 34(2):44–52, March/April 2017.

[GSK+18] Eduard C. Groen, Jacqueline Schowalter, Sylwia Kocpzyńska, Svenja Polst, and Sadaf Alvani. Is there really a need for using NLP to elicit requirements? A benchmarking study to assess scalability of manual analysis. In Klaus Schmid and Paolo Spoletini, editors, *Requirements Engineering – Foundation for Software Quality (REFSQ) Joint Proceedings of the Co-Located Events, CEUR Workshop Proceedings 2075*, 2018.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[HS13] Steffen Hedegaard and Jakob Grue Simonsen. Extracting usability and user experience information from online user reviews. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 2089–2098, 2013.

[KC07] B. A. Kitchenham and S Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE–2007–01, School of Computer Science and Mathematics, Keele University, 2007.

[KGM16] Preet Chandan Kaur, Tushar Ghorpade, and Vanita Mane. Topic extraction and sentiment classification by using latent dirichlet markov allocation and SentiWordNet. In *Proceedings of the International Conference on Advances in Information Communication Technology & Computing (AICTC)*, Article 86, 2016.

[PFMM08] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, 2008.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.