Accessibility Rank: A Machine Learning Approach for Prioritizing Accessibility User Feedback

Xiaoqi Chai[1,2], James Tizard[1], Kelly Blincoe[1]

[1]Human Aspects of Software Engineering Lab, University of Auckland, New Zealand.
[2]School of Software, Beihang University, Beijing, China.

Abstract

Online user feedback, like app reviews, can provide valuable insights into software product improvements, offering development teams direct insights into customer experiences, preferences, and pain points. There are many studies that have proposed promising methods to automatically prioritize online user feedback, helping development teams identify the most salient software issues that need to be addressed. However, these methods may not take into account the accessibility-related needs of end users.

Our study addresses this limitation by developing a novel approach to analyze and prioritize app store reviews that discuss accessibility concerns. This new approach involves the evaluation of seven distinct machine learning (ML) algorithms, as well as three state-of-the-art large language models (LLMs), all leveraging features of app reviews relevant to accessibility. Utilizing validated accessibility reviews, we assess the effectiveness of our proposed approach and compare its performance with a leading general prioritization tool.

The results show that our novel method surpasses the leading general tool in prioritizing accessibility reviews, achieving an F1-score of 83.6%. This represents an improvement over the prior study's F1-score of 69.0%. Additionally, our approach outperforms the existing method across all three priority classifications, with the most notable improvement seen in the identification of high-priority reviews, where we achieved a +59.8% increase in F1-score. We hope our findings will inspire more research and innovation in this area and ultimately contribute to a more inclusive and accessible digital landscape for all users.

Keywords: Accessibility reviews, Prioritization, Machine learning, Requirements engineering

1 Introduction

In an increasingly digital world, the usability and accessibility of software applications have become key concerns for both developers and users alike. For example, consider a visually impaired user attempting to use a food delivery app. If the app lacks screen reader support or fails to provide accessible navigation options, the user may struggle to complete essential tasks such as placing an order. Situations like this highlight the importance of ensuring that applications are accessible to users of all abilities. To support this, various regulations and standards, such as the Web Content Accessibility Guidelines (WCAG) [1], have been developed to guide developers in creating inclusive applications.

Previous research has found that users report their accessibility concerns in online feedback, such as app reviews [2]. Accessibility issues can include vision, hearing, motor-control, and cognitive challenges, which limit or prevent the use of an application. Effective identification of these issues, in online feedback, can help developers enhance the overall usability of their applications [3], which in turn fosters inclusivity for diverse user groups, including people with disabilities [4].

However, as the number of users grows, manually extracting actionable insights from vast amounts of online feedback becomes increasingly challenging for developers [5]. This challenge is exacerbated by constraints such as limited budgets, time pressures, and a lack of awareness about accessibility, which contribute to many applications remaining inaccessible to users with disabilities [6, 7]. While many previous studies have proposed automatic methods for general user feedback prioritization [2, 8–11], such as the consensus algorithms utilized by Etaiwi et al. [9], and the PAID framework proposed by Gao et al. [10], there is still a lack of work specifically addressing the automatic prioritization of accessibility concerns in user feedback.

To address this gap, we first evaluate the state-of-the-art general prioritization tool proposed by Malgaonkar et al. [8], on its ability to prioritize app reviews for accessibility concerns. Malgaonkar's approach was selected due to it's excellent performance in general review prioritization, and its availability. Following that, we propose an approach specifically designed to identify the most pressing accessibility concerns in user feedback, leveraging state-of-the-art machine learning techniques. We conduct a comparative analysis of these approaches, aiming to identify the most effective method for prioritizing accessibility concerns. Ultimately, our findings aim to advance accessibility prioritization tools, enabling developers to focus on addressing the most impactful issues and creating more inclusive applications.

We were guided by the following high level research questions:

[RQ1] *What is the effectiveness of the existing prioritization method (multi-criteria heuristic prioritization technique) in prioritizing accessibility reviews?*

[RQ2] *Can a prioritization tool specifically designed to identity accessibility issues outperform a state-of-the-art general prioritization tool?*

The following are the key contributions of this paper:

1. We proposed a novel set of criteria to (manually) prioritize accessibility-related user issues based on their impact on app usability, and accessibility. We present content

coding guidelines for this new accessibility framework, and apply it to manually label a dataset of 307 accessibility reviews.
2. We evaluated a state-of-the-art general prioritization tool [8], and found that its performance dropped significantly when prioritizing reviews for impact on accessibility. This suggests that general prioritization tools may be insufficient, and that tools specifically designed to prioritize accessibility issues may be required.
3. We developed a novel prioritization method called Accessibility Rank, specifically designed for prioritizing accessibility reviews based on their impact on app usability and accessibility. This new approach outperforms the existing general review prioritization method, when applied to accessibility-related feedback. Accessibility Rank achieves a weighted F1 score of 83.6%. Additionally, Accessibility Rank significantly outperforms the existing method in identifying the important "high" priority accessibility reviews, with a F1 of 71.6%, compared to the existing tool's 11.8%.
4. We have made our research code and labeled data publicly available[1], and we encourage future researchers to use and improve upon our proposed approach.

The remaining sections of this paper are structured as follows: In Section 2, a thorough literature review is presented, highlighting the significance of user reviews and exploring related research on accessibility, the prioritization of user reviews, and the application of large language models in Requirements Engineering. Section 3 details the methodology utilized in this study. The results are presented in Section 4. Section 5 discusses the findings and their implications, potential threats to the validity, and potential avenues for future exploration. Lastly, Section 6 concludes the paper.

2 Related Work

In this section, an overview of previous research directly influencing the current work is presented. First, we discuss online user feedback and techniques proposed to help software development teams automatically identify the important product improvement insights more generally. Then we discuss accessibility concerns in online user feedback specifically. Finally, we explore the role of Large Language Models (LLMs) in requirements engineering.

2.1 Online User Feedback

In today's digital landscape, online user feedback is crucial in shaping software development. For instance, Ciurumelea et al. explore how reviews impact mobile app release planning [12], while Palomba et al. highlight the role of crowdsourced reviews in app evolution [13]. Fu et al. discuss the impact of negative feedback on user satisfaction [14]. Genc-Nayebi and Abran review opinion mining in app store reviews for improved development [15], and Pagano and Maalej show how app store feedback guides requirements engineering [16]. Given the large quantity of feedback that is available online, techniques have been proposed to automatically classify user feedback to help development

[1] https://zenodo.org/records/14890324

teams identify useful product improvement insights [17]. For instance, various techniques have been developed to classify user feedback into categories like bug reports and feature requests (e.g., [18], [19] [20]). Techniques have also been proposed to classify app reviews into specific categories (e.g, performance, resources, battery, memory, etc.) [21] [22] [23] and to uncover positive and negative sentiments in app reviews [24]. These classifications can help developers focus on the user feedback that is most likely to contain product improvement insights.

Building on this work, methods have also been proposed to prioritize online user feedback. For instance, Etaiwi et al. [9] introduced a consensus algorithm considering features such as frequency, post date, rating score, and category to prioritize reviews. By using this algorithm, developers can gain a more comprehensive understanding of user feedback and make more informed decisions about which issues to address first. Gao et al. [10] introduced the PAID framework for prioritizing user reviews at the phrase level. This framework enabled developers to track reviews across different versions of their applications, providing valuable insights into user feedback over time. Although effective, this approach was found to require a large number of reviews for meaningful insights. Malgaonkar et al. [8] proposed an automated prioritization approach consisting of entropy, frequency, TF-IDF and sentiment methods to rank informative user reviews. Their approach extended the rule-based approach proposed by Chen et al. [18].

While general prioritization methods have been applied to categorize user feedback, these approaches do not account for the unique needs of users with disabilities, such as those with vision, hearing, or motor impairments. In applications like mobile banking apps or e-commerce platforms, accessibility issues—such as improper screen reader support, missing alt text for images, or lack of keyboard navigation—can severely hinder users with disabilities. By prioritizing accessibility-related feedback, developers can directly address issues that impact these users, ultimately ensuring that software applications are usable by all, regardless of ability. This gap in research underscores the need for new methods that specifically prioritize accessibility-related concerns in user feedback, ensuring that these issues are given the attention they deserve.

It can also be noted that manual requirements prioritization techniques are well established in development teams, especially in the context of Agile development. For example, MoSCoW, Numerical scale, and Timeboxing/budgeting are well known within the software industry [25]. This work distinguishes its self by focusing on the automatic prioritization of requirements, focusing on improving application accessibility.

2.2 Accessibility Concerns in User Reviews

To ensure the accessibility of digital products, various accessibility guidelines and frameworks have been established such as the Web Content Accessibility Guidelines (WCAG) [1] and the BBC Accessibility Guidelines [26]. These guidelines offer valuable insights into best practices for inclusive design.

In a study conducted by Eler et al. [27], 213 keywords were extracted from the BBC standards and guidelines [26]. These keywords were employed to identify accessibility reviews given to 701 distinct apps, available on the Google Play Store. Using a manual

validation process, they found that only 1.24% (2263) of all reviews left to these apps were actually related to app accessibility. While the process required considerable manual effort, the study emphasized the importance of addressing these accessibility-related issues. However, an automated approach for efficiently identifying such reviews was not proposed in their work.

AlOmar et al. [2] designed a method to automatically identify accessibility issues within user reviews. Their approach categorized app reviews into two fundamental sets: those related to accessibility, and those unrelated to accessibility. They achieved an 85% accuracy rate. However, the approach did not aim to prioritse the identified accessibility reviews. Similarly, Aljedaani et al. [11] evaluated six diverse machine learning techniques to classify accessibility reviews into four distinct types aligned with accessibility guidelines: Principles, Audio/Images, Design, and Focus. Their evaluation obtained an accuracy rate of 93%. However, despite the advancements showcased by these methodologies, a common limitation persists: the absence of a prioritization mechanism based on the impact to application accessibility.

This study addresses the previously highlighted limitations in prioritizing accessibility reviews, by proposing an approach to prioritize accessibility-related reviews. We leverage the validated dataset provided by Eler et al. [27] as a foundation for developing and evaluating an automated approach to prioritize accessibility-related user feedback.

Our approach stands out as it is not solely focused on prioritizing accessibility reviews. Instead, it is specifically designed to prioritize them based on their impact on application accessibility. This unique focus fills a crucial gap left by previous methodologies, ensuring that the most critical accessibility concerns are addressed promptly. By focusing on impact, our approach not only enhances digital product accessibility but also ensures compliance with legal standards, such as WCAG (Web Content Accessibility Guidelines) [1], which require digital products to be accessible to all users, including those with disabilities. Additionally, it aligns with ethical principles of justice and equity, ensuring equal access for all users. This focus on legal compliance and fairness is key to fostering an inclusive digital environment.

2.3 Large Language Models in Requirements Engineering

In recent years, there has been a strong research focus on developing LLMs, where deep-learning (DL) models are trained using large volumes of text data. In particular, transformer-based DL architectures have achieved state-of-the-art performance in numerous natural language (NL) tasks. Transformer models lead on well established NL benchmarks (e.g., GLUE [28], SQuAD [29]), showing excellent performance in text classification, semantic similarity, sentiment analysis, question and answering, and more [30].

In requirements engineering, several recent studies have applied transformer models to NL tasks. In 2021, Devine et al. evaluated an extensive set of approaches, including transformer models, to group semantically similar user feedback. They found that transformer architectures, led by the Universal Sentence Encoder (USE) [31], outperformed the alternatives [32]. Tizard et al. leveraged USE to perform a semantic search,

identifying matching requirements between user forums and developer issues trackers [33]. Similarly, Haering et al. applied DistilBERT to match problem reports in app reviews to bug reports in issue trackers, achieving promising results.

Mekala et al. applied a BERT-based process to classify user reviews based on their relevance (or irrelevance) for requirements engineering tasks [34]. Their approach proved effective even in extremely low-volume dataset environments, outperforming the previous benchmarks. In 2024, a preliminary study from Sami et al. [35] applied OpenAI's GPT-3.5 to automatically prioritize requirements within the agile framework. Their paper details early work, which does not yet include a detailed evaluation of the approach.

Our study builds on the existing literature by applying transformer models to prioritize user reported issues, based on their impact on application accessibility. We apply state-of-the-art models that have recently outperform the alternatives on NL benchmarks, including T5, GPT4o, and o1-mini [30]. We also provide a detailed performance evaluation, based on a manually labeled dataset used as ground-truth, which is detailed in Section 3.1.

3 Method and Design

In this section, we present the design of our study. First, we describe the evaluation dataset utilized in this study. Following this, we introduce the state-of-the-art general prioritization approach, the multi-criteria heuristic prioritization technique proposed by Malgaonkar et al. [8], which has been selected as the benchmark approach. Next, we introduce our proposed prioritization methods that leverage features related to impact on accessibility. Finally, we describe the evaluation methodology.

3.1 Evaluation Dataset

In this study, we utilised the dataset of 2663 accessibility reviews[2] that were collected by Eler et al. [27]. It is worth noting that a separate study by AlOmar et al. [2] also confirmed these 2,663 reviews to be true-positive accessibility reviews. This validated dataset can help reduce the risk of errors and inconsistencies in the data, which could potentially impact the validity of the study's findings. Therefore, we utilized this dataset to conduct our analysis and the evaluation of prioritization approaches.

To enable prioritization of these reviews, we manually coded the priority of a subset of the reviews in this dataset according to their impact on user accessibility (described below). To conduct a robust and statistically representative analysis, we selected a random sample of 336 reviews from the dataset. This sample size was determined based on a 95% confidence level with a 5% margin of error, following best practices in quantitative research [36]. Random sampling was used to minimize selection bias.

Our manual coding process followed best practices as outlined by Krippendorff et al. [37]. While other content coding frameworks are available (e.g., Elo and Kyngäs [38], or Neuendorf [39]), Krippendorff's guidelines were familiar to the authors, and are well suited for purpose, with a focus on replicability, and bias mitigation. Additionally, the use of Krippendorff's guidelines is in-line with many recent requirements engineering

[2] https://github.com/marceloeler/data-ihc2019

studies (e.g., [40] [41], [20], [33]). The manual coding was primarily conducted by two authors, one with content analysis experience in several previous studies, and one with limited previous experience. A third author, with significant content analysis experience, gave guidance and oversight.

To ensure consistent and accurate manual labeling of reviews, we first developed a guideline for all coders. The creation of this guideline was inspired by the prioritization guideline used in the study by Malgaonkar et al. [8]. In this study, our prioritization criteria focused on assessing the impact of the reviews on the app's usability, aligning with our overarching goal of enhancing user experience, and removing barriers, for diverse user groups. While aspects like cost to implement features were not considered in this study's prioritization, future work could explore how additional factors can be incorporated into the priority assessment for real projects.

There were three priority levels for reviews: Low, Medium, and High. Table 1 illustrates the standard that we used for labeling reviews in our study, along with example reviews that fit into each priority level.

Table 1: Priority Assignment Guideline

Priority	**Description**	**Review Example**
Low	Reviews that mention nice-to-have enhancements or features with a limited impact on the core functionality of the app, typically reflecting personal preferences rather than critical issues.	*"Nice and useful application. One thing I would like to recommend is to have an option to change the font size of contents."*
Medium	Reviews that suggest improvements that affect app usability but do not prevent usage altogether, often addressing issues that would enhance the user experience for a broader group of users.	*"Having two events at the same time makes the calendar unreadable due to poor resizing."*
High	Reviews that report issues preventing or significantly hindering the use of the app, particularly those related to accessibility needs. These reviews often include phrases such as "I can't see," or "I can't use,", indicating urgent concerns that must be addressed for the app to remain accessible to the user.	*"What happened to the themes? My eyes are very light sensitive and I really appreciate dark themes. As is - I can't use the app long enough to really see how good it may be. Too uncomfortable and my eyes keep watering."*

The Low level was assigned to reviews that had a limited impact on the use of the app, typically involving nice-to-have enhancements or features. While these reviews may not be critical for the app's basic usage, it's worth noting that some of the reviewers might have specialized accessibility needs. These suggestions for improvements could potentially enhance the app's functionality, taking into consideration the diverse requirements of users with accessibility needs.

The Medium level encompassed reviews that had a moderate impact on the use of the app. These reviews typically highlighted issues that should be addressed or

enhancements that would improve user experience. While not critical, these issues or suggestions had a noticeable impact on the app's usability, taking into account the diverse needs of users, including those with specialized accessibility requirements.

The High level was assigned to reviews that significantly reduced or prevented the use of the app. These reviews represented must-have enhancements or features that were crucial for users, particularly those with specialized accessibility needs. Issues falling into this category had a severe impact on user experience and required immediate attention.

After developing the labeling standard guidelines, two coders independently manually analysed and assigned a priority label (high, medium, or low) to each review in our random sample. This process was performed in multiple rounds, with meetings to discuss and reconcile disagreements between coders. The discussions during meetings were focused on understanding each coder's perspective, clarifying any ambiguities in the guidelines, and finding common ground for assigning the most appropriate priority label. By engaging in this collaborative process, the coders aimed to minimize subjective biases and achieve a more objective and reliable assessment of the review priorities. In cases where the two coders were unable to reach an agreement after discussion, the respective reviews were discarded. In total 29 reviews were excluded due to coder disagreement. Upon inspection, we found that 23 out of the 29 excluded reviews (approximately 79%) contained mixed sentiments. For example, one review states: "This app is super useful! Kudos to the developer. I have just one issue of low contrast UI. The text throughout the app is hard to read at daylight. Other than that this app is great!" While the issue of low-contrast text may be serious for some users, the overwhelmingly positive tone made it difficult to assign a clear priority label, as different interpretations of the reviews led to disagreements among coders. Since our prioritization task requires clear judgments about the urgency of accessibility issues, we opted to exclude these ambiguous cases to maintain the consistency and reliability of the labeled dataset. Other excluded reviews were too vague (e.g., "ALLOW ME TO CHANGE THE VOLUME. ") or lacked sufficient context to determine the scope or severity of the accessibility concern. As a result, the final dataset consisted of 307 reviews. The impact of this reduction in sample size on the representativeness of the sample is further discussed in the Threats to Validity (Section 5.2).

The reliability of the manual labeling process was assessed by calculating intercoder reliability using the ReCal tool [42]. The results of this analysis (presented in Table 2) demonstrated a high level of agreement between the two main annotators, validating the accuracy and consistency of the labeling process. After resolving disagreements, a final dataset consisting of 307 reviews was retained for analysis. The reviews were categorized into three priority levels: High priority: 30 reviews, Medium priority: 81 reviews, Low priority: 196 reviews. The detailed distribution of reviews is shown in Table 3.

Table 2: Intercoder reliability

Dataset	**Initial Agreement**	**Cohen's Kappa**
336 accessibility reviews	91.4%	84.1%

Table 3: Ground truth dataset

Label	**Review Numbers**
High	30
Medium	81
Low	196
Total	307

3.2 Existing Prioritization Tool

In this paper, we chose the multi-criteria heuristic prioritization technique proposed by Malgaonkar et al. [8] as our benchmark method. This decision was made after careful consideration of several factors that establish its suitability and effectiveness for our research objectives. Firstly, the chosen technique has demonstrated high accuracy in prioritizing general app reviews. This is crucial for our study, as by selecting a benchmark method with a proven track record of accurately identifying general app reviews, we can ensure the reliability and validity of our comparisons.

Secondly, this prioritization technique is open source[3] and offers flexibility in its application. This adaptability is essential for addressing the specific requirements and challenges of our research domain. Furthermore, the functionality of the selected technique covers all the necessary data analysis steps, including pre-processing. This comprehensive coverage ensures that our benchmark method provides a complete framework for evaluating and comparing the effectiveness of our proposed approach.

3.2.1 Implementation of Existing Tool

The existing prioritization tool, multi-criteria heuristic prioritization technique, was initially designed to identify and rank general app reviews based on various factors. It employed a combination of features, including entropy, frequency, TF-IDF, and sentiment score to prioritize the app reviews. The general priority (P_R) score of a review ($_R$) is calculated according to function 1, which takes into account these various features to prioritize reviews that are deemed the most informative and relevant.

$$P_R = \alpha \cdot E_R + \beta \cdot F_R + \gamma \cdot \text{TF-IDF}_R + \delta \cdot (-(SC_R)) \tag{1}$$

In equation 1, E_R, F_R, $TF - IDF_R$, and SC_R represent the normalized priority score of reviews generated by the entropy variable, frequency variable, TF-IDF variable, and sentiment variable, respectively, and the values of all constants should be set in such a way that it satisfies the constraint $\alpha+\beta+\gamma+\delta = 1$. In their study, these four constants were all set to 0.25, achieving an overall accuracy of 77.22% while prioritizing general app reviews. The value of 0.25 was chosen as it represents the default seed value for these variables, as recommended by prior research [43]. This default recommendation provides a balanced starting point for the prioritization model, ensuring a consistent and fair evaluation across different datasets.

[3] https://tinyurl.com/y4hynj5j

3.3 New Prioritization Approach

We evaluated both traditional machine learning (ML) algorithms, as well as state-of-the-art large language models (LLMs) in prioritizing accessibility focused app reviews. The ML approach is detailed below in Section 3.3.1, and then the LLMs approach is detailed in Section 3.3.2. The evaluation results for both approaches are given in the results (Section 4).

3.3.1 Traditional Machine Learning Prioritization

In this section, we applied machine learning (ML) techniques to prioritize accessibility-related reviews. The process involves several stages: dataset splitting, feature extraction, model training, and model evaluation.

Data Splitting

The dataset, comprising 307 samples, was divided into a training set (80%) and a testing set (20%). The training set was used for model development and hyperparameter tuning, while the test set was reserved for evaluating the performance of the final models. The model training is detailed later in this section (under, Model Training and Hyperparameter Tuning), and the evaluation is detailed in Section 3.4.

Feature Extraction and Selection

General Priority Score: To get the general priority score, we first calculated the priority scores of the accessibility reviews by calling the multi-criteria heuristic prioritization method [8]. To ensure consistency and reliability, we used the same variable settings as applied in the original study. This step is necessary to eliminate any potential bias that may have been introduced by using different settings. Next, we converted these numerical scores into meaningful categories, namely "High", "Medium", and "Low". The numerical ranges for these categories were determined based on the thresholds and guidelines established in Malgaonkar et al. [8]. Specifically, these ranges were designed to align with their approach for balancing scores across categories while maintaining meaningful differentiation between them. The numerical thresholds are detailed in Table 4. These thresholds were directly derived from prior work and validated through their alignment with domain-specific prioritization frameworks, ensuring that the conversion accurately reflects practical priorities.

Table 4: Priority score and numerical range conversion

Priority Label	Numerical Range
Low	0 - 0.3
Medium	0.4 - 0.6
High	0.7 - 1.0

User Ratings: User ratings refer to a mechanism that allows users to express their opinion or evaluation of a product, service, or experience. It has been widely utilized in

various studies for prioritizing reviews (e.g [44], [45], [18], [9]). A rating score of 1 indicates the lowest level of satisfaction or quality, which means the user had a very poor experience with the product or service, while a rating score of 5 indicated the highest level, which means that the user had an excellent experience. Notably, low rating scores often correlate with difficulties in using or accessing the app, highlighting the critical connection between user satisfaction and accessibility/usability challenges [46]. The distribution of user ratings in the labeled dataset is shown in Figure 1. The high-priority reviews tend to have lower users ratings, with an average of 1.9. The lower priority reviews tend to have higher user ratings, with the medium-priority average being 3.3, and the low-priority average being 4.6.

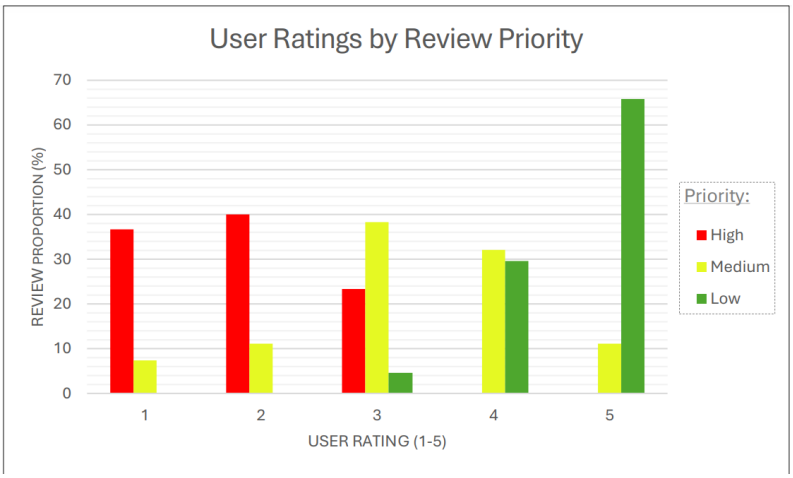

Fig. 1: User rating proportions in labeled app reviews. User ratings (*x-axis*) are given with an app review. A rating of 1 indicates the lowest level of satisfaction, while a rating score of 5 indicates the highest satisfaction level. The proportion of different ratings, per manually labeled priority level (*high, medium, low*), are shown.

Model Training and Hyperparameter Tuning

We trained seven different ML algorithms-Support Vector Machines (SVM), Random Forest (RF), Gradient Boosting Machines (GBM), and others to prioritize accessibility-related reviews. Each model undergoes hyperparameter tuning to maximize performance, followed by model evaluation using several key metrics.

To optimize the performance of each model, we used grid search, a method that systematically tests all possible combinations of hyperparameters within a predefined search space. We selected grid search for the following reasons:

Parameter Space Size: Since the parameter space for our models is relatively small, grid search is computationally feasible and provides a thorough search for the best hyperparameter combination.

Reproducibility: Grid search ensures high reproducibility of results, which is crucial for validation and transparency in machine learning experiments.

While methods like random search and Bayesian optimization are more efficient for larger search spaces, grid search is appropriate for our study's scope. We also combine grid search with k-fold cross-validation (with k=5) to ensure the hyperparameters are optimized across different data subsets, enhancing the robustness and generalization of the model.

Model Evaluation

Model performance was evaluated using the weighted F1 score, which is particularly suitable for imbalanced datasets. This metric provides a balanced measure of precision and recall, accounting for the unequal distribution of classes in our dataset. Additionally, we tracked the standard deviation of F1 scores across the folds of cross-validation to assess the stability of each model's performance. A lower standard deviation indicates more consistent performance across different subsets of the training data.

The final evaluation of each model was conducted on the separate test set to assess its real-world performance. This step ensures that the model's ability to generalize to unseen data is accurately measured. A detailed explanation of the evaluation metrics and the rationale behind their selection can be found in Section 3.4.

3.3.2 Large Language Model Prioritization

In addition to traditional ML models, we also evaluated state-of-the-art large language models (LLMs) in prioritizing accessibility focused app reviews. We investigated two distinct approaches, based on different model architectures, which are described in detail below.

Classification with Text Embeddings

Language models that directly output vectorized text embeddings have been shown to achieve state-of-the-art performance on a variety of natural language tasks, including text classification [47], [33]. Once vectorized, text embeddings that are closer in the high-dimensional vector space can be considered more semantically similar. Classification is performed by first creating an embedded centroid representing each possible classification, then embedding the text documents to be classified, and finally identifying their respective closest class centroid. This process is described in detail below.

Algorithm selection: In this work we applied Sentence-T5 to generate text embeddings. Sentence-T5 was selected as it has been found to outperform other leading models (e.g., USE [31], Sentence-BERT [48]), including in requirements engineering tasks [47], [30]. While Sentence-T5 is an excellent option for this study, we discuss the potential evaluation of alternative models in the Threats to Validity (Section 5.2). To measure the distance between text embeddings (and therefore semantic similarity),

we applied cosine distance. Cosine distance has previously been found to outperform other similarity measures for dense vector embeddings, such as Sentence-T5 embeddings [49], [33].
Preprocessing: For each app review, it's associated user rating was prepended to it's review text, with a label (i.e., User rating: {rating}) before embedding, so the model could use it in it's prediction. An additional label was also added before the app review text (i.e., App review: {review-text}), to clearly identify it.
Classification process: An illustrative example of the classification process is shown in Fig. 2. Similar to the process outlined by Tizard et al. [33], we first randomly selected a subset (detailed below) of the labeled app reviews, per possible classification (High, Medium, Low), as class references reviews. Next, we embedded each class reference review using Sentence-T5, and averaged the embeddings, per class, to find the three class centroids. The reference app reviews were then removed from the dataset, with the remaining reviews used for the evaluation.

During evaluation, the app reviews, not used as references, were each embedded using Sentence-T5, and the semantic similarity scores to the three class centroids were calculated. The most semantically similar class was taken as the prediction for each app review. Finally, the predicted class for each review is assessed as being correct when it matches the manually assigned label. The evaluation metrics are described in detail in Section 3.4.

To evaluate this approach, we applied k-fold cross validation, a common technique for evaluating the performance of a classifier [50]. Here, we used five-fold cross validation, a good option according to the variance bias trade-off [51]. Specifically, each of the five folds takes a turn being the test dataset, with the other folds acting as reference reviews to form the centroids (i.e., the training data). This was repeated five times, to use each fold as the test set. Finally, the evaluation metrics were averaged across the five runs. As there were no parameters tuned for this model, all data was used in the cross-valuation.

Classification with Text-to-Text Models

Pre-trained text-to-text models take a sequence of input text and generate corresponding output sequences. This model architecture has been found to achieve excellent performance in many natural language tasks, including text classification [30].
Algorithm selection: In this work we evaluated OpenAI's GPT-4o, and o1-mini text-to-text models, through their developer API[4]. OpenAI's models were selected as they have achieved best-in-class performance on many natural language benchmarks [52]. The focus on GPT-4o and o1-mini is further discussed in the Threats to Validity (Section 5.2)
Classification process: The input text for a language model is often called a prompt, with prompt engineering emerging as a research focus. We followed current best practices in prompt engineering (detailed below), as described by Sahoo et al. [53], in their survey of prompt engineering techniques. The prompts we applied to both models are shown in Fig. 3. We passed each review to be classified in its own API call (307 total calls), with no information retained between each call.

[4]https://platform.openai.com/docs/quickstart

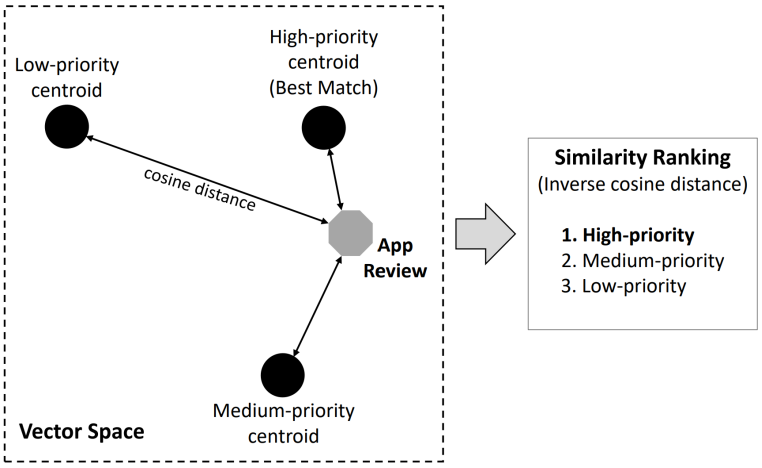

Fig. 2: Illustrative example: Predicting the classification priority for each app review. Each review is vectorized with Sentence-T5 and the most semantically similar class is calculated.

System message	You are a helpful assistant. Below you will be given an app review, with a user describing an issue they are having with a mobile app. You will also be given the associated rating the user gave with the review, where 1 is the lowest rating, and 5 is the highest rating. Please estimate how severely the issue is impacting the user's ability to use the app, on the following scale: 1. Low. - A review that has a limited impact on the use of the app. (nice to have enhancements/features) 2. Medium. - A review that has a moderate impact on the use of the app.(should have enhancements/features) 3. High. - A review that significantly reduces, or prevents, the use of the app. (must have enhancements/features) Include the number of the estimate you give, and identify it by surrounding it with *, i.e., *1*, *2*, or *3*.
Review message	The associated rating the user gave is: **{rating}** The user's review of the app is: **{review}**

Fig. 3: GPT-4o and o1-mini API input messages (prompts).

For each API call, we passed both an unchanging system message, and a review message, containing the app review to be classified, with its associated user rating. Our system message, gives context to the model about the task to be performed, including

the model's role (helpful assistant), problem domain (mobile app reviews), and the classification task description and scale. Additionally, each possible classification is given an associated numerical identifier (*1*, *2*, *3*) to easily identify the models classification within it's text output, using standard string matching (Python).

We chose to use zero-shot prompting, where no reference examples are given in the prompt messages. Using zero-shot ensures the model will not over-fit to any given examples, and is evidence for the generalizability, and ease of implementation of the approach. We achieved promising results with zero-shot prompts (see Section 4), however few-shot (or more) prompting has the potential to improve performance. Our focus on zero-shot prompting is discussed in the Threats to Validity (Section 5.2).

Finally, reasoning approaches like Chain-of-Thought (CoT) prompting can encourage the model to take a step-by-step reasoning approach, often improving performance [53]. For example, including in the prompt, "Let's think step-by-step" will generate reasoning chains in the output text. In this work, we did not explicitly prompt Chain-of-Thought reasoning, but instead, in addition to the standard GPT-4o model, also called the o1-mini model, which has been configured to undertake a reasoning approach [54]. We found the reasoning o1-mini model modestly improved performance over the standard GPT-4o (see Section 4).

During the evaluation, the predicted class for each app review is assessed as being correct when it matched the manually assigned label. The evaluation metrics are described in detail in Section 3.4. Unlike the other approaches evaluated in this work, this text-to-text approach had no "training data" (zero-shot), and therefore all 307 labeled app reviews were used directly used as test data.

3.4 Evaluation Methodology

Evaluation Metrics

To assess the effectiveness of the prioritization techniques, we employed multiple classification performance metrics, as summarized in Table 5. Given the inherent class imbalance in our dataset, we employed weighted average scores as our evaluation metrics. Here the evaluation parameters (i.e., *TP, FP, TN, FN*) and the associated metrics are first found for each individual class, then averaged based on each classes prevalence, or weight (Wi), in the total dataset [55]. Weighted metrics are particularly suitable for imbalanced classification scenarios, as they assign a proportional weight to each class based on its prevalence in the dataset. This ensures that minority classes are adequately represented, preventing the evaluation from being skewed by majority classes [56].

4 Results

In this section, we delve into the analysis and interpretation of the results obtained from our research, focusing on addressing the research questions that guided our study.

Table 5: Performance Metrics and Definitions

Metric	Formula	Definition
Precision (Weighted)	$\sum_{i=1}^{N} w_i \frac{TP_i}{TP_i+FP_i}$	The weighted average precision across all classes, adjusting for class imbalance to prevent bias toward majority classes.
Recall (Weighted)	$\sum_{i=1}^{N} w_i \frac{TP_i}{TP_i+FN_i}$	The weighted average recall across all classes, ensuring fair representation of minority classes.
F1 Score (Weighted)	$\sum_{i=1}^{N} w_i \frac{2\times Precision_i Recall_i}{Precision_i+Recall_i}$	The weighted harmonic mean of precision and recall, balancing both metrics to account for class imbalance.

4.1 RQ1 Results

[RQ1] *What is the effectiveness of the existing prioritization method (multi-criteria heuristic prioritization technique) in prioritizing accessibility reviews?*

Table 6 presents the evaluation of the multi-criteria heuristic prioritization technique in prioritizing accessibility reviews, focusing on three priority levels: High, Medium, and Low. The performance metrics include precision (Prec.), recall (Rec.), F1-score (F1), and support (the number of reviews in each priority category). Additionally, accuracy (Acc) and weighted average metrics are provided to summarize the method's overall performance on the dataset.

Overall, the existing method achieves a weighted precision of 77.0%, a recall of 65.0%, and an F1 score of 69.0%. These results indicate moderate effectiveness in prioritizing accessibility reviews, with significant variations across different priority levels.

For the "High" priority level, precision and F1 scores were notably low, reflecting the method's difficulty in accurately identifying and prioritizing reviews with high-severity issues. This can be attributed to the small support size and the inherent challenges in distinguishing features associated with high-priority accessibility concerns.

In contrast, the technique demonstrated strong performance for the "Low" priority level, achieving a precision of 88.0% and a robust F1 score of 77.0%. This indicates that the method effectively identifies and prioritizes reviews with low severity, with a high proportion of true positives among the predicted "Low" priority reviews.

For "Medium" priority reviews, the method exhibited moderate effectiveness, with a balance between precision, recall, and F1 score, though these metrics leave room for further refinement to better address medium-severity accessibility issues.

The results highlight that while the multi-criteria heuristic prioritization technique performs well for "Low" priority reviews, it struggles significantly with "High" priority reviews due to limited support and feature differentiation challenges. Medium-priority reviews also require methodological enhancements to achieve greater accuracy and consistency. Later we will discuss this in Section 5.

Table 6: Performance of the multi-criteria heuristic prioritization technique across priority levels

Priority Level	Acc.(%)	Prec.(%)	Rec.(%)	F1(%)
High	6.7	6.7	50.0	11.8
Medium	21.0	21.0	40.5	27.9
Low	89.8	89.8	67.7	77.0
Weighted Avg	**63.5**	**79.3**	**63.5**	**69.4**

Answer to RQ1: The selected benchmark prioritization technique demonstrates moderate performance in accessibility review prioritization, with an F1 score of 69.4%. It performs well on low-priority (Low) reviews, achieving high precision and recall. However, it is less effective on high-priority (High) and medium-priority (Medium) reviews, indicating that the method is less effective in handling complex or critical accessibility issues.

4.2 RQ2 Results

[**RQ2**] *Can a prioritization tool specifically designed to identity accessibility issues outperform a state-of-the-art general prioritization tool?"*

To address RQ2, we evaluated the performance of our proposed approach against the existing benchmark method, evaluated in RQ1. For our approach, We considered seven traditional ML algorithms, as well as three LLM-based methods. Table 8 presents the performance of each approach when applied to our manually labeled accessibility dataset (see Section 3.1). The evaluated algorithms are ordered by their respective weighted F1-scores, with accuracy, weighted precision, weighted recall, and the standard deviation of the F1-score also presented.

To aid in the reproducibility of our results, we provide the selected main parameters of the ML techniques used in our study. These parameters are summarized in Table 7.

Table 7: Best hyperparameters for each ML algorithm

Algorithm	Hyperparameter	Value
MNB	alpha	0.1
DT	max_depth	5
	min_samples_split	10
SVM	C	10
	kernel	linear
RF	max_depth	5
	n_estimators	200
ETC	max_depth	5
	n_estimators	50
GBM	learning_rate	0.01
	n_estimators	200
AdaBoost	learning_rate	0.01
	n_estimators	50

Table 8: Comparison of different algorithms' performance

Algorithm	**Acc.(%)**	**Prec. (%)**	**Rec. (%)**	**F1-Score (%)**	**Std. Dev.** (F1%)
o1-mini	84.0	83.8	84.0	**83.6**	NA
GPT-4o	83.4	83.4	83.4	82.9	NA
Sentence-T5	79.7	83.1	79.7	80.2	4.75
ETC	77.4	78.1	77.4	77.4	1.92
RF	75.8	77.6	75.8	76.5	1.86
GBM	72.6	74.6	72.6	73.4	4.36
SVM	72.6	73.2	72.6	72.9	4.42
DT	71.0	72.3	71.0	71.6	4.56
ADA	74.2	67.2	74.2	70.5	4.29
MNB	64.5	41.6	64.5	50.6	1.03

Note: All metrics (Precision, Recall, F1-Score) reported are weighted averages, calculated according to the class distribution in the dataset. For algorithms using cross-validation, the reported values represent the mean of the scores across all folds. The standard deviation (Std. Dev.) of the F1-Score is provided for models that employed cross-validation; for models without cross-validation (e.g., o1-mini and gpt-4o), "NA" (Not Applicable) is noted.

Overall, nine of the ten evaluated algorithms outperformed the benchmark tool's 69.4% F1-score. OpenAI's o1-mini model achieved the top F1-score of 83.6%, outperforming the baseline by +14.2%. Additionally, our proposed approach demonstrated more consistent performance across the three possible classifications (high, medium, low). The per-class performance for the top performing o1-mini model is presented in Table 9, with the per-class performance of all evaluated algorithms given in our replication package.

o1-mini significantly outperformed the baseline across all three classes. In the important high-priority class, o1-mini achieved an F1-score of 71.6%, compared to just 11.8% for the baseline (*+59.8%*). In the medium-priority class, o1-mini achieved an F1-score of 66.7% compared to 27.9% (*+38.8%*). Finally, in the low-priority classification, o1-mini achieved 92.5% compared to 77.0% (*+15.5%*).

Table 9: Performance of o1-mini across priority levels

Priority Level	**Acc.(%)**	**Prec.(%)**	**Rec.(%)**	**F1(%)**
High	93.8	64.9	80.0	71.6
Medium	84.0	74.2	60.5	66.7
Low	90.2	90.7	94.4	92.5
Weighted Avg	**84.0**	**83.8**	**84.0**	**83.6**

Answer to RQ2: We found that an LLM based approach (*o1-mini*), specifically designed to identify acceptability issues, outperformed Malgaonkar et al.'s [8] general prioritization tool, in prioritizing accessibility-reviews. Our proposed approach achieved an F1-score of 83.6%, compared to the existing tool's 69.4% (*+14.2%)*.

Additionally, the proposed approach outperformed the baseline within all three priority classifications, especially in identifying the important high-priority class (*+59.8% F1-score*).

5 Discussion

In this section, we discuss the implications of our research. We also describe potential threats to the validity of our results, and outline avenues for future research to further enhance the effectiveness of accessibility prioritization methods.

5.1 Implications

The following section discusses the implications of our findings, and makes recommendations for both practitioners and researchers.

Identifying accessibility requirements in practice: Previous work found accessibility requests only make up 1.24% of all app reviews [27]. Therefore, standard methods that prioritize issues based on their popularity will almost certainly miss many significant accessibility issues. Similar equity issues in "crowd-sourcing" software requirements online have been highlighted in previous work. For example, Tizard et al. found that online feedback platforms, such as app stores, are significantly over-represented with men (compared to women), and users 35-44 years old, compared to those older and younger [57, 58]. As there is no direct method to identify user demographics on app stores, requirements sourced from app stores will inevitably under-represent less vocal groups (e.g., women). Tizard et al. proposed that alternative channels would be needed to engage these underrepresented groups.

In line with these recommendations, extra steps are needed to ensure accessibility related feedback is not missed, given their sparsity. We envision Accessibility Rank being used as part of a pipeline to help developers identify the most pressing accessibility concerns of their users. Starting with an unfiltered set of app reviews (for a single application), requirements engineers would first identify accessibility-relevant reviews, using an existing accessibility classification tool, such as that proposed by AlOmar et al. [2]. Next the identified accessibility reviews would be prioritized using Accessibility Rank, to help identify the most severe issues impacting the use of the application. Finally, development teams can weigh the high-priority accessibility issues against those found with traditional methods, considering not only business factors, but also equity within their user-base.

Importance of accessibility: With business pressures, such as time and resource constraints, accessibility is often overlooked in modern software development. Leveraging automated tools, such as Accessibility Rank, can help development teams efficiently

identify their users' most salient accessibility needs, and yield significant benefits. Our prioritization approach focuses on app usability and accessibility, distinguishing it from general prioritization approaches. While other approaches prioritize reviews based on criteria such as request frequency, we not only consider this, but emphasise the impact on access to the application.

Although accessibility requests only make up 1.24% of all app reviews [27], this likely significantly under-represents the needs of actual user-bases, as accessibility needs in the general population are much higher than 1.24%. For example, as of 2021, the World Health Organisation estimated that approximately 1.3 billion people – about 16% of the global population, live with some form of disability [59].

Additionally, designing products with the needs of under-represented users in mind can lead to improvements that benefit a much broader audience. Take closed captioning for example, originally introduced to assist individuals with hearing impairments. Over time, its usefulness has expanded, particularly with the rise of social media. Captions now enhance accessibility for all users, allowing them to watch videos without sound while scrolling through their feeds or even engage with content in different languages [60]. Therefore, addressing accessibility concerns is not only important from an equity stand-point, but may also benefit an application by growing its user-base.

5.2 Threats to Validity

5.2.1 Internal Validity

One potential threat to internal validity in this study stems from the manual (priority) labeling of accessibility reviews, which introduces the possibility of bias. To mitigate this, we developed a labeling guideline, offering detailed descriptions for each priority level, along with illustrative examples. This framework aimed to standardize the labeling process and enhance consistency. However, it is important to note that the guideline might have unintentionally introduced bias, especially if it did not fully capture the diverse range of accessibility concerns or if it emphasized certain types of feedback over others. In future studies, a more inclusive review of the guidelines by accessibility experts and user feedback could help ensure that they comprehensively address all aspects of accessibility needs.

To further reduce bias, we employed two independent coders who achieved an inter-coder reliability of 91.4%, demonstrating a high level of agreement. Disagreements between the coders were resolved by excluding those reviews from the truth set, ensuring that only consistent classifications were included in the final analysis. Although this approach helped maintain the accuracy of the dataset, it is important to acknowledge that the exclusion of reviews due to disagreements may still introduce some bias, particularly if the ambiguous reviews were systematically different from the rest in terms of content or priority. The percentage of discarded reviews remained low, at 8.6%, which suggests a limited impact on the overall validity of our findings. However, further analysis into the characteristics of the excluded reviews could provide additional insights. For example, if these reviews were disproportionately from the "high" or "medium" priority categories, their exclusion might have impacted the distribution

of priorities. Additionally, exploring alternative methods of resolving coder disagreements, such as employing more annotators, might help preserve more reviews while still ensuring the quality of the labeling process.

Additionally, the dataset used in this study was imbalanced, with varying quantities of reviews across different priority classes (low, medium, and high). The disproportionate representation of the different priority levels could influence the prioritization results, as models might be biased toward the majority class. To address this, we employed weighted evaluation metrics, such as weighted average score, which help mitigate the impact of class imbalance by giving more importance to minority classes, ensuring a more balanced assessment. While this approach reduces the risk of imbalance-related bias in the evaluation phase, it is important to acknowledge that the skewed distribution of the dataset could still impact the model's generalizability to real-world scenarios, where such imbalances are often present.

Another potential internal validity concern is related to the selection of features utilized by the proposed approach. While we focused on these key features, it is acknowledged that additional, unexplored features could also be highly informative for prioritizing accessibility reviews. Factors such as implementation cost and other contextual features may play a significant role in determining the priority of accessibility issues. Although our analysis is limited to the features explored in this work, this study presents a promising first step in developing a prioritization method. Future work should aim to expand the set of features, incorporating both technical and practical aspects, to create a more comprehensive and realistic prioritization framework.

In terms of model selection, we evaluated seven traditional ML models, and three LLMs, that have demonstrated excellent performance in related applications [30]. Additionally, we focused on zero-shot prompting in our text-to-text LLM classifier (see Section 3.3.2), to demonstrate the generalizability of the approach. Future work can investigate the effectiveness of alternatives models and techniques (e.g., few-shot prompting), especially as new state-of-the-art LLMs are developed. While effective alternative models may exist, our proposed approach significantly outperformed the benchmark, and represents a promising step in the automatic identification of user reported accessibility issues.

5.2.2 External Validity

A potential threat to external validity is the impact of the reduced sample size on statistical confidence. During the labeling process, 29 reviews were excluded. As a result, the margin of error slightly increased from $\pm 5.0\%$ to $\pm 5.3\%$, introducing a minor increase in uncertainty. However, the confidence level remains 95%, ensuring the findings are still statistically robust. While this reduction in sample size slightly decreased the confidence level of our results, the effect was minimal and does not significantly impact the validity of the findings.

Another potential threat is the generalizability of our findings. Our evaluation utilized a dataset from a previous study [27], specifically centered on English-language reviews, which may not fully represent accessibility concerns expressed in other languages. Future studies should explore multilingual datasets to assess the applicability of our approach across diverse linguistic contexts.

Additionally, our dataset was derived solely from the Google Play Store, excluding other platforms such as the Apple App Store. Since accessibility issues and review patterns may vary across platforms, our findings may not be fully generalizable to other app ecosystems. Expanding this research to include multiple platforms could provide a more comprehensive understanding of accessibility-related feedback.

5.3 Future Work

Based on the threats to validity and limitations of our study, there are several potential avenues for future work. One possible direction would be to explore the impact of additional features on the prioritization of accessibility reviews. While our current approach concentrates on some prominent features, it is worth exploring the inclusion of additional factors, such as cost to the implement, to further enhance the optimization of accessibility review prioritization. Additionally, researchers could investigate the impact of different ML algorithms on the prioritization of accessibility reviews, as well as explore the use of different deep-learning methods and LLMs, to improve the accuracy and effectiveness of our approach.

In addition, future work can extend the current scope of accessibility reviews prioritization approach by further developing the ground truth dataset. This could involve adding more accessibility reviews from sources beyond Google App Stores and incorporating non-English reviews. By doing so, we can gain a better understanding of the accessibility-related phrases that are commonly used in app reviews. This expanded dataset will enable us to improve the accuracy and effectiveness of our approach, as well as provide valuable insights into the language and terminology used by users when discussing accessibility issues. Additionally, expanded dataset will allow us to explore other potential features that may impact the prioritization of accessibility reviews, further enhancing the flexibility and customization of our approach. We also plan to explore the impact of these excluded cases on the model's performance in future work to better understand how they might influence the results.

6 Conclusion

In this study, our primary goal was to design an effective method for prioritizing accessibility reviews. We began by exploring the multi-criteria heuristic prioritization technique [8], which has shown strong performance in prioritizing general app reviews. However, we identified that this technique, while effective in its broader application, lacked the specificity necessary for addressing accessibility-related concerns.

To address this gap, we introduced Accessibility Rank, a novel approach that combines the multi-criteria heuristic prioritization technique with machine learning (ML) and large language models (LLMs). We evaluated several different algorithms as part of this new framework, and found that the o1-mini model outperformed the others, leading to its integration into our final approach.

Our evaluation demonstrated that Accessibility Rank (o1-mini) significantly outperforms the benchmark approach in prioritizing accessibility reviews. With a weighted

F1 score of 83.6%, our model showed a substantial improvement over the benchmark's weighted F1 score of 69.4%, highlighting its superior capability in accurately prioritizing accessibility issues.

Additionally, Accessibility Rank excels not only in overall performance but also across individual labels, offering improved accuracy in prioritizing accessibility concerns with nuanced differentiation across various categories. This makes our approach more effective in addressing the diverse range of accessibility issues that may arise in app reviews.

While this study has provided valuable insights into the potential of Accessibility Rank, there are several opportunities for future enhancements. Expanding the dataset to include reviews from additional platforms beyond the Google Play Store and integrating other state-of-the-art LLMs could further improve the relevance and accuracy of our approach. Furthermore, refining the model's ability to handle a wider variety of accessibility issues will help ensure it remains robust and adaptable to evolving needs.

In conclusion, the development of Accessibility Rank represents a significant step forward in helping development teams prioritize accessibility concerns within app reviews. This tool not only improves product development processes but also fosters a more inclusive user experience. Moving forward, we remain committed to advancing this research and exploring new methodologies to refine accessibility prioritization, contributing to a more accessible and equitable digital environment for all users.

References

[1] W3C: Web Content Accessibility Guidelines (WCAG) Overview. https://www.w3.org/WAI/standards-guidelines/wcag/. Accessed: October 2022 (2018). https://www.w3.org/WAI/standards-guidelines/wcag/

[2] AlOmar, E.A., Aljedaani, W., Tamjeed, M., Mkaouer, M.W., El-Glaly, Y.N.: Finding the needle in a haystack: On the automatic identification of accessibility user reviews. Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (2021) https://doi.org/10.1145/3411764.3445281

[3] Yan, S., Ramachandran, P.G.: The current status of accessibility in mobile apps. ACM Transactions on Accessible Computing (TACCESS) **12**, 1–31 (2019)

[4] Heron, M., Hanson, V.L., Ricketts, I.: Open source and accessibility: advantages and limitations. Journal of Interaction Science **1**(1), 2 (2013) https://doi.org/10.1186/2194-0827-1-2

[5] Stanik, C., Haering, M., Maalej, W.: Classifying multilingual user feedback using traditional machine learning and deep learning. In: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), pp. 220–226 (2019). IEEE

[6] Crabb, M., Heron, M., Jones, R., Armstrong, M., Reid, H., Wilson, A.: Developing accessible services: Understanding current knowledge and areas for future

support. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, pp. 1–12 (2019). https://doi.org/10.1145/3290605.3300446 . Association for Computing Machinery

[7] Putnam, C., Wozniak, K., Zefeldt, M.J., Cheng, J., Caputo, M., Duffield, C.: How do professionals who create computing technologies consider accessibility? In: Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility, pp. 87–94 (2012). https://doi.org/10.1145/2384916.2384932 . Association for Computing Machinery

[8] Malgaonkar, S., Licorish, S.A., Savarimuthu, B.T.R.: prioritizing user concerns in app reviews – a study of requests for new features, enhancements and bug fixes. Information and Software Technology **144**, 106798 (2022) https://doi.org/10.1016/j.infsof.2021.106798

[9] Etaiwi, L., Hamel, S., Guéhéneuc, Y.-G., Flageol, W., Morales, R.: Order in chaos: prioritizing mobile app reviews using consensus algorithms. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 912–920 (2020). https://doi.org/10.1109/COMPSAC48688.2020.0-151

[10] Gao, C., Wang, B., He, P., Zhu, J., Zhou, Y., Lyu, M.R.: Paid: prioritizing app issues for developers by tracking user reviews over versions. IEEE Computer Society, USA (2015). https://doi.org/10.1109/ISSRE.2015.7381797

[11] Aljedaani, W., Mkaouer, M.W., Ludi, S., Javed, Y.: Automatic classification of accessibility user reviews in android apps. In: 2022 7th International Conference on Data Science and Machine Learning Applications (CDMA), pp. 133–138 (2022). https://doi.org/10.1109/CDMA54072.2022.00027

[12] Ciurumelea, A., Schaufelbühl, A., Panichella, S., Gall, H.C.: Analyzing reviews and code of mobile apps for better release planning. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 91–102 (2017). https://doi.org/10.1109/SANER.2017.7884612

[13] Palomba, F., Linares-Vásquez, M., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D., De Lucia, A.: User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In: 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 291–300 (2015). https://doi.org/10.1109/ICSM.2015.7332475

[14] Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., Sadeh, N.: Why people hate your app. Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13 (2013) https://doi.org/10.1145/2487575.2488202

[15] Genc-Nayebi, N., Abran, A.: A systematic literature review: Opinion mining studies from mobile app store user reviews. Journal of Systems and Software **125**,

207–219 (2017)

[16] Pagano, D., Maalej, W.: User feedback in the appstore: An empirical study. In: 2013 21st IEEE International Requirements Engineering Conference (RE), pp. 125–134 (2013). IEEE

[17] Lim, S., Henriksson, A., Zdravkovic, J.: Data-driven requirements elicitation: A systematic literature review. SN Computer Science **2**(1), 16 (2021)

[18] Chen, N., Lin, J., Hoi, S.C.H., Xiao, X., Zhang, B.: Ar-miner: mining informative reviews for developers from mobile app marketplace. Proceedings of the 36th International Conference on Software Engineering (2014) https://doi.org/10.1145/2568225.2568263

[19] Guzman, E., Alkadhi, R., Seyff, N.: An exploratory study of twitter messages about software applications. Requirements Engineering **22**(3), 387–412 (2017)

[20] Tizard, J., Wang, H., Yohannes, L., Blincoe, K.: Can a conversation paint a picture? mining requirements in software forums. In: IEEE 27th International Requirements Engineering Conference (RE), pp. 17–27 (2019). IEEE

[21] Ciurumelea, A., Schaufelbühl, A., Panichella, S., Gall, H.C.: Analyzing reviews and code of mobile apps for better release planning. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 91–102 (2017). https://doi.org/10.1109/SANER.2017.7884612

[22] Talele, P., Phalnikar, R.: Classification and prioritization of software requirements using machine learning – a systematic review. In: 2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence), pp. 912–918 (2021). https://doi.org/10.1109/Confluence51648.2021.9377190

[23] Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the automatic classification of app reviews. Requirements Engineering **21**(3), 311–331 (2016) https://doi.org/10.1007/s00766-016-0251-9

[24] Oyebode, O., Alqahtani, F., Orji, R.: Using machine learning and thematic analysis methods to evaluate mental health apps based on user reviews. IEEE Access **8**, 111141–111158 (2020) https://doi.org/10.1109/ACCESS.2020.3002176

[25] Jarzębowicz, A., Sitko, N.: Agile requirements prioritization in practice: Results of an industrial survey. Procedia Computer Science **176**, 3446–3455 (2020)

[26] Mobile Accessibility Guidelines - Accessibility for Products - BBC. https://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile. Accessed: October 2022. https://www.bbc.co.uk/guidelines/futuremedia/accessibility/mobile Accessed 2023-06-19

[27] Eler, M.M., Orlandin, L., Oliveira, A.D.A.: Do android app users care about accessibility? Proceedings of the 18th Brazilian Symposium on Human Factors in Computing Systems (2019) https://doi.org/10.1145/3357155.3358477

[28] Wang, A.: Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461 (2018)

[29] Rajpurkar, P.: Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)

[30] Rahali, A., Akhloufi, M.A.: End-to-end transformer-based models in textual-based nlp. AI **4**(1), 54–110 (2023)

[31] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R.S., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., Kurzweil, R.: Universal sentence encoder. arXiv preprint arXiv:1803.11175 (2018)

[32] Devine, P., Koh, Y.S., Blincoe, K.: Evaluating unsupervised text embeddings on software user feedback. In: 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), pp. 87–95 (2021). IEEE

[33] Tizard, J., Devine, P., Wang, H., Blincoe, K.: A software requirements ecosystem: Linking forum, issue tracker, and faqs for requirements management. IEEE Transactions on Software Engineering **49**(4), 2381–2393 (2022)

[34] Mekala, R.R., Irfan, A., Groen, E.C., Porter, A., Lindvall, M.: Classifying user requirements from online feedback in small dataset environments using deep learning. In: 2021 IEEE 29th International Requirements Engineering Conference (RE), pp. 139–149 (2021). IEEE

[35] Sami, M.A., Rasheed, Z., Waseem, M., Zhang, Z., Herda, T., Abrahamsson, P.: Prioritizing software requirements using large language models. arXiv preprint arXiv:2405.01564 (2024)

[36] Wonnacott, T.H., Wonnacott, R.J.: Introductory Statistics. John Wiley Sons Incorporated, New York (1990)

[37] Krippendorff, K.: Content Analysis: An Introduction to Its Methodology. Sage Publications, Thousand Oaks, California (2018)

[38] Elo, S., Kyngäs, H.: The qualitative content analysis process. Journal of advanced nursing **62**(1), 107–115 (2008)

[39] Neuendorf, K.A.: The Content Analysis Guidebook. Sage, Thousand Oaks, California (2017)

[40] Dalpiaz, F., Dell'Anna, D., Aydemir, F.B., Çevikol, S.: Requirements classification with interpretable machine learning and dependency parsing. In: 2019 IEEE 27th

International Requirements Engineering Conference (RE), pp. 142–152 (2019). IEEE

[41] Alhoshan, W., Ferrari, A., Zhao, L.: Zero-shot learning for requirements classification: An exploratory study. Information and Software Technology **159**, 107202 (2023)

[42] ReCal2: Reliability for 2 Coders – Deen Freelon, Ph.D. http://dfreelon.org/utils/recalfront/recal2/. Accessed: December 2022. http://dfreelon.org/utils/recalfront/recal2/

[43] Arcuri, A., Fraser, G.: Parameter tuning or default values? an empirical investigation in search-based software engineering. Empirical Software Engineering **18**, 594–623 (2013) https://doi.org/10.1007/s10664-013-9249-9

[44] Mahmud, O., Niloy, N.T., Rahman, A., Siddik, M.S.: Predicting an effective android application release based on user reviews and ratings (2019) https://doi.org/10.1109/icscc.2019.8843677

[45] Keertipati, S., Savarimuthu, B.T.R., Licorish, S.A.: Approaches for prioritizing feature improvements extracted from app reviews. Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering - EASE '16 (2016) https://doi.org/10.1145/2915970.2916003

[46] Di Sorbo, A., Grano, G., Aaron Visaggio, C., Panichella, S.: Investigating the criticality of user-reported issues through their relations with app rating. Journal of Software Evolution and Process **33**, 2316 (2021) https://doi.org/10.1002/smr.2316

[47] Ni, J., Abrego, G.H., Constant, N., Ma, J., Hall, K.B., Cer, D., Yang, Y.: Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. arXiv preprint arXiv:2108.08877 (2021)

[48] Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992. Association for Computational Linguistics, Hong Kong (2019)

[49] Ayata, D.: Applying Machine Learning and Natural Language Processing Techniques to Twitter Sentiment Classification for Turkish and English

[50] Maimon, O., Rokach, L.: Data Mining and Knowledge Discovery Handbook vol. 2. Springer, New York (2005)

[51] Fortmann-Roe, S.: Understanding the bias-variance tradeoff. 2012. URL: http://scott. fortmann-roe. com/docs/BiasVariance. html (visited on

12/12/2017) (2015)

[52] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)

[53] Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A.: A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927 (2024)

[54] Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al.: Openai o1 system card. arXiv preprint arXiv:2412.16720 (2024)

[55] Developers, S.-l.: Classification Metrics — Scikit-learn Documentation. Accessed: 2025-02-15 (2024). https://scikit-learn.org/stable/modules/model_evaluation.html

[56] Shahhosseini, M., Hu, G., Pham, H.: Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. Machine Learning with Applications **7**, 100251 (2022)

[57] Tizard, J., Rietz, T., Liu, X., Blincoe, K.: Voice of the users: an extended study of software feedback engagement. Requirements Engineering **27**(3), 293–315 (2022)

[58] Tizard, J., Rietz, T., Blincoe, K.: Voice of the users: A demographic study of software feedback behaviour. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 55–65 (2020). IEEE

[59] Organization, W.H., *et al.*: Global Report on Health Equity for Persons with Disabilities. World Health Organization, Geneva (2022)

[60] Tizard, J., Rietz, T., Blincoe, K.: In: Damian, D., Blincoe, K., Ford, D., Serebrenik, A., Masood, Z. (eds.) Elicitation Revisited for More Inclusive Requirements Engineering, pp. 91–104. Apress, Berkeley, CA (2024). https://doi.org/10.1007/978-1-4842-9651-6_6