Classifying User Requirements from Online Feedback in Small Dataset Environments using Deep Learning

Rohan Reddy Mekala*§, Asif Irfan*§, Eduard C. Groen†§, Adam Porter* and Mikael Lindvall*
* *Fraunhofer USA Center Mid-Atlantic*, Riverdale, MD, USA {rreddy;airfan;aporter;mikli}@fraunhofer.org
† *Fraunhofer IESE*, Kaiserslautern, Germany & *Utrecht University*, the Netherlands, eduard.groen@iese.fraunhofer.de

***Abstract*—An overwhelming number of users access app repositories like App Store/Google Play and social media platforms like Twitter, where they provide feedback on digital experiences. This vast textual corpus comprising user feedback has the potential to unearth detailed insights regarding the users' opinions on products and services. Various tools have been proposed that employ natural language processing (NLP) and traditional machine learning (ML) based models as an inexpensive mechanism to identify requirements in user feedback. However, they fall short on their classification accuracy over unseen data due to factors like the cost of generating voluminous de-biased labeled datasets and general inefficiency. Recently, Van Vliet et al. [1] achieved state-of-the-art results extracting and classifying requirements from user reviews through traditional crowdsourcing. Based on their reference classification tasks and outcomes, we successfully developed and validated a deep-learning-backed artificial intelligence pipeline to achieve a state-of-the-art averaged classification accuracy of ∼87% on standard tasks for user feedback analysis. This approach, which comprises a BERT-based sequence classifier, proved effective even in extremely low-volume dataset environments. Additionally, our approach drastically reduces the time and costs of evaluation, and improves on the accuracy measures achieved using traditional ML-/NLP-based techniques.**

***Index Terms*—Crowd-based requirements engineering, deep learning, feedback analysis, machine learning, user feedback, user reviews**

I. Introduction

As a growing body of requirements engineering (RE) literature shows, substantial amounts of online user feedback in textual format provide information on user perceptions, encountered problems, suggestions, and demands [2]–[5]. Additionally, user feedback needs to be accounted for from multiple interfaces to cover all customer touch points, including customer reviews, satisfaction surveys, web forms, and social media posts. This information is a valuable source of intelligence and ideas for product improvement, customer outreach, brand development, and competitive analysis. However, the amount of feedback typically obtained for a digital or physical product is too large to be processed manually [6], [7], and established requirements elicitation techniques—such as interviews and focus groups—are not suitable for engaging and involving the large number of users providing feedback. Additionally, feedback comprising user reviews is quite different from fairly well-structured technical requirements created for product development, due the prevalence of inordinate amounts of noise in the form of internet slang, shortcut grammar, bad formatting, etc. [8].

To efficiently extract structured requirement labels in various forms from user feedback data while accounting for the above-mentioned constraints, various classification approaches have been proposed, particularly artificial intelligence (AI) based algorithms that usually employ traditional machine learning (ML) (see Santos et al. [9] for a review). Traditional ML algorithms (e.g., [4], [10]–[13]) need intervention/guidance from domain experts on feature extraction in order to achieve decent classification results towards the corresponding classification task. The classification accuracy of such approaches is adversely affected in a completely automated setup because of the models' inability to generalize understanding of the syntactic and semantic context, thereby leading to poor feature extraction results and overall model accuracy. Other semi-automated classification approaches beside traditional ML include heuristics-based and pattern-matching approaches [9].

Recently, Van Vliet et al. [1] demonstrated that crowdsourced classification can be considered another effective technique for the identification and categorization of user requirements in online feedback. Crowdsourcing is an appealing approach, given its usefulness in harnessing the collective intelligence and skills of a distributed workforce towards RE. Since there are still many contextual aspects of annotation that human beings can do much more efficiently than computers, crowdsourcing has some distinct advantages over ML-backed solutions. Van Vliet et al. [1] proposed *Kyōryoku*, a process in which the complex task of categorizing (parts of) an online review into RE-relevant labels is decomposed into multiple micro tasks to reduce inaccuracies stemming from the variance in domain expertise over a crowd of annotators. They established and followed a 3-step process for the identification of requirements from user feedback, as depicted in the workflow diagram shown in Figure 1 (see Section V-A for details). However, while the crowdsourcing approach yields impressive state-of-the-art accuracy metrics on a range of classification tasks, these techniques outsource the work to a crowd of workers in return for pay, causing them to be limited by their increasing evaluation cost and time as the size and volume of the feedback corpus grows.

This paper proposes a deep learning (DL) backed pipeline

§The first three authors contributed equally to this work.

2332-6441/21/$31.00 ©2021 IEEE
DOI 10.1109/RE51729.2021.00020

for requirements extraction in low-volume training dataset environments. DL-based language modeling architectures such as *BERT* [14], *ELMo* [15], *StarSpace* [16], and *OpenAI GPT* [17] (see Section IV-B) exhibit a deeper understanding of language context and have achieved state-of-the-art results on a wide range of language modeling tasks, including question answering and language inference, classification/labeling, summarization, machine translation, similarity learning, recommendation systems, etc. Our approach eliminates drawbacks faced by traditional ML and crowdsourcing approaches while delivering state-of-the-art results using the gold standard developed by Van Vliet et al. [1].[1] Additionally, using pre-trained representation learning [18] on higher-volume public datasets to capture semantic context in user feedback, we were able to successfully demonstrate the efficacy of the approach even in low-volume dataset environments. Since the pre-trained models holistically capture syntactic and semantic context in user writing styles without needing task-specific training data, subsequent model training steps for task-specific classification have drastically lower compute times with much higher accuracy metrics. As part of the AI pipeline proposed in this paper, we implemented various approaches for requirements extraction from online user feedback using traditional ML as well as DL architectures, and compare the results against the current state-of-the-art benchmark achieved using crowdsourcing. Our best-performing approach built around the BERT architecture achieved state-of-the-art results on the gold standard dataset, with a precision and recall of 92% and 92%, respectively, for classification task P1, and 91% and 91%, respectively, for task P2 (see Section V-A for a description of these tasks).

The remainder of this paper is structured as follows. Section II covers major contributions and takeaways from our research on requirements classification using DL-based pipelines. Section III delves into related research conducted in the field of requirements classification/extraction, and Section IV discusses fundamental aspects of this work. In Section V, we provide a detailed report on the broader AI pipeline proposed, including (a) experiments conducted across the sub-sections of the pipeline and (b) dataset development and preprocessing, researching traditional ML as well as DL implementations along with their architectures and training setup. Section VI summarizes our results from our experiments and post analytics, Section VIII presents the key threats to validity, Section VII discusses the implications of our work, and in Section IX we conclude.

II. Contribution

The paper builds on the *Kyōryoku* crowdsourcing approach of Van Vliet et al. [1], with our approach using the results of their classification tasks P1 and P2 (see Section V-A). By using a DL-backed AI pipeline, we advance the state-of-the-art in user feedback analysis in the following ways:

- This is a novel attempt at developing a DL-based approach for user feedback analysis for RE in low-volume training dataset environments, which was found to be successful in terms of achieving better/comparable classification accuracy than the current benchmark established for classification tasks P1 and P2 [1] using crowdsourcing, and outperforming earlier applications of DL in RE (cf. [9]).
- Our DL-based approach is moreover capable of achieving 91% and 92% summarization, respectively, which Berry [19] denotes as the fraction of (irrelevant) material in the original dataset that is eliminated from the return, diminishing the role of recall in subsequent user feedback analysis in favor of precision.
- Our approach drastically reduces the time and cost of evaluation experienced when using crowdsourced annotations while improving on the accuracy measures achieved using traditional ML / natural language processing (NLP) based techniques.
- This is a novel attempt at determining and comparing the classification accuracy metrics for the above-mentioned classification tasks for user feedback analysis using a near-exhaustive range of evaluation techniques including: 1) traditional crowdsourcing methods, 2) traditional ML models, and 3) DL models.
- This paper goes one step further towards graphically showcasing the potential of producing even better accuracy metrics given an incremental increase in training data beyond the currently used dataset volume.

III. Related Work

The subdiscipline of RE that deals with obtaining user feedback from large groups of current and potential users (i.e., a "crowd") is typically referred to as Crowd-based RE, or CrowdRE in short [20], [21]. CrowdRE encompasses such aspects as motivating the crowd to provide (more) user feedback, eliciting requirements from user feedback such as text-based and usage data, and validating the identified requirements with the crowd. Especially text mining approaches have received much attention (see Khan et al. [5] for reviews). Among these, NLP techniques backed by traditional ML models have been proposed most often for solving classification problems involving user feedback analysis for RE (see Santos et al. [9], [22] and Wang et al. [23] for a review). The main points of contention using this broad class of techniques that make use of traditional ML are: 1) They require manual intervention in the form of an expert (often a linguist or domain expert) for feature selection; 2) they yield sub-par accuracy metrics in general, making them infeasible for use in fully automated production environments; and 3) they insufficiently capture semantics and context in the dataset, leading to inordinate sensitivity/variance in output labels with respect to minor perturbations in the input data.

Although early work in CrowdRE foresaw a synergy between CrowdRE and traditional crowdsourcing [24], few works have so far explored the involvement of traditional crowdsourcing. To

[1]The crowd-annotated results of Van Vliet et al. [1] are publicly available through Zenodo at https://doi.org/10.5281/zenodo.3626185. The data and code of our work are publicly available through Figshare at https://doi.org/10.6084/m9.figshare.14273594.

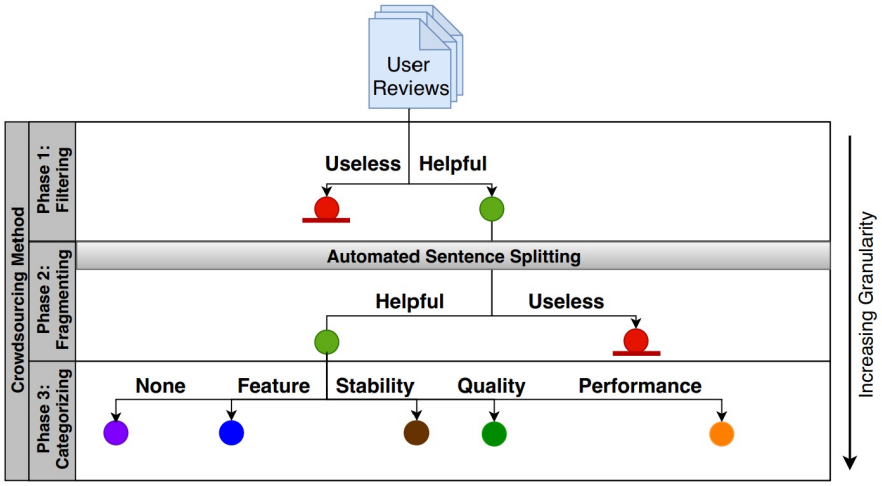

Fig. 1. Overview of the crowdsourced annotation method *Kyōryoku*, adapted from Van Vliet et al. [1]

our knowledge, only two recent works outsourced user feedback classification to a crowd in this domain. Van Vliet et al. [1] investigated the ability of crowd workers to accurately classify user requirements in online reviews. Their *Kyōryoku* approach divides the complex task of annotating online reviews into a series of consecutive micro tasks earmarked for crowdsourced annotation, with a subset of a micro task being classified at greater granularity in a subsequent micro task (see Section IV-C). The results obtained through the crowdsourcing platform *figure eight*[2] yielded good precision (P) metrics between 75–93% and recall (R) metrics between 74–84%. Stanik et al. [25] crowdsourced the annotation of 10,000 English and 15,000 Italian tweets, which were classified into problem report, request, inquiry, or irrelevant. For each language, they set up a micro task in *figure eight*. The outcomes were used as training data, as discussed below. Although they have merit, the major hindrances faced by these traditional crowdsourcing-based approaches include the costs (comprising platform fees and a fixed price per annotation, multiplied by the number of items and the number of annotators per item) and the evaluation time constraint stemming from a slow ramp-up and a long tail to reach 100% of the annotations.

Further notable contributions on employing crowdsourcing techniques in RE include the *Organizer & Promoter of Collaborative Ideas* (*OPCI*; [26]), *StakeSource* [27], *Requirements Bazaar* [28], and *REfine* [29], which all mobilize a crowd to collaboratively elaborate requirements, e.g., by proposing, discussing, analyzing, prioritizing, and/or validating them. Specifically for annotating user feedback, the *Crowd-Annotated Feedback Technique* (*CRAFT*; [30]) provides a three-step approach for crowd members to assign a classification based on predefined user requirement dimensions to a selected text snippet. With the exception of StakeSource, these approaches typically rely on the intrinsic motivation of crowd members to participate.

Researchers have also proposed some DL-based techniques to classify requirements in technical documents, predominantly using the PROMISE dataset [31]. Navarro-Almanza et al. [32] used Convolutional Neural Networks (CNNs) to predict functional and non-functional requirements (FRs and NFRs) using data from the PROMISE corpus for training, with good results ($P = 81\%, R = 79\%$). Tamai and Anzai [33] successfully used CNNs, again as a multi-layered perceptron, to predict quality requirements in software requirements specifications ($\bar{P} = 76\%, \bar{R} = 77\%$). Hey et al. [34] proposed NoRBERT, which fine-tunes a BERT-based classifier to solve binary and multi-class classification problems in RE by employing transfer learning to account for poor generalization problems in unseen projects. They classified requirements from the PROMISE dataset into FRs and NFRs, as well as subclasses of FRs and NFRs. Although they used a well-structured dataset, the training dataset comprised only 625 requirements. For binary classification tasks, they reported P metrics between 88–93% and R metrics between 86–95%. The results attained validate the capabilities of using BERT-based language models with transfer learning techniques to produce state-of-the-art results even with low-volume datasets.

Another recent work compared several traditional ML approaches and the DL-based classification techniques Bidirectional Long Short-Term Memory (BiLSTM) and BERT on a multilabel classification of obligation sentences in large software engineering contracts [35]. Their models were trained on a large set of 20 contracts from different domains, each spanning 100–500 pages for a total of 18,614 sentences, for which the classification into 14 contract requirements types was performed manually, and tested on a 250-page contract with 1,608 sentences. They found that BERT outperformed other classifiers for almost every classification. The main difference

[2]https://www.figure-eight.com/, later acquired by Appen.

between all the aforementioned studies and our proposed approach is that they used well-structured and well-organized technical product requirement datasets, while we deal with noisy online feedback text, which can be challenging to model. Although our work has most similarities with NoRBERT, the most notable difference is NoRBERT classifies existing technical requirements that have previously been validated, while our analysis takes inherently ambiguous online user feedback as its input and distinguishes content that is irrelevant to RE from relevant content.

In several works, DL has been applied in requirements classification research to analyze online user feedback for RE. Zhou et al. [36] used a combination (i.e., an ensemble) of Multinominal Naïve Bayes and Bayesian Network classifiers that predicted whether or not a user review contains a bug report, and they achieved good results. Guzmán et al. [37] analyzed user feedback across eight dimensions using Naïve Bayes, Support Vector Machines, Logistic Regression, Neural Network classifiers, and ensembles of these. They attained mediocre results overall, with precision values between 69% and 75%, respectively, for Ensemble B and Logistic Regression, and recall values between 42% and 63% for Logistic Regression and Ensemble B, respectively. The ensembles performed best, particularly with respect to average recall. Of the four individual classifiers, the DL-based Neural Network classifier was the slowest, but performed significantly better statistically ($\bar{P} = 74\%, \bar{R} = 59\%; \bar{F}_1 = 62\%$). Its best results were attained on classifying bug reports ($P = 83\%, R = 75\%; F_1 = 79\%$), and its poorest on classifying complaints ($P = 45\%, R = 8\%; F_1 = 14\%$).

Stanik et al. [25] compared the performance of traditional ML algorithms to a CNN-based DL model with a pre-trained FastText embedding layer to gauge the effectiveness of these models in classifying user feedback. They used a corpus of 6,406 app reviews and 10,364 tweets for training the classifier, predicting requirements as belonging to one of the categories Problem Report, Inquiry, or Irrelevant. Their reported average precision and recall results for the best-performing models were average, with 65% and 78%, respectively, using traditional ML, and 60% and 64%, respectively, using DL techniques. Hence, works that have applied DL for user feedback classification for RE have not demonstrated the true potential of DL yet.

IV. Background

Our work builds on several fundamental aspects that we discuss in this section. We first provide an overview of selected existing work on traditional ML approaches and DL approaches in Section IV-A, respectively Section IV-B, after which Section IV-C discusses prior research that produced our training data.

A. Traditional Machine Learning Implementations

Although our work focuses on DL pipelines, we used various traditional ML techniques to set a reference benchmark to test our DL pipelines against. In this section, we showcase the results from the two techniques from among the different variants of traditional ML approaches used that performed best in our experiments, namely, 1) a TF-IDF-based classifier and 2) a Naïve Bayes classifier.

TF-IDF-based Classifier: Term frequency – inverse document frequency (TF-IDF; [38]) is a statistical information retrieval technique used in research to generate vectorized meanings of words within reviews. This technique converts a document into a vectorized representation in an n-dimensional space, where n is the size of the vocabulary. It assigns a weight to each word in the review based on its term frequency (TF), assigns a weight to each word based on its rarity across the entire review corpus (IDF), and finally multiplies the two to obtain a TF-IDF score vector on both the word/token and review level. These review document vectors are then used as input to train over said classification objectives using Support Vector Machine (SVM) [39] based binary classifiers.

Naïve Bayes Classifier: The Bayesian Network aided Naïve Bayes classifier [40] for text categorization belongs to a broad family of classifiers based on the Bayes probability theorem, and is implemented in our research across two phases. The first phase comprises computing a priori probabilities for each word and each class in the dataset. For a word, this is the number of times it occurs in a particular review class (for example, "Useful review") compared to the total number of reviews belonging to that class in the dataset; for a class, it is the number of reviews in that class compared to the total reviews in the dataset. In the second step, given the database of probabilities for terms appearing across the classes, and assuming the words in every review are conditionally independent (according to the naïve assumption), Bayesian decision rules are used to compute the class-conditional probabilities for each review.

B. Deep Learning Implementations

With major advancements in DL driven by the availability of voluminous datasets and compute capacity, DL architectures have achieved state-of-the-art performance metrics on a wide range of NLP tasks. These models can be implemented in an independent end-to-end pipeline and do not rely on manual feature engineering relative to the task in question. Additionally, the DL approach exhaustively addresses the drawbacks faced by traditional ML models, like the inability to capture semantics or context from word interrelationships, the inability to deal with out-of-vocabulary words, etc.

In this section, we provide a general introduction to the three DL-based model architectures that performed best in our experiments: FastText, ELMo, and BERT. All three models output *deep-contextualized word embeddings or representations* that model the syntactic/semantic context and variance across linguistic contexts (polysemy). These word embeddings are vectorized mathematical representations of words or sentences in much lower dimensions and exhibit interesting properties, like vectors for words/phrases representing synonyms that are mathematically close to each other (using the L2-distance; the Euclidean or "straight-line" distance between two points or vectors). The first step of the implementation involves training word embeddings from reviews; a process abstracting

both language modeling and feature extraction techniques. These word embeddings are then used as intermediary input to train/fine-tune classifiers.

FastText-based Classifier: FastText [41] is a word representation model developed by Facebook AI Research based on the skip-gram model [42], where each word is represented as a bag of character n-grams. Each character n-gram in this model has a vector representation associated with it and the words are represented as the sum of the character n-grams. This also helps to efficiently compute word representations for words that did not appear in the training data (out-of-vocabulary words). In skip-gram, the model tries to predict the words representing the surrounding context conditioned on the current word. Once a threshold accuracy is reached, the output layer is discarded and the hidden layer weights are used as word embeddings for subsequent classification models. FastText provides several pre-trained models, trained on different datasets.

ELMo-based Classifier: Embeddings from Language Models (ELMo [15]) is a deep contextualized word representation model developed by the Allen Institute for Artificial Intelligence. ELMo encodes complex contextual characteristics surrounding words, sentences, and reviews using character embeddings. This character-level contextual dependence allows ELMo to form accurate representations of out-of-vocabulary words. The ELMo model architecture used in our research comprises two stacked layers of deep bi-directional long short-term memory (LSTM) models to produce pre-trained word embeddings. Each of these layers produces intermediate word embeddings that are passed to the next layer. The final word representation, which is a 512-dimensional vector, is a weighted sum of the initial word vector and the intermediate word vectors from the two layers.

BERT-based Classifier: The Bidirectional Encoder Representations from Transformers (BERT; [14]) architecture is a state-of-the-art DL-based pipeline based on Transformers [43], which has achieved state-of-the-art accuracy metrics on all major NLP tasks including question answering (SQuAD v1.1) [44], Natural Language Inference (MNLI) [45], and others. BERT uses a neural network architecture called Transformers, based on the attention mechanism [46]. The Transformer architecture comprises an encoder that generates embeddings, and a decoder that produces a prediction for the corresponding classification task. BERT just uses the encoder section of the Transformer while employing a bi-directional training approach. This means that the model learns by processing the whole sentence left to right and right to left simultaneously. The BERT model would first be used to develop pre-trained embeddings on a public dataset corpus. However, there are different versions of pre-trained BERT models for a range of publicly available data domains.

C. Crowdsourced Data Annotation

Our DL-based approach was performed over previous work on CrowdRE by one of this paper's authors, particularly the recent study of Van Vliet et al. [1]. From an earlier user feedback dataset taken from the work of Groen et al. [47], Van Vliet et al. created a custom sample by randomly selecting 1,000 online user reviews from two app stores pertaining to eight apps. The sampling was bootstrapped to have an equally distributed variation of good and bad reviews. Groen et al.'s dataset consists of 132,194 English-language user reviews for 36 apps, curated by selecting a free and a paid app in each of six app categories across three app stores. Each app had to be available in all three app stores, and each of the stars of the 5-star rating scale made up for at least 5% of the user reviews for each app. The dataset was originally used to identify software product quality characteristics [47].

Van Vliet et al. [1] crowdsourced the annotation work of their custom sample through the crowdsourcing platform *figure eight*. To make the crowdsourced micro task more comprehensible to crowd workers who have no prior knowledge of RE, they created simplified classification categories of software product qualities. These reviews were then crowd-annotated sequentially in three phases for classification: P1, P2, and P3 (see Figure 1), each of which was a separate micro task. To measure the precision and recall of the crowd workers, they also created a gold standard according to these classification categories that served as the ground truth. The model development reported in the paper utilized their sample and their gold standard. The phases were as follows:

P1: Filter out useless reviews. In this first phase, the crowd workers were presented with individual reviews and asked to label them as either *helpful* to RE (based on whether the review mentions pertinent aspects of the app like features, bug reports, performance issues, etc. that would help developers improve the app) or *useless* to RE (if the review does not relate to the app or its functions, contains user feelings, jokes, etc.). The result of this phase is an assessment of the relevance of user reviews from an RE perspective. Because the focus is on filtering out irrelevant reviews, the reviews in the useless category are considered the "positives" in the analysis. By eliminating these positives, summarization is achieved at the review level [19].

P2: Filter out useless sentences. In the second phase, all the useful reviews obtained from P1 (i.e., the "negatives") were split into individual sentences and the same process of marking useful and useless content from an RE perspective was presented to the crowd workers, in this phase sentence by sentence. The result of this phase is a more fine-grained assessment of the user reviews. Irrelevant sentences from relevant user reviews are discarded, achieving summarization at the sentence level.

P3: Classify useful sentences. In the third phase, all the useful sentences obtained from P2 were labeled by the crowd workers into one of the predefined requirement categories: Feature Request, Stability Feedback, Performance Feedback, Quality Feedback, or Other/None of the Above. The result of this phase is a crowdsourced classification of sentences relevant to RE.

For each of these phases, the participants were presented with detailed guidelines and representative examples to ensure optimal accuracy in the labeling process. Additionally, the participants were required to partake in an eligibility test, which served as an additional layer of validation.

V. Method and Design

We developed an AI-backed pipeline to allow for holistic development and validation of various state-of-the-art ML- and NLP-based model architectures, of which we have made the data and code publicly available.[3] The objective of this process was to identify an AI pipeline flow that matches or improves the benchmark accuracy metrics achieved using crowdsourcing for classification tasks P1 and P2, thereby summarizing [19] a corpus of online user feedback by retaining only the contents that are relevant to RE. Figure 2 presents a high-level overview of the proposed development pipeline, with custom modules for data labeling and pre-processing, model research and development (R&D), testing and benchmarks, and post analytics. In the following subsections, we will describe these modules in more detail.

A. Data Development, Labeling, and Pre-Processing

The data labeling and pre-processing module deals with the acquisition and labeling of training and testing datasets, and with performing the requisite cleaning/pre-processing tasks to make our datasets ready for consumption by the ML models. As part of the work done for this paper, we developed an AI-backed proof of concept using the sample of user reviews and the gold standard of Van Vliet et al. [1] described in Section IV-C to solve classification problems pertaining to tasks P1 (1,000 reviews) and P2 (1,242 sentences). While these numbers can both be considered small, we nevertheless were able to achieve good precision and recall measures by using a measured combination of transfer learning and advanced DL-based pipelines for the generalization of syntactic and semantic understanding. Since P3 is a multi-label (5 categories) classification task, we posit that it requires a much higher volume of training data, and we intend to account for this in a future iteration of our research.

We furthermore acknowledge that a per-sentence analysis of user reviews, as done in P2 and P3, entails a relatively arbitrary subdivision, but one that can be achieved easily through sentence splitters and other simpler NLP heuristics. A known trade-off is that a sentence might contain multiple relevant (shorter) snippets, while a sentence that relies on information from another sentence might not be understood when assessed individually. Prototypes exist that support manual annotation of text snippets, varying from parts of a word to an entire review (e.g., [30]), but AI-based means of portioning user feedback into snippets of appropriate sizes for the purpose of RE currently do not exist. In fact, we posit that DL approaches such as very deep convolutional networks (so-called deep nets) can provide the means to identify such text snippets by revealing the weighing of how individual words contributed to its classification decision. Until this technology can be applied to user feedback, the classification of individual sentences as performed in P2 helps provide intermediary insights into the performance of DL classifiers at greater granularity, beyond predicting the existence of contents relevant to RE in longer review texts.

Each review for task P1 and each sentence for task P2 was pre-processed as follows: (i) tokenize each review/sentence into individual words and add special characters '[CLS]' at the start and '[SEP]' at the end; (ii) map each word to a unique numeric ID, and (iii) pad each review/sentence to the maximum length of a review/sentence using a special '[PAD]' token, and then consequently add attention masks [46] for each of them.

B. Model Research & Implementation

The module for model R&D (see Figure 2) involves the development, optimization, testing, and benchmarking of various ML-/DL-based model architectures. To develop ML-based models, we used the *scikit-learn* [48] and *Keras* [49] frameworks because they enable fast experimentation with commonly used networks. For DL models, we used the *pytorch* framework [50] along with the libraries provided by developers for their individual models, which allowed us to fine-tune them on our dataset. We trained the models for 15–25 epochs, with a batch size of 16 and a learning rate between $2e^{-5}$ and $2e^{-4}$. We also used *scikit-learn* to obtain evaluation metrics for each classification task. All classifiers were trained on a machine with 32 GB RAM, a 12-core 3.50 GHz processor, and an NVidia GeForce RTX 2080 Ti graphics card with 12 GB VRAM.

1) Target Dataset: Following the pre-processing, to create the target dataset, we randomized the labeled datasets of the two tasks, P1 and P2, and split them in a ratio of 95:5 to produce the training and test datasets. The training dataset was used to fit the ML models and the test dataset to evaluate the fit of the trained ML models over various predefined performance metrics. For larger datasets, a greater train to test ratios is likely better, but this ratio helped us ensure maximal dataset size for training/learning, which achieved a slightly better result than a 90:10 split. A larger dataset could also facilitate experiments with various cross-validation techniques to extract generalizable characteristics from our trained models.

2) Target ML Models: We applied traditional ML approaches in our experiments to solve classification tasks P1 and P2 in order to set a reference benchmark to test our DL pipelines against. Of these, a TF-IDF vector transformation paired with an SVM classifier and a Bag-of-Words vector representation with a Naïve Bayes classifier performed best (see Section IV-A for a description of these algorithms). In both approaches, we used statistical techniques regarding the occurrences of words to make inferences about the classification category to which a specific review would correspond. Unlike DL models, these traditional ML models cannot explicitly incorporate the semantic context of the feedback since they work on the assumption that similar documents would be comprised of

[3]The data and code of our work are publicly available through Figshare at https://doi.org/10.6084/m9.figshare.14273594.

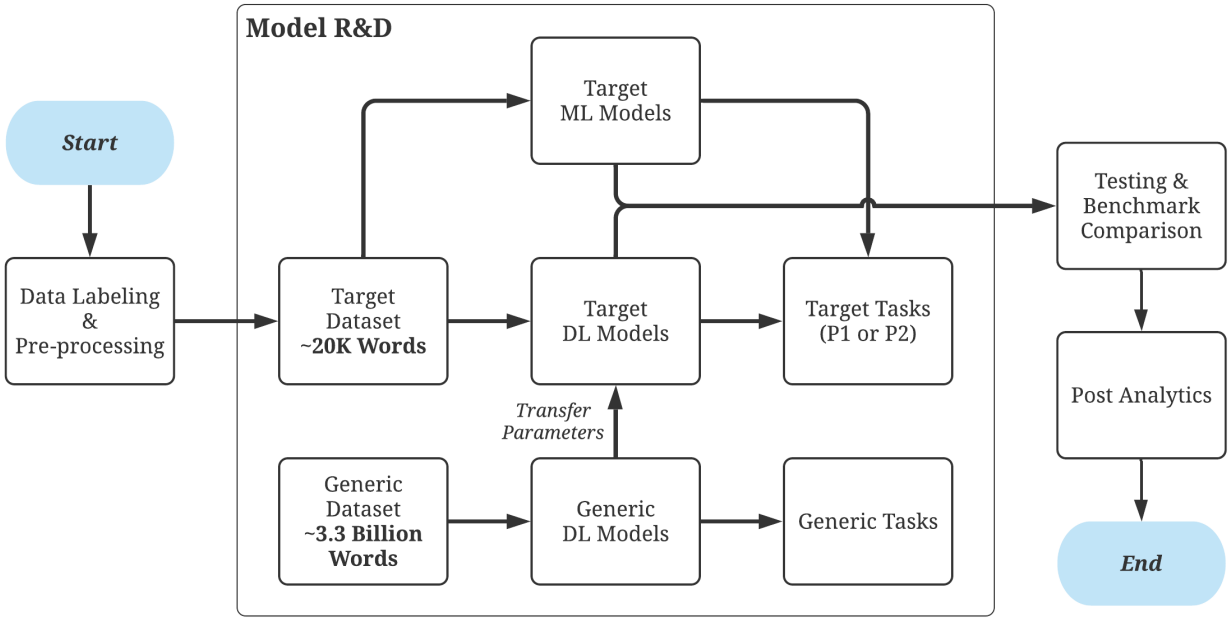

Fig. 2. Overview of the development pipeline used

nearly identical word and word–interrelationship distributions. They also ignore the natural order in which words appear within a sentence, thus losing information pertaining to syntactic structure. However, these models are easy to implement and provide a good baseline for judging the performance of the more advanced DL pipelines.

3) Generic Dataset, DL Models, and Tasks: To perform well, DL models require massive amounts of data to generalize and learn robust representations. As a result, if we were to develop DL-based word representations by directly training models on our low-volume gold standard from Van Vliet et al. [1], this would impose a major drawback. Besides, it is not always possible for these deep-layered models to be trained from scratch because it would engender significant costs in terms of compute capacity and time. We overcame these constraints by taking a *transfer learning based approach*. For our best performing models developed using DL, the model R&D includes an implementation of syntactic and semantic context transfer using models of word representations pre-trained on voluminous public datasets (for example, ∼3.3B words collected from the BookCorpus [51] and the English-language Wikipedia, or other text corpora derived from social media, news websites, etc.) over generic text-based tasks such as question answering, text classification, part-of-speech tagging, etc. This enables near-exhaustive learning of context in natural language text and serves as a useful starting point for subsequent requirements classification task specific training, even in low dataset volume environments. Additionally, the first few layers, comprising the embedding-generating encoder of a language modeling network, have been found to learn the basic contextual features, which can be transferred to most, if not all NLP tasks [52], [53]. Hence, using embeddings trained on a huge corpus in a similar domain as a starting point is a highly effective practice, as our results confirm.

4) Target Dataset, DL Models, and Tasks: After the models were trained, we used the word embeddings resulting from the transfer learning as intermediary input to train and fine-tune classifiers for tasks P1 and P2. The three architecture pipelines we implemented—BERT, ELMo, and FastText (see Section IV-B for a general introduction)—are as follows:

For the **FastText**-based classifier, we used a model pre-trained on 6,00B words from the Common Crawl dataset[4], which consists of raw webpage data. To represent each word, this model uses a 300-dimensional vector. Consequently, for classification, a perceptron layer with softmax activation was added to the model. We fine-tuned the final model by using the weights of the pre-trained model as our initial weights, before re-training it separately for classification tasks P1 and P2. The final language model comprised 300 hidden units, followed by a linear classifier.

For the **ELMo**-based classifier, we used a model pre-trained on the *One Billion Word Benchmark* [54] dataset, which consists of approximately 800M words. Consequently, a sequence of layers for classification, comprising a 1-D convolutional layer followed by a perceptron layer with softmax activation, was added to the model. We fine-tuned the final model by using the weights of the pre-trained model as our initial weights,

[4]https://commoncrawl.org/

TABLE I
CLASSIFICATION RESULTS FOR TASKS P1 AND P2, COMPARING VARIOUS ML-/DL-BASED MODELS WITH A CROWDSOURCED MICRO TASK

	P1: Useless reviews			**P1: Useful reviews**			**P2: Useless reviews**			**P2: Useful reviews**		
Method	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Crowdsourcing	0.93	0.84	0.88	0.83	0.93	0.88	0.88	0.81	0.84	0.83	0.89	0.85
SVM	0.90	0.79	0.84	0.83	0.92	0.87	0.73	0.92	0.82	0.83	0.56	0.67
Naïve Bayes	0.83	0.79	0.81	0.81	0.85	0.83	0.75	0.82	0.79	0.63	0.52	0.57
FastText	0.75	0.60	0.67	0.84	0.91	0.87	0.68	**1.00**	0.81	**1.00**	0.25	0.40
ELMo	0.83	0.80	0.82	0.81	0.84	0.82	0.78	0.78	0.78	0.68	0.68	0.68
BERT	**0.95**	**0.88**	**0.92**	**0.88**	**0.96**	**0.92**	**0.93**	0.93	**0.93**	0.91	**0.91**	**0.91**

before re-training it separately for classification tasks P1 and P2. The final language model comprised the input embedding transformed using 2,048 convolutional filters, LSTM layers containing 4,096 units, and 1,024 1-D convolutional filters followed by a linear classifier.

For the **BERT**-based classifier, we used the *bert-base-uncased* pre-trained model, which was developed by training the model on a corpora comprising ∼3.3B words collected from the BookCorpus [51] and English Wikipedia datasets available for public use. To represent each word, the model used a 768-dimensional embedding vector. Consequently, for classification, a perceptron layer with softmax activation was added to the model. We fine-tuned the final model by using the weights of the pre-trained model as our initial weights, before re-training it separately for classification tasks P1 and P2. For the final model, most hyper-parameters remained the same as those used for developing the pre-trained embeddings. The final model developed consisted of 12 Transformer blocks [43], 768 hidden layers, 12 attention heads, and a total of 110M parameters.

5) Testing & Benchmark Comparison and Post Analytics: The trained classification models were then passed through the testing & benchmark comparison and post analytics modules to validate the model results on unseen test data and generate detailed insights on the model performance metrics.

VI. RESULTS

Table I details the performance of all five classifiers for each class for tasks P1 and P2 compared to the gold standard of Van Vliet et al. [1]. The performance scores were computed using unseen reviews in the test dataset, which was not used in the training phase. In addition to the cumulative comparison of the results attained with the traditional ML and DL model types used in our research, we included the crowdsourcing results of Van Vliet et al. [1], which were judged based on the same gold standard. This provided a benchmark that can help determine how well automated processing approaches hold up against the classification task performed manually. In addition to the precision (P) and recall (R) values, we report the F_1-measure, which is the harmonic mean of P and R. We chose not to adjust for the relative importance of P and R by determining a different F_β [19], [55], as we lacked accurate manual processing time measures and because the high occurrence of irrelevant fragments ($\sim 50\%$) would have likely resulted in β being approximately 2, favoring R values only slightly more than a β of 1.

The **TF-IDF**-based classifier performed reasonably well for both tasks P1 and P2, achieving weighted average P and R measures of 86% and 86%, respectively, for task P1, and 78% and 76%, respectively, for task P2.

The **Naïve Bayes** classifier performed slightly worse than the TF-IDF-based classifier; it reached weighted average P and R measures of 82% and 82%, respectively, for task P1, and achieved only 71% and 71%, respectively, for task P2. The difference in performance between the two tasks can be attributed to inherent inefficiencies of these techniques in capturing semantics and context, leading to lower accuracy metrics on task P2. The lower number of words available per input unit for P2 means that less intrinsic context information can be captured. The input datapoint for P1 comprises the entire review, while P2 operates on a single sentence, which caused causing a significant drop in accuracy for P2.

The **FastText**-based classifier attained scores that were similar to those of the ML-based classifiers, achieving weighted average P and R measures of 81% and 82%, respectively, for task P1, and 80% and 71%, respectively, for task P2. Interestingly, on P2, it outperformed other classifiers on two occasions. It achieved 100% recall classifying useless reviews, but at the expense of reaching only 68% precision. On classifying useful reviews, we observed the reverse situation; it reached merely 25% recall, but was 100% precise on those it did classify as useful.

The **ELMo**-based classifier showed reasonable results, achieving weighted average P and R measures of 82% and 82%, respectively, for task P1, and 74% and 74%, respectively, for task P2. The disparity in metrics between the two classification task types indicates a similar problem as faced when using traditional ML models with low-volume datasets.

The **BERT**-based classifier performed best overall, achieving good results, with weighted average P and R measures of 92% and 92%, respectively, for task P1, and 91% and 91%, respectively, for task P2.

Table II presents the processing times required by each model to classify a single review and sentence and the total training time prior to processing for tasks P1 and P2, respectively. All classifiers can be considered highly time-efficient in performing

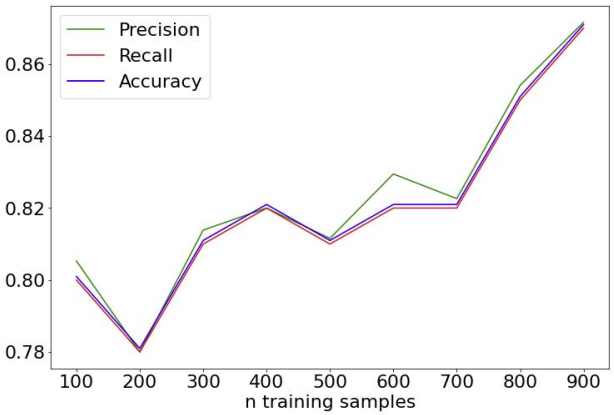

Fig. 3. Evaluation metrics for the BERT implementation trained with varying dataset sizes on task P1

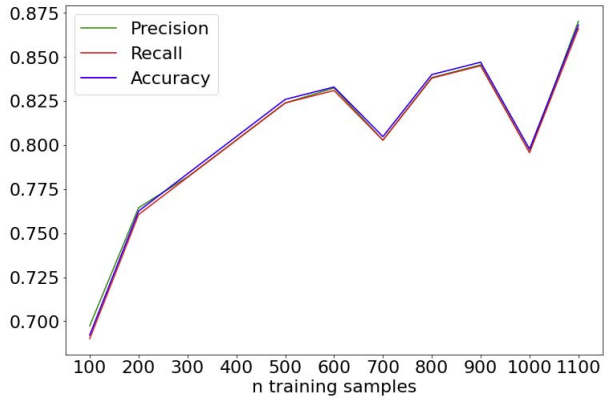

Fig. 4. Evaluation metrics for the BERT implementation trained with varying dataset sizes on task P2

classifications, requiring 8 milliseconds or less per inference. As expected, the two traditional ML classifiers were much faster than the DL-based classifiers, while the ELMo and BERT architectures took far longer because they required up to 6:18 minutes (for BERT on P1) to build their models.

The results of our experiments show that using pre-trained embeddings from larger corpora before training our core classification models is a very effective way of achieving state-of-the-art accuracy metrics in low-volume dataset environments. In our post analytics research, we delved deeper into understanding the effect of dataset size on the accuracy of our best-performing DL pipeline based on BERT. As part of our experiments, we fine-tuned our best performing BERT implementation over dataset sizes ranging from 100 to 900 training samples for task P1 and between 100 and 1,100 training samples for task P2, with increments of 100 on both. As illustrated in Figures 3 and 4, we observed progressively increasing values of precision, recall, and accuracy with a corresponding increase in training data for both classification tasks. The results are highly encouraging and clearly show scope for getting even better results in terms of discarding user feedback irrelevant to RE than those achieved by the current state-of-the-art if we increase the size of the training dataset.

TABLE II
PROCESSING AND TRAINING TIME PER CLASSIFIER FOR TASKS P1 AND P2

Classifier	Processing Time per Inference (in seconds)		Training Time (in seconds)	
	P1	P2	P1	P2
SVM	.002	.007	1.42	1.75
Naïve Bayes	.0002	.0001	0.04	0.05
FastText	.0002	.00003	56.10	50.25
ELMo	.008	.006	331.49	219.08
BERT	.005	.003	377.95	284.98

VII. DISCUSSION

As part of the research conducted in this paper, we successfully developed an AI-backed pipeline using the low-volume gold standard dataset of Van Vliet et al. through their state-of-the-art crowdsourced annotation approach *Kyōryoku* [1]. We trained classifiers with the results from two requirements classification tasks: Task P1 discerns useless user reviews from useful ones, and P2 does so at a per-sentence level. Among the various traditional ML-based and DL-based models developed and tested as part of the pipeline, the pre-trained representation-backed BERT implementation achieved the best accuracy metrics on both tasks, with comparable or better results than the current state-of-the-art achieved using traditional crowdsourcing. The results clearly showed that the BERT- and crowdsourcing-based approaches stand out on the accuracy metrics for both tasks, with BERT even outperforming the manually performed task while being significantly faster and inducing virtually no evaluation costs. BERT took only a fraction of the time it took for each crowdsourced micro task to classify a single review for task P1 (0.005 s vs. 13.15 s, or approx. 1:2,630) or sentence for task P2 (0.003 s vs. 11.15 s, or approx. 1:3,717). Van Vliet et al. [1] spent $130.80 and $106.32 to crowdsource the annotation tasks P1 and P2, respectively.

Overall, the BERT-based classifier proved effective in low-volume dataset environments. Additionally, our post analytics tests to study the influence of training dataset size on the accuracy metrics show clear opportunities to achieve even better accuracy results by increasing the training dataset size. These outcomes have great implications for NLP approaches applied in RE. So far, few works have convincingly demonstrated the contribution DL-based approaches can potentially make to classifying user feedback for RE. Akin to the recently published NoRBERT approach [34], which successfully employed the BERT architecture to classify requirements even with limited training data, our experience with using BERT to classify user feedback shows great promise, and we expect to reach even greater accuracy as the amount of training data increases, as seen in Figure 3.

Moreover, the ability to reliably distinguish between what is relevant or irrelevant to RE, can contribute to an important shift in the focus of automated classification tools. Berry [19] has coined "summarization" (S) as a measure for the fraction of irrelevant material removed from a document (e.g., a dataset), which is what the classification tasks P1 and P2 essentially do, thereby increasing S. With higher S, focus shifts away from optimizing a tool's R in favor of greater P. In our results, the BERT-based classifier achieved a summarization of 92%. For user feedback analysis in RE in general, and CrowdRE in particular, this opens up new avenues. On the one hand, by incorporating the classification presented in this paper as a pre-processing step, the emphasis of classification tools and related research may shift from optimizing recall towards optimizing precision. On the other hand, this work may spawn further research into employing DL-based classification of user feedback, because we might now have reached the point where DL outperforms traditional ML classifiers, and—within the scope of this study—even manual classification.

By validating the usefulness of the above-mentioned techniques over solving various requirements classification problems, our research also serves as an important step towards the broader adoption of DL-based approaches in commercial applications concerning user feedback analysis. The high precision and recall achieved, mixed with the benefits of very low compute times and costs during evaluation and no need to rely on voluminous labeled data for the training sets, should drive further research needed to make DL-based approaches a mainstay in corresponding production-level software.

VIII. Threats to Validity

In terms of construct validity, we tried many measures to create the best model. Limitations introduced by *Kyōryoku* [1] also affect this work because we used the data resulting from their experiment. This particularly pertains to incorrect classifications in P1 that were perpetuated in P2. Moreover, only three judgments were made per item in P1 and P2, respectively; more judgments could have raised the quality. We assert that the dataset's age does not have any influence on judging what content is relevant or irrelevant to RE.

Internal validity is threatened by the small sample size, but the main finding of this work on online feedback classification is that high precision and recall can be achieved on unseen test data even with low-volume training datasets. Earlier, Hey et al. [34] found that a BERT-based classifier can be trained to accurately classify requirements in a low-volume data environment, which our results corroborate for unstructured user feedback. By employing weights/embeddings from models trained over large publicly available corpora for use in our requirements classification models, we circumvented the need for directly developing large-volume datasets for our classification tasks while also achieving encouraging generalizations regarding the syntactic and semantic understanding of our datasets. Our results are expected to be further improved when more training data is available, as Figures 3 and 4 suggest.

As for external validity, we believe the findings from this work can be reused well on other app store reviews and perhaps also other user feedback such as tweets, but our approach might not identify useless content so well in other RE documents.

IX. Conclusion & Future Work

In this work, we have shown how the DL-based architecture BERT can reliably determine what portions of user feedback are irrelevant from an RE perspective. It did so with high accuracy, even with a small training dataset. These outcomes are promising for user feedback classification in CrowdRE, for which traditional ML approaches have usually been employed [9]. We demonstrated that, when applied to distinguish useful from useless contents for RE, DL-based pipelines are now capable of outperforming traditional ML approaches in classifying inherently ambiguous user feedback.

As an extension to the current proof of concept developed, we intend to further develop the gold standard dataset by adding more reviews and corresponding annotations. Consequently, with enough training data points, we intend to push the state-of-the-art and achieve accuracy metrics of between 95–100% for all three classification tasks P1, P2, and P3, where P3 entails classifying the remaining user feedback (not classified as irrelevant in P1 and P2) through multi-label classifications into the RE dimensions of the function and quality of a piece of software. During the course of this work, we also encountered a limitation in the work of Van Vliet et al. [1] imposed by their focus on the manual annotation of user feedback by a crowd of workers, namely that P2 and P3 rely on a subdivision of user reviews into individual sentences. In addition to being arbitrary, it is contradictory and quite likely detrimental to the ability of bidirectional representations in the DL models to take the context of a word into account. Therefore, as an alternative to P2, we propose a future research direction into DL as a means of assessing on a per-word basis whether it is relevant to RE, so that positive word weightings allow the creation of text snippets of appropriate sizes, where a classification can rely on anything from an individual word to an entire review. In turn, as an alternative to P3, this can form the basis for creating gold standards according to text snippets that provide the DL models with more efficient training data to correctly classify online user feedback into requirements categories.

Acknowledgment

We thank Sonnhild Namingha for proofreading this paper.

References

[1] van Vliet, M., Groen, E.C., Dalpiaz, F., Brinkkemper, S.: Identifying and classifying user requirements in online feedback via crowdsourcing. In: Proc. of REFSQ, pp. 143–159 (2020)
[2] Chen, N., Lin, J., Hoi, S.C., Xiao, X., Zhang, B.: AR-miner: Mining informative reviews for developers from mobile app marketplace. In: Proc. of ICSE, pp. 767–778 (2014)
[3] Lu, M., Liang, P.: Automatic classification of non-functional requirements from augmented app user reviews. In: Proc. of EASE, pp. 344–353 (2017)
[4] Maalej, W., Nabil, H.: Bug report, feature request, or simply praise? On automatically classifying app reviews. In: Proc. of RE, pp. 116–125 (2015)

[5] Khan, J.A., Liu, L., Wen, L., Ali, R.: Crowd intelligence in requirements engineering: Current status and future directions. In: Proc. of REFSQ, pp. 245–261 (2019)

[6] Groen, E.C., Schowalter, J., Kopczyńska, S., Polst, S., Alvani, S.: Is there really a need for using NLP to elicit requirements? A benchmarking study to assess scalability of manual analysis. In: Proc. of NLP4RE (2018)

[7] Hosseini, M., Phalp, K.T., Taylor, J., Ali, R.: Towards crowdsourcing for requirements engineering. In: Proc. of REFSQ Co-Located Events (2014)

[8] Nayebi, M., Cho, H., Ruhe, G.: App store mining is not enough for app improvement. Empirical Software Engineering, vol. 23, no. 5, pp. 2764–2794 (2018)

[9] Santos, R., Groen, E.C., Villela, K.: An overview of user feedback classification approaches. In: Proc. of REFSQ Co-Located Events, CEUR 2376 (2019)

[10] Dalpiaz, F., Parente, M.: RE-SWOT: From user feedback to requirements via competitor analysis. In: Proc. of REFSQ, pp. 55–70 (2019)

[11] Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C.A., Canfora, G., Gall, H.C.: How can I improve my app? Classifying user reviews for software maintenance and evolution. In: Proc. of ICSME, pp. 281–290 (2015)

[12] Guzman, E., Maalej, W.: How do users like this feature? A fine grained sentiment analysis of app reviews. In: Proc. of RE, pp. 153–162 (2014)

[13] Williams, G., Mahmoud, A.: Mining Twitter feeds for software user requirements. In: Proc. of RE, pp. 1–10 (2017)

[14] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of NAACL-HLT (2019)

[15] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of NAACL-HLT (2018)

[16] Wu, L.Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J. (2018). StarSpace: Embed all the things! ArXiv, abs/1709.03856.

[17] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. Technical report, OpenAI (2018)

[18] Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, vol. 22, pp. 1345–1359 (2010)

[19] Berry, D.M.: Evaluation of tools for hairy requirements engineering tasks. Empirical Software Engineering (in press) doi: 10.1007/s10664-021-09986-0

[20] Groen, E.C. et al., The crowd in requirements engineering: The landscape and challenges. IEEE Software, vol. 34, no. 2, pp. 44–52 (2017)

[21] Glinz, M.: CrowdRE: Achievements, opportunities and pitfalls. In: Proc. of RE Workshops, pp. 172–173 (2019)

[22] Santos, R., Groen, E.C., Villela, K.: A taxonomy for user feedback classifications. In: Proc. of REFSQ Co-Located Events, CEUR 2376 (2019)

[23] Wang, C., Daneva, M., van Sinderen, M., Liang, P.: A systematic mapping study on crowdsourced requirements engineering using user feedback. Journal of Software: Evolution and Process, vol. 11, no. 10, pp. e2199 (2019)

[24] Groen, E.C.: Crowd out the competition: Gaining market advantage through Crowd-based Requirements Engineering. In: Proc. of RE Workshops, 13–18 (2015)

[25] Stanik, C., Haering, M., Maalej, W.: Classifying multilingual user feedback using traditional machine learning and deep learning. In: Proc. of AIRE (2019)

[26] Castro-Herrera, C., Duan, C., Cleland-Huang, J., Mobasher, B.: Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes. In: Proc. of RE, pp. 165–168 (2008)

[27] Lim, S.L., Finkelstein, A.: StakeRare: Using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Transactions on Software Engineering, vol. 38, no. 3, 707–735 (2012)

[28] Renzel, D., Behrendt, M., Klamma, R., Jarke, M.: Requirements Bazaar: Social requirements engineering for community-driven innovation. In: Proc. of RE, pp. 326–327 (2013)

[29] Snijders, R., Dalpiaz, F., Brinkkemper, S., Hosseini, M., Ali, R., Ozum, A.: REfine: A gamified platform for participatory requirements engineering. In: Proc. of CrowdRE, pp. 1–6 (2015)

[30] Hosseini, M., Groen, E.C., Shahri, A., Ali, R.: CRAFT: A crowd-annotated feedback technique. In: Proc. of CrowdRE, pp. 170–175 (2017)

[31] Shirabad, J.S., T. Menzies.: The PROMISE repository of software engineering databases (2005)

[32] Navarro-Almanza, R., Juarez-Ramirez, R., Licea, G.: Towards supporting software engineering using deep learning: A case of software requirements classification. In: Proc. of CONISOFT, pp. 116–120 (2017)

[33] Tamai, T., Anzai, T.: Quality requirements analysis with machine learning. In: Proc. of ENASE (2018)

[34] Hey, T., Keim, J., Koziolek, A., Tichy, W.: NoRBERT: Transfer Learning for Requirements Classification. In: Proc. of RE, pp. 169–179 (2020)

[35] Sainani, A., Anish, P.R., Joshi, V., Ghaisas, S.: Extracting and Classifying Requirements from Software Engineering Contracts. In: Proc. of RE, pp. 147–157 (2020)

[36] Zhou, Y., Tong, Y., Gu, R., Gall, H.: Combining text mining and data mining for bug report classification. In: Proc. of ICSME, pp. 311–320 (2014)

[37] Guzmán, E., El-Haliby, M., Bruegge, B.: Ensemble methods for app review classification: An approach for software evolution (N). In: Proc. of ASE, pp. 771–776 (2015)

[38] "Term frequency by inverse document frequency," in Encyclopedia of Database Systems, p. 3035, 2009.

[39] Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. Journal of Machine Learning Research, vol. 2, pp. 139–154 (2001)

[40] Claster, W.: Naïve Bayes Classifier (2020)

[41] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146 (2017)

[42] Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. In: Proc. of ICLR (2013)

[43] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proc. of NIPS, pp. 6000–6010 (2017)

[44] Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: Proc. of EMNLP (2016)

[45] Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proc. of NAACL-HLT (2018)

[46] Agarwal, R. (2019, March 08). Attention, CNN and what not for text classification. Retrieved from https://towardsdatascience.com/nlp-learning-series-part-3-attention-cnn-and-what-not-for-text-classification-4313930ed566

[47] Groen, E.C., Kopczyńska, S., Hauer, M.P., Krafft, T.D., Doerr, J.: Users—The hidden software product quality experts? A study on how app users report quality aspects in online reviews. In: Proc. of RE, pp. 80–89 (2017)

[48] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. et al.: Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res., vol. 12, 2825–2830 (2011)

[49] Chollet, F.: Keras: The Python Deep Learning library (2018)

[50] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G. et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. NeurIPS (2019)

[51] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., Fidler, S.: Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: Proc. of ICCV, pp. 19–27 (2015)

[52] Mihaylov, T., Kozareva, Z., Frank, A. (2017). Neural skill transfer from supervised language tasks to reading comprehension. ArXiv, abs/1711.03754.

[53] Sarkar, D. (2018, December 04). Deep transfer learning for natural language processing: Text classification with universal embeddings. Retrieved from https://towardsdatascience.com/deep-transfer-learning-for-natural-language-processing-text-classification-with-universal-1a2c69e5baa9

[54] Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., Robinson, T.: One billion word benchmark for measuring progress in statistical language modeling. ArXiv, abs/1312.3005 (2014)

[55] Winkler, J.P., Grönberg, J., Vogelsang, A.: Optimizing for recall in automatic requirements classification: An empirical study. In: Proc. of RE, pp. 40–50 (2019)