inventions

MDPI

Article

Data-Driven Requirements Prioritization Framework for App Reviews

Fatma A. Mihany [1,*], Galal H. Galal-Edeen [2,3], Ehab E. Hassanein [2] and Hanan Moussa [2,3]

[1] Information Systems Department, Faculty of Computers and Artificial Intelligence, Damietta University, Damietta 34511, Egypt
[2] Information Systems Department, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo 12613, Egypt; galaledeen@aucegypt.edu (G.H.G.-E.); e.ezat@fci-cu.edu.eg (E.E.H.); h.moussa@fci-cu.edu.eg (H.M.)
[3] Heikal Department of Management, Onsi Sawiris School of Business, American University in Cairo, Cairo 11835, Egypt
* Correspondence: f.mihany@du.edu.eg

Abstract

The rapid expansion of market-driven software product development has led to the increasing use of User-Generated Content (UGC), such as mobile application user reviews, as a valuable source of requirements. However, unlike the traditional requirements engineering (RE) process, data-driven RE introduces several challenges, particularly in requirements elicitation and prioritization. Traditional requirements prioritization techniques typically rely on stakeholders' involvement; however, in data-driven and market-driven development contexts, explicit stakeholders are often absent. Thus, we propose a DAta-driven Requirements Prioritization (DARP) framework that integrates Natural Language Processing (NLP), topic modeling, and Large Language Models (LLMs) to automate requirements prioritization in a data-driven development context. The proposed framework utilizes BERTopic to identify latent topics in user reviews and incorporates Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) to group semantically related requirements. The proposed framework introduces a robust and automated prioritization applied to mobile app reviews. The scope of the proposed framework is user-perspective prioritization. Our objective is to detect insights from app reviews to reflect the voice of the customer. The results indicate that leveraging NLP and topic modeling techniques provides an effective data-driven approach to requirements prioritization.

Keywords: prioritization; app reviews; data-driven software development; topic modeling; BERTopic

Academic Editor: Chung-Der Hsiao

Received: 15 February 2026
Revised: 19 March 2026
Accepted: 27 March 2026
Published: 31 March 2026

1. Introduction

Recently, there have been numerous software products developed for open marketplaces rather than for individual clients. This approach is called "Market-Driven Software Development". Eliciting requirements for a market-driven software product differs from traditional customer-driven software development. In a market-driven software development context, there are diverse sources of requirements, such as social media channels and mobile app platforms (e.g., App Store and Google Play) [1]. App Store research has gained attention across various domains, including requirements engineering, release planning, software design, and mobile application development. The App Store can be considered a source of valuable data (technical and non-technical), such as user reviews, rankings, and

ratings [2]. Developers can monitor the market landscape through the App Store, including customer behaviors and competitor activity, allowing them to proactively identify new opportunities and mitigate potential challenges [3].

Requirements Engineering (RE) is a process that is concerned with collecting, defining, organizing, documenting, and prioritizing requirements [4]. Market-Driven Requirements Engineering (MDRE) is a branch of RE that focuses on the elicitation and management of requirements for software products offered to a broad and open market.

In market-driven software development, release planning is crucial. The goal of release planning is to specify the optimal group of requirements to be implemented for a particular release [5]. Choosing which requirements are highly prioritized to be implemented for a certain release is not a trivial task, as it affects the overall software product's success. Release planning balances stakeholders' priorities against different constraints. Its outcome is a list of features or requirements for each release, considering resource availability and stakeholders' desires [6].

Requirements prioritization (RP) is essential for effective and successful software products across single or multiple releases. This process considers several factors such as a feature's importance, associated risks, costs, and more. In traditional software product development, decisions are made by various stakeholders, such as users, managers, and developers. In the requirements prioritization process, there is a set of alternatives that needs further prioritization and filtering [7]. Requirements Prioritization (RP) can be defined as a decision-making process that determines the order of implementing the software requirements according to several criteria [7]. Thus, RP is characterized as a multi-criteria decision-making process [8].

Traditional requirements prioritization techniques are highly dependent on input from the stakeholders, which often leads to subjectivity, inefficiency, and problems with scalability [7]. Stakeholders' involvement in the RP process is inadequate for data-driven software development contexts, since stakeholders are not fully known, and user needs and feature requests may be extracted only from their reviews. The objective of this work is to propose a prioritization framework that reflects the voice of the customer, which can complement stakeholder-driven prioritization criteria [8].

Text mining is the practice of analyzing textual data to uncover relevant information and produce meaningful insights for a specific purpose. Natural Language Processing (NLP) provides the tools and methods to process, interpret, and understand the structure of natural language, which are needed by text mining. Topic Modeling is considered a text mining technique that uncovers latent topics across multiple documents, in which a single document may include multiple topics [9,10]. In general, topic modeling can be used to assign certain topics to the documents. As digital textual data became widely available across platforms, such as social media and app stores, manual analysis of such data has become time consuming [11]. Topic Modeling facilitates extracting hidden patterns and topics automatically within a large volume of data [12]. Topic Modeling has been applied across a wide range of applications [13], especially in software engineering research [10]. Compared to other techniques, topic modeling facilitates the analysis of long textual content by grouping it into semantically coherent topics instead of individual words.

The topic modeling process begins with a document corpus as input, resulting in a set of extracted topics as output. The term frequency matrix M (Equation (1)) serves as input to topic modeling algorithms, such as LDA, which analyze word frequencies to uncover topics [9].

$$M = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \quad (1)$$

where:

$A_{i,j}$ is the weight (e.g. term frequency) of word w_i in document d_j. Rows correspond to documents, and columns correspond to words.

There are various techniques for topic modeling, ranging from probabilistic to embedding-based approaches, to extract hidden topics from large corpora. Some of the existing techniques will be presented in the following sub-sections.

Bidirectional Encoder Representations from Transformers and Topic Modeling (BERTopic) is an embedding-based model that maintains semantic meaning and the original text structure [9,14]. BERTopic can use any sentence-level-based model for generating dimensional vector text embeddings, such as BERT [15], ROBERTa, MPNET [16], and others. BERTopic generates document-level embeddings using a sentence-transformer model [15,16]. Text vectorization is an essential step in generating embeddings, which maps words, sentences, and documents into a vector space. The generated vectors may be sparse (contain zeros), so a dimensionality reduction technique may be needed. Dimensionality reduction is the process of reducing the dimensions of the generated data vectors while preserving the same positions and relationships of data points. BERTopic includes three parts: sentence-level embedding, dimensionality reduction, and a clustering technique [14].

Uniform Manifold Approximation and Projection (UMAP) is a popular technique for dimensionality reduction that can be used to reduce the dimensionality of the embeddings. Density-based clustering techniques, such as Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN), group data points based on high-density areas in the embedding vector space. In this space, the distance between vector spaces reflects the semantic similarity between textual data points. Semantically related requirements are positioned closer to each other, forming dense areas in the vector space. HDBSCAN identifies these dense areas, and the nearby data points are grouped into clusters. As a result, each cluster represents a coherent requirement topic, as the grouped requirements share similar semantic meanings [17,18]. The HDBSCAN follows the process below [14,18]:

1. Compute the core distance of each data point;
2. Compute the Mutual Reachability Distance metric;
3. Construct the Minimum Spanning Tree (MST);
4. Build a hierarchical tree based on MST;
5. Use the minimum cluster size parameter to condense the hierarchy;
6. Extract the final clusters.

The following research questions guide this research in addressing the identified research gaps:

1. What are the available software requirements prioritization techniques?
2. How can AI-based techniques, such as NLP, topic modeling, and Large Language Models, be leveraged to automate the RP process in a data-driven software development context?

The remainder of this paper is structured as follows: Section 2 presents the theoretical background of the materials and methods, discusses the related work on software requirements prioritization, introduces the proposed framework, and explains the materials and methods used in this research. Section 3 reports the experimental results. Section 4 discusses our findings and their elaborations. Finally, Section 5 concludes this paper and outlines directions for future work.

In this research, we propose an automatic approach to requirements prioritization for app reviews by integrating NLP, topic modeling, and Large Language Models (LLMs). The proposed framework utilizes topic modeling to automatically detect the topics of

the requirements, thereby supporting an effective requirements prioritization process. BERTopic, a well-known embeddings-based topic modeling technique, is utilized in the proposed framework [14].

2. Materials and Methods

This section reviews the theoretical background for the topic modeling addressed in this paper. In addition, an overview of requirements prioritization techniques will be presented.

2.1. Related Work

This section describes the various techniques proposed in the literature for software requirements prioritization.

There are several traditional prioritization techniques. Pair-wise comparisons-based techniques achieve better results, as they compare each requirement with the remaining ones to specify the corresponding importance. Otherwise, weight-based approaches focus on assigning weights and scores to requirements [19]. A group of well-known prioritization techniques with their related citations is presented in Table 1, which answers RQ1.

Analytical Hierarchical Process (AHP): This mainly depends on pair-wise comparisons in a matrix format to compute weighted scores for all requirements. AHP is the most popular prioritization technique. Moreover, it achieves reliable results.

Fuzzy AHP: It utilizes weighted scores for prioritizing the requirements according to a specific criterion.

Numerical Assignment Technique (NAT): It is based on categorizing requirements into different groups (high, medium, and low).

Cost-Value Approach (CVA): The basis of this approach is gained from AHP. It uses graph plots to visualize the requirements' importance against their cost of implementation, so that the top requirements can be identified easily.

Cumulative Voting ($100 Test): This is used to identify the top requirements by allocating a hypothetical $100 across them based on importance criteria. Once the allocation is completed, the requirements are ranked in ascending order according to the total number of dollars collected.

EVOLVE Technique: It iteratively applies machine learning-based optimization methods to maximize aggregated stakeholder-weighted prioritization scores.

Bubble Sort, Binary Search Tree, Minimal Spanning Tree Matrix, MoSCoW, win–win, priority groups, planning game, top-ten, and quality functional deployment face significant challenges in calculating the relative priority among requirements [20,21].

In [7], a fully automated framework was developed, which consists of three stages: preprocessing, clustering, and prioritization. Machine Learning (ML) techniques were used to process the requirements written in natural language. The framework leveraged BERT to understand the semantic meaning. In addition, the authors used the K-means clustering method for clustering the requirements with similar characteristics into groups.

Table 1. Traditional requirements prioritization techniques.

No.	Technique	Reference
1	Analytical Hierarchical Process (AHP)	[3,7,8,19,20,22–28]
2	Fuzzy AHP	[29]
3	Numerical Assignment Technique (NAT)	[24,30–32]
4	Cost-Value Approach (CVA)	[29,32]
5	100$ Test	[20]
6	Bubble Sort	[20,24]

Table 1. *Cont.*

No.	Technique	Reference
7	Win–Win	[20,33]
8	Planning Game	[31,34]
9	Case-Based Ranking	[34,35]
10	Kano Model	[36]
11	MoSCoW	[20]
12	Binary Search Tree	[20]
13	Minimal Spanning Tree Matrix	[37]

This section addresses the second research question (RQ2) by reviewing the recent studies that have applied AI and LLMs in the prioritization process, presenting promising achievements. The effectiveness of AI-based prioritization relies on key factors, including stakeholder collaboration, scalability, automation, and accuracy [37].

In Ref. [3], the authors proposed a data-driven framework for the task of user requirements prioritization. They focused on requirements from video conferencing apps (Zoom, Google Meet, Microsoft Teams). The framework consists of three phases: data collection, embedding, and prioritization. Their proposed framework integrates sentiment analysis, topic modeling, network analysis, machine learning, and Explainable Artificial Intelligence (XAI) for prioritizing user requirements. Sentiment analysis was applied using the SentiStength tool to classify the user review into one of three classes: positive, negative, or neutral. Structural Topic Modeling (STM) was used to distribute negative reviews into one of eight predefined topics. Network analysis was applied to extract relationships between topics using the Gephi tool. The framework leveraged XAI to identify weak points in these apps (e.g., security issues, audio/video quality). Negative reviews were first identified, and topic modeling was applied to detect key topics in the requirements. Network analysis was utilized to explore the relationships between the identified topics. A set of regression techniques, including Gradient Boosting Machine (GBM), were utilized to map topic frequency to the ratings.

In Ref. [38], the authors introduced a web-based tool for enhancing the requirements prioritization process by utilizing Large Language Models (LLMs). The proposed tool combines React for the front end, Flask for the back end, and OpenAI's APIs to leverage LLMs, such as GPT 3.5, to automatically extract and prioritize software requirements. Their proposed tool receives input data from the user, and then it generates user stories. After that, the developed tool offers various prioritization approaches, including AHP, MoSCoW, and others. Finally, their developed tool offers prioritized requirements and user stories for downloading in different file formats. Their developed tool can be easily integrated with project management tools, including JIRA, Trello, and Azure DevOps.

The authors in [26] presented an automated approach called "ReFeed" that supports the software requirements prioritization process using user feedback. Mainly, ReFeed resolves the semantic gap between developers and the vocabulary of users. ReFeed leverages Natural Language Processing (NLP) approaches to process feedback and input and extract quantifiable properties (such as sentiment and severity). ReFeed integrates sentiment analysis, speech act analysis, and severity metrics to quantify the feedback. ReFeed uses a domain ontology to map feedback into requirements. Moreover, the Jaccard similarity was computed between the feedback and the requirements.

In Ref. [39], an NLP-based approach was introduced to reduce manual processing in the Agile release planning process. The approach automatically clusters user stories into suitable releases. The approach starts by collecting user stories, then breaks down the project into modules for each release, generating a specific word corpus that represents a

relevant theme. The top three most frequent words were selected for each requirement. An NLP algorithm called the RV coefficient was applied to measure the correlation between each release corpus and the embedding vector. The RV coefficient is a statistical measure that evaluates the similarity between two multivariate sets of data. The output was a list of clustered user stories distributed over releases. Their proposed approach achieved 82.35% accuracy, and the planning time was reduced by 78.57%. A limitation of their study is the lack of detailed methodological and algorithmic descriptions.

In Ref. [40], an NLP-based approach called "NLP4IP" was introduced that leverages ML techniques in the prioritization process. It is a semi-automated method that predicts the rank of new or modified issues by combining the textual features (e.g., title and description) and attributes defined by stakeholders (e.g., priority). The authors claimed that the NLP4IP approach resolves the limitation of existing prioritization methods by offering a production-ready tool, which responds to newly added issues dynamically. The study evaluated the approach on some projects from JIRA repositories and achieved an accuracy of 81%.

In Ref. [41], the authors proposed a requirements prioritization method that employs Artificial Intelligence to support prioritizing the functional and non-functional prioritization process. Traditional prioritization techniques have scalability and a lack of automation limitations, especially with a large volume of data. That research addresses these limitations by applying machine learning (ML) and Deep Learning (DL) techniques, including Random Forest (RF). The approach reduces the involvement of stakeholders by focusing on applying a data-driven methodology but still needs their validation.

2.2. Research Gaps and Motivation

Despite the growing interest in applying Artificial Intelligence and NLP techniques to requirements prioritization and release planning, significant gaps persist. There is still a shortage of research that addresses requirements prioritization within a data-driven software development context. Many studies, which have focused on addressing requirements prioritization in the Agile software development lifecycle model and applied to stakeholder-driven projects, have mainly relied on stakeholders' rankings. Such approaches have limitations, including subjectivity that may lead to biased prioritization decisions. This motivated our proposed framework, which leverages topic modeling to support the automation of the requirements prioritization process, especially within the data-driven software development context. The proposed framework reflects user-perspective prioritization derived from app reviews.

2.3. Proposed Framework

This section outlines the proposed framework, which is organized as a set of sequential phases. This research proposes a DAta-driven Requirements Prioritization (DARP) framework that aims to automate the requirements prioritization process by leveraging Natural Language Processing (NLP), topic modeling techniques, and Large Language Models (LLMs). The proposed framework transforms a large volume of user app reviews collected from market-driven software products into a prioritized set of software requirements.

The DARP framework formulates the prioritization task as a ranking problem and incorporates topic modeling to support the ranking process with extracted latent themes. DARP works upon the output of our previously published AREAR framework, which performs initial collection and classification of the app reviews [42]. The DARP framework extends and complements the earlier work, introducing an integrated and comprehensive pipeline for market-driven software product release planning.

The DARP framework consists of three main phases: Data Preparation, topic modeling, and requirements prioritization. An overview of the proposed framework is illustrated in Figure 1, and a detailed description of each phase is presented in the following sub-sections.

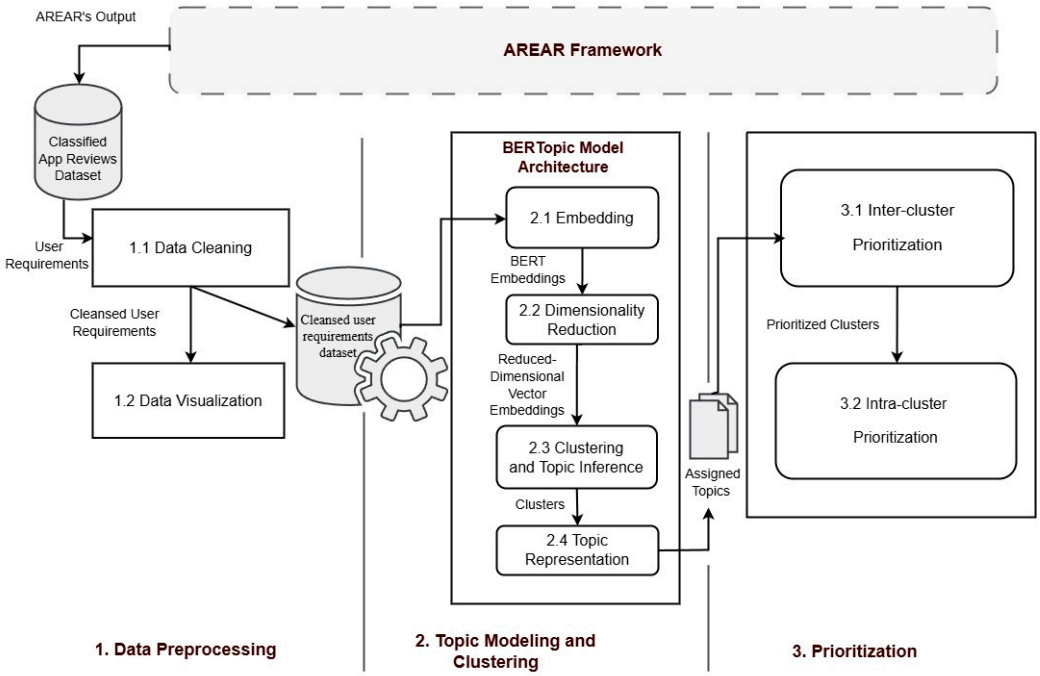

Figure 1. DAta-driven Requirement Prioritization (DARP) Framework.

2.3.1. Data Preprocessing

This phase focuses on preparing the data for subsequent stages of the proposed DARP framework. The classified feature requests of a software product were gathered by applying our previously published AREAR framework [42]. The AREAR framework focuses on classifying app reviews into predefined classes. The AREAR framework automates the requirements elicitation process, supporting the development of market-driven software products. So, only the reviews that are classified as feature requests will be retained, which are further rechecked to ensure consistency. The data preprocessing phase includes two sub-phases: Data Cleaning and Data Visualization. These sub-phases support the formalization and structuring prior to embedding extraction and topic modeling.

Data Cleaning

This sub-phase mainly rechecks the quality of feature requests obtained from the Data Gathering sub-phase. The processing is not extensive, as all the collected reviews were earlier checked in our previously published AREAR framework [42]. Nevertheless, the reviews are rechecked to handle any existing noise, including emoticons, jargon expressions, embedded links, and spelling errors, to ensure the quality and consistency of the data.

Data Visualization

This sub-phase employs data visualization techniques, including distribution charts, word frequency plots, and relationship analysis (e.g., heatmaps), to visually understand and explore data patterns and insights before proceeding to subsequent phases of the proposed framework.

2.3.2. Topic Modeling and Clustering

Topic Modeling aims to identify the latent topics to semantically group similar documents together. The main objective of this phase is to automatically detect the main topics of

the requirements and organize and associate them with their relevant topics. Topic Modeling provides the structure for systematic prioritization and informative decision-making for the next phase of the framework. We employed Bidirectional Encoder Representations from Transformers and Topic Modeling (BERTopic), a well-established topic modeling approach that combines language models with clustering. BERTopic is an embedding-based model; it leverages semantic embedding models, similar to BERT, to maintain the original semantic structure [43].

Embedding

BERTopic is an embedding-based model that contextualizes embeddings to structurally represent the input text (review text) in a continuous vector space. Each requirement is first represented using a pretrained language model, like BERT, to extract the embeddings, maintaining the semantic meaning [15]. Unlike traditional topic modeling techniques that use a bag-of-words representation, recent embedding-based techniques explore semantic meaning with contextual relationships.

Dimensionality Reduction

The input for this sub-phase is the embeddings generated by an embedding-based model, as explained in the previous sub-phase. In BERTopic, dimensionality reduction is essential, as it reduces the high dimensionality of the generated embedding of the requirements. As high-dimensional vectors may contain noise and cause computational complexity. Moreover, it may have a negative effect on clustering performance. UMAP is used as a dimensionality reduction technique while preserving the same relationships and positions.

Clustering and Topic Inference

HDBSCAN is applied to cluster the embeddings after applying UMAP for dimensionality reduction. The HDBSCAN technique is employed as a clustering technique to semantically group requirements that share similar semantic meaning. Unlike traditional clustering techniques such as k-means, HDBSCAN can explicitly identify noisy data points. A noisy data point refers to a requirement that does not belong to any stable high-density area in the embedding space. Traditional clustering techniques assign every data point to the closest cluster; however, HDBSCAN keeps such data points unassigned. This helps maintain cluster coherence by preventing unrelated data points from being assigned to inappropriate clusters. The identified noisy data points can be analyzed separately without distorting the coherent clusters. The expected output of this phase is a set of coherent clusters that contains semantically related requirements [17,18].

When a new requirement is received, BERTopic performs topic inference to assign it to the most relevant topic. The new incoming requirement is first converted into an embedding. Next, the embedding is projected into the reduced vector space using UMAP. Finally, the transformed vector is assigned to the semantically closest cluster using HDBSCAN; otherwise, it may be labeled as noise.

Topic Representation

The main objective of this sub-phase is to assign a topic label for each cluster. A topic representation model is integrated within the BERTopic framework. Every cluster is characterized by a topic, representing a semantic overview of the grouped requirements. For each cluster, a set of representative keywords is identified according to the semantic similarity within the embedding space. Since user reviews are relatively short texts, an embedding-driven representation is more suitable [43].

2.3.3. Requirements Prioritization

The final phase of the proposed framework addresses the main objective of this research, which is the prioritization of the requirements. In the context of market-driven software product development, where there is no single predefined customer, traditional requirements prioritization techniques are inadequate. Accordingly, this phase embraces a data-driven methodology that employs semantic similarity to explore the importance of requirements.

After organizing the requirements into topic-oriented clusters, semantic embeddings of requirements are employed to aid in prioritization. Sentence-level embeddings extracted during the previous phase are reused. There is no need to re-extract embeddings in this phase.

The proposed framework performs the prioritization on two distinct levels: Inter-cluster and intra-cluster prioritization, which will be explained in the following sub-sections.

Inter-Cluster Prioritization

Inter-cluster prioritization focuses on assessing the relative importance of different requirement clusters. Following the previous phase (topic modeling and clustering), each cluster contains a set of semantically related requirements. Prioritizing these clusters helps identify which clusters are most important from the user's perspective.

In addition, stakeholder involvement in the RP process is often inadequate in a market-driven development context. Prior studies reveal that utilizing UGC, such as user app reviews and feedback, enables capturing the voice of the customer. The extracted information can help uncover valuable insights from the perspective of the user, such as user requests, levels of dissatisfaction, preferences, and other relevant information that can support the prioritization process [1,25].

Each cluster is assigned a score equal to the number of requirements it contains. This score reflects how frequently users request features related to that cluster and therefore serves as an indicator of its relevant importance. This score reflects the collective demand for features shared across many users, enabling data-driven requirements prioritization. This approach enables inter-cluster prioritization to be applied in a simple and objective manner, reducing dependence on traditional stakeholder ranking methods. The resulting cluster score is then used in the prioritization across all clusters by ordering them in descending order based on their computed scores. This score reflects the relative level of user demand associated with each cluster, as elaborated in Algorithm 1 below.

The formulation uses requirements frequency as an indicator for cluster importance. The cluster containing a larger number of requirements represents features that are requested by a large number of users. The following formula of cluster importance takes into consideration the demand criterion represented in the number of user requirements (Equation (2)).

$$F(C_i) = |C_i| \tag{2}$$

where:

C_i is the cluster number i;

$|C_i|$ is the number of requirements in cluster i;

$F(C_i)$ represents cluster importance.

Then, the priority score for each cluster is computed, as shown in Equation (3):

$$\text{Priority\_score}(C_i) = \text{Rank}_{\text{desc}}(F(C_i)) \tag{3}$$

where:

$\text{Priority\_score}(C_i)$ represents the priority score for cluster C_i;

$\text{Rank}_{\text{desc}}$ assigns the ranking score after sorting clusters in descending order based on the computed frequencies.

Algorithm 1: Inter-Cluster Prioritization

Input: A set of clusters $C = \{C_1, C_2, C_3, \ldots, C_n\}$
Output: Priority score (rank) for each cluster
Steps:
1. For each cluster $C_i \epsilon C$, compute the frequency of the contained requirements according to Equation (1).
2. Sort all clusters in descending order according to $F(C_i)$.
3. Assign the priority score based on the sorted order:
   The first cluster takes rank 1 (highest priority)
   The second takes the value of 2, and so on
4. Return the ranked list

Intra-Cluster Prioritization

Intra-cluster prioritization focuses on prioritizing individual user requirements within each cluster. Since the grouped requirements within each cluster share a common semantic topic, each requirement is considered a separate data point represented by its embedding vector. For each cluster, a centroid is determined by aggregating the embeddings of all associated requirements, capturing the center that reflects the most relevant semantic requirement to the cluster's main topic. The cluster centroid can be considered a surrogate representation, providing a useful guide for stakeholders on where to begin.

Based on the identified cluster centroid, requirements are prioritized according to their semantic relevance to the centroid, which represents its corresponding cluster. Since all user requirements within the same cluster belong to similar topics for cluster x, a centroid is computed using the following equation:

$$C_x = \frac{1}{|D_x|} \sum_{i \ \epsilon \ D_x} X_i \tag{4}$$

where:

C_x is the centroid for cluster x;
D_x is the set of requirements assigned to cluster x;
$|D_x|$ number of requirements in the cluster;
X_i is the embedding vector of requirements i;

Then a semantic similarity measure is computed, such as cosine similarity, using the following equation:

$$sim_i = \cos(x_i, c) = \frac{x_i \,.\, c}{\|x_i\| \|c\|} \tag{5}$$

where:

sim_i is the cosine similarity of requirement i to the centroid;
x_i is the embedding vector of requirements i;
c cluster centroid vector;
$\|c\|$ length of the embedding vector c;
$\|x_i\|$ length of the embedding vector x_i.

Finally, all requirements are ordered in descending order according to the computed similarities to the cluster centroid, as elaborated in Algorithm 2 below.

Algorithm 2: Intra-Cluster Prioritization

Input: A list of embedding vectors $V_i \epsilon C_i$, $V_j = \{v_1, v_2, \ldots, v_n\}$ corresponding to the contained requirements $R = \{r_1, r_2, \ldots, r_n\}$
Output: A list of prioritized requirements R
Steps:
1. Compute the cluster centroid
   For each cluster $pc \epsilon PC$, identify the centroid requirement
2. Compute the cosine similarity to the centroid
3. Order the requirements in descending order according to the computed similarities to produce the P list
4. Return the list of Ps

The following section describes the technical implementation details, dataset characteristics, and evaluation metrics.

The following sub-section describes the dataset, development details, and methodological architecture of the proposed DARP framework.

2.4. Dataset Description

The dataset used in this paper is part of a dataset used in our previous work [42]. The dataset contains app reviews of 39 mobile applications from the Google Play Store. The dataset was originally collected by Sebastiano and Ciurumelea [44] in the time period June–July 2016. The dataset has been reused in prior research [1,45]. The original dataset covers 17 different categories, such as instant messaging, social network, games, education, science, reading, communication, and more [45]. Additional details about each app and the collected reviews are available in an online repository [44]. The dataset used in this research represents the output of applying our published AREAR framework, which aimed at automatically eliciting requirements from app reviews [42]. Our study considers market-driven software development, with no predefined users or known stakeholders; the extracted feature requests represent the voice of the customer. Building on these extracted and elicited sets of users' feature requests, the next phase of the pipeline focuses on requirements prioritization (RP).

Basically, app reviews within the original dataset are labeled according to five predefined labels, including feature request, information seeking, information giving, problem discovery, and others [45]. A significant challenge was encountered during dataset preparation, which was the uneven distribution of reviews across applications. When reviews were filtered per every single application, the resulting subsets frequently contained a relatively limited number of reviews, thereby restricting the data-driven software development and topic modeling approach. To overcome this challenge, all reviews labeled as "feature request" and "information giving" were merged and aggregated into one pooled corpus. Table 2 presents the types of reviews along with their counts, reflecting the dataset after the filtering process. Table 3 summarizes the descriptive statistics of the review text, including mean, median, maximum, minimum, and standard deviation.

Table 2. App reviews statistics in the dataset [42,45].

Label	Description	Count
Feature Request	A user suggests a new feature	85
Information Giving	A user provides a suggestion or a "thank you" message	1043

Table 3. Descriptive statistics of the review length in the dataset.

Parameter	Value
Average	26.4
Median	21
Min	2
Max	165
Std	19.899

The proposed framework was applied to the whole dataset that consists of 1128 app reviews, with all records of app name, version, author, and review text. Figure 2 illustrates a word cloud representation of the dataset, in which word size corresponds to frequency. For instance, feature, game, and app appear in larger font sizes, indicating a higher frequency, whereas words like work, screen, and battery are displayed in smaller sizes, reflecting lower occurrence.

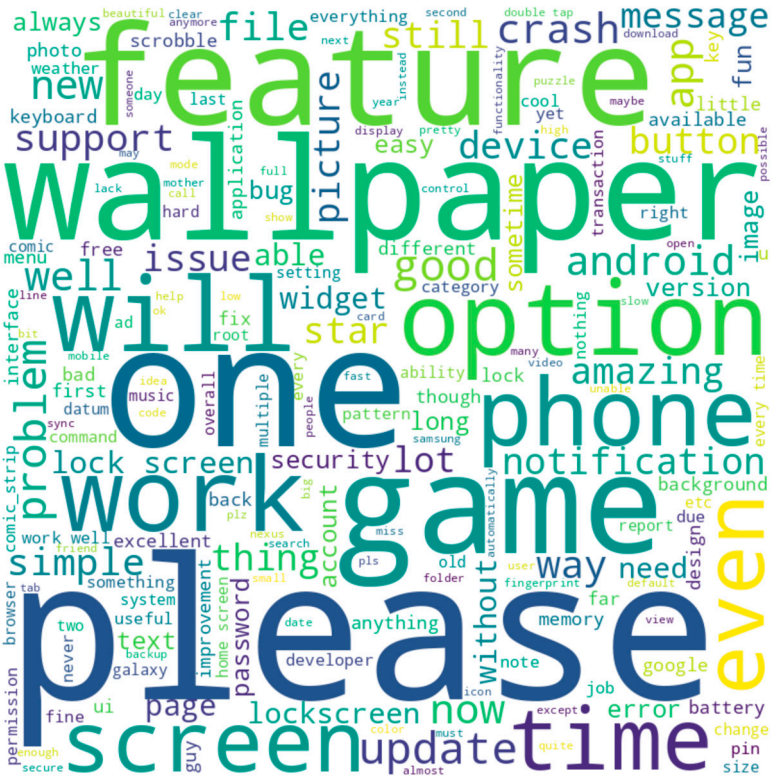

Figure 2. Word cloud of the dataset, where word size represents frequency, highlighting the most frequent words.

2.5. BERTopic Settings

All experiments were conducted using Python 3.12.12 (Python Software Foundation, Wilmington, DE, USA) on Google Collaboratory (Google LLC, Mountain View, CA, USA). The detailed environment settings are provided in Table 4. App reviews preprocessing was performed using functions from the NLTK Python library. BERTopic combines several steps, including embedding generation, dimensionality reduction, clustering, vectorization, topic extraction, topic representatives' generation, and topic inference. In this research, we utilized the "all-mpnet-base-v2" model to generate dense dimensional vector representations of the input app reviews. It is a pretrained sentence-level transformer-based model that relies on Microsoft's MPNET. This model was chosen due to its efficient performance in capturing strong semantic representations. In addition, it is suitable for short text, such

as comments, reviews, and requirements. It is well suited to various tasks, including sentence similarity, clustering, and information retrieval [16]. The "all-mpnet-base-v2" model takes the input of cleansed app reviews and generates a 768-dimensional sentence vector, capturing the semantic meaning.

Table 4. Parameter tuning.

Phase	Technique	Parameter
Topic Modeling	BERTopic	Version = 0.17.4 top_n_words = 10 min_topic_size = 10
Embedding	Sentence-Transformer-Based Embedding Model ("all-mpnet-base-v2")	
Dimensionality Reduction	UMAP	n_components = 5 n_neighbors = 15 min_dist = 0.0 similarity metric = cosine
Vectorization	CountVectorizer	min_df = 5
Clustering	HDBSCAN	min_cluster_size = 15 min_samples = 5 similarity metric = Euclidean
Topic Representation	keyBERT	keyBERTInspired

The UMAP technique was employed for dimensionality reduction, which involves several configural parameters. For visualization purposes, the min_components parameter is recommended to be two (for two-dimensional plots), but according to [46], it is not ideal for clustering. The embedding phase is performed using the "all-mpnet-base-v2" model, while each user requirement is transformed into a high-dimensional vector space (768-vector embeddings). These embeddings represent the semantic meaning of the input user requirements. UMAP reduces the dimensionality from 768 to 15 dimensions only, while preserving the same positions and relationships in the new vector space.

HDBSCAN is applied to map the semantically related requirements close to each other in the vector space when sentence embeddings (e.g., sentence-BERT embeddings) are generated. Traditional clustering techniques assign every data point to the closest cluster; HDBSCAN keeps such data points unassigned. This helps maintain cluster coherence by preventing unrelated data points from being assigned to inappropriate clusters [17,18].

2.6. Evaluation Metrics

Spearman's Rank (ρ):

Spearman's rank correlation coefficient (Equations (6)–(8)) is a non-parametric measure that evaluates the correlation between different variables based on their scored relationship [47,48]. It disregards the actual relation between the variables, as it measures how one variable increases or decreases while the second one increases [48].

$$d_i = x_i - y_i \tag{6}$$

$$d_i^2 = (x_i - y_i)^2 \tag{7}$$

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{8}$$

where:

- d_i is the difference between the scores;
- x_i and y_i are the different scores;
- n is the number of paired scores;
- $\sum_{i=1}^{n} d_i^2$ is the addition of squared differences.

3. Results

This section presents the experimental results obtained from applying the proposed DARP framework to the dataset described in Section 2.4, along with the validation of these results.

The prioritization across all clusters is referred to as inter-cluster prioritization. This process is performed by following the steps of the proposed DARP framework, as described in Section 2.3. The dataset presented earlier was used as input to the framework, resulting in 52 clusters.

The second column in Table 5 summarizes the prioritization scores for the top 20 clusters obtained by the proposed framework. The prioritization scores are ranked in ascending order, where a score of 1 indicates the highest priority, followed by 2, and so on. Topic modeling offers the advantage of generating a set of representative keywords for each cluster, thereby capturing its primary topic. Subsequently, Large Language Models (LLMs) are employed to transform these keywords into a representative label for each cluster, as shown in the column titled "Label" in Table 5.

Table 5. Inter-cluster prioritization validation results.

Label	DARP	Expert (1)	Expert (2)	Expert (3)	Expert (4)	Expert (5)
Usability/UI Refinement Requirements	1	1	13	1	1	1
Secure Messaging Features and Communication Functionality Requirements	2	2	10	7	2	2
Lock Screen Security and Customization Requirements	3	9	9	10	9	3
Wallpaper Content Management and Customization Requirements	4	17	14	19	18	15
Transaction Management, Reporting, and Financial Insights Requirements	5	12	15	2	13	4
Crash Issues	6	5	19	6	11	5
Gameplay Enhancement Requirements	7	18	20	14	17	16
File and Input Operation Failures	8	4	18	20	15	12
Uninstall/Removal Failure	9	19	11	12	16	17
Account Creation/Login Failure	10	5	17	8	3	6
OS/Device Compatibility and Support	11	6	21	16	12	7
Connectivity and Protocol Issues	12	11	1	15	10	13
Scrobbling/Playback Tracking Failure	13	20	16	11	14	18
Battery/Memory Consumption Requirements	14	21	2	22	4	19
Cloud and Remote Storage Integration	15	12	6	21	5	8
Data Sync and Update Failures	16	13	7	9	6	9
Weather Data/API Failure	17	7	5	13	19	20
Keyboard and Input Handling Issues	18	14	8	8	8	14
Notifications Visibility Requirements	19	8	12	5	7	10
Fingerprint Support Requirements	20	15	4	18	20	11

The prioritization of the requirements within each cluster is referred to as "Intra-cluster Prioritization". The second column in Table 6 presents the prioritization scores for the requirements in one cluster, titled "Lock Screen Security and Customization Requirements", as an illustrative example. This cluster comprises 24 requirements identified using the DARP framework. The prioritization scores are sorted in ascending order; a score of 1 represents a higher value than a score of 2, and so on.

Table 6. Intra-cluster prioritization validation.

Requirement ID	DARP Framework	Expert (1)	Expert (2)	Expert (3)	Expert (4)	Expert (5)
R 1	0	0	17	2	2	2
R 2	1	1	6	1	1	1
R 3	2	8	1	4	4	4
R 4	3	16	5	5	5	5
R 5	4	11	19	10	10	10
R 6	5	4	20	16	16	16
R 7	6	17	14	12	12	12
R 8	7	3	9	0	0	0
R 9	8	18	3	9	9	9
R 10	9	4	12	7	7	7
R 11	10	5	23	18	18	18
R 12	11	10	16	17	17	17
R 13	12	19	7	20	20	20
R 14	13	20	24	15	15	15
R 15	14	11	15	3	3	3
R 16	15	12	10	21	21	21
R 17	16	6	4	8	8	8
R 18	17	13	8	11	11	11
R 19	18	7	2	13	13	13
R 20	19	14	21	6	6	6
R 21	20	15	11	21	21	21
R 22	21	12	18	20	20	20
R 23	22	13	13	22	22	22

The validation process includes assessing the clustering and topic modeling output by seeking human expert opinion. In a traditional requirement clustering pipeline, the embedding vectors are generated, and then a clustering technique is applied. Examples of clustering techniques include HDBSCAN and K-means. The evaluation metrics are computed to assess clustering quality. The traditional clustering techniques provide similar groups of requirements without distinct topic labels. On the other side, the BERTopic pipeline includes embeddings, dimensionality reduction, HDBSCAN clustering, topic representation, and topic referencing. The output includes clusters, topic labels, and different visualization tools.

Figure 3 visualizes the inter-topic distance map obtained from BERTopic, which was applied in the topic modeling and clustering phase of the DARP framework. In the inter-topic distance map, a circle represents a separate discovered topic, and the distance between the circles represents the semantic dissimilarities among the topics. These dissimilarities were computed in the reduced dimensionality embedding space after applying UMAP as a dimensionality reduction technique. The number of requirements included in a certain topic is referred to as topic density in this context. It refers to how tightly topics are grouped in a two-dimensional space [49]. The topic density is reflected in the circle size, which means larger circles represent higher density.

Figure 3. Inter-topic distance map, where larger circles represent higher topic prevalence.

The proposed DARP framework is applied to generate priority scores at both the inter-cluster and intra-cluster levels. A qualitative validation process was then conducted, in which five domain experts independently prioritized the two lists for both levels. For each prioritization level, lower scores indicate higher priority, whereas higher scores correspond to lower priority.

We have compared the inter-cluster prioritization results by examining the priority scores of the top 20 clusters as generated by the DARP framework against those independently assessed by five domain experts, as presented in Table 5.

Table 6 summarizes the priority scores for intra-cluster prioritization of a single cluster as generated by the DARP framework and independently assessed by five domain experts.

Comparing the prioritization produced by the DARP framework and the experts' opinions, Spearman's rank correlation coefficient [47] was computed, as presented in Table 7. For inter-cluster prioritization, Expert 1 exhibited a positive moderate correlation (0.376), Expert 2 showed a negative correlation (−0.522), Expert 3 achieved a strong positive correlation (0.886), Expert 4 achieved a strong positive correlation (0.0.759), and Expert

5 achieved a strong positive correlation (0.156). In intra-cluster prioritization, Expert 1 showed a positive correlation of 0.656, Expert 2 at 0.099, Expert 3 at 0.601, Expert 4 at 0.676, and Expert 5 at 0.087.

Table 7. Spearman's correlation coefficient for inter-cluster and intra-cluster prioritization.

Expert	Inter-Cluster Prioritization	Intra-Cluster Prioritization
Expert 1	0.376	0.656
Expert 2	−0.522	0.099
Expert 3	0.886	0.601
Expert 4	0.759	0.676
Expert 5	0.156	0.087

4. Discussion

Figure 3 presents the inter-topic distance map, which shows a qualitative evaluation of topic relationships and their patterns. There is a clear separation between the extracted topics, which are represented by the distances between topics, reflecting BERTopic's ability to effectively capture distinct topics. Such separation is essential in the data-driven context, as it decreases semantic ambiguity among the extracted topics. This separation supports an enhanced inter-prioritization phase. As observed in Figure 2, there are some inter-connected circles that reflect semantically related topics, which is expected in data-driven software products, as multiple users may share related feature requests. This indicates that there is semantic overlap in the dataset. The relative enhancements indicate that density-based clustering more accurately represents and clusters the data patterns.

The Spearman's correlation coefficient varies the levels of correlation between the participating experts and the proposed DARP framework. For inter-cluster prioritization, strong positive correlations are observed for Expert 3 (ρ = 0.886) and Expert 4 (ρ = 0.759), suggesting that the proposed DARP framework closely matches these experts' prioritization direction. A moderate positive correlation with Expert 1 (ρ = 0.376) and Expert 5 (ρ = 0.156). These differences may be attributed to subjective factors in expert opinion, such as different interpretations or domain experience. In contrast, Expert 2 shows a negative correlation (ρ = −0.522), showing a variation in the prioritization scores. This negative relationship may have resulted from the expert considering additional criteria not captured by the frequency-based method, such as technical considerations. In addition, the human-based requirements prioritization process entails subjective and domain-dependent judgments by nature.

On the intra-cluster prioritization level, the results show a strong positive correlation with Expert 1 (ρ = 0.656), Expert 3 (ρ = 0.601), and Expert 4 (ρ = 0.676), indicating a good level of alignment between the proposed framework ranking and the experts' opinions.

The correlation with Expert 2's (ρ = 0.099) and Expert 5's (ρ = 0.087) prioritization scores indicate minimal alignment in the prioritization direction. This highlights the drawback of subjectivity in the human-based prioritization process.

5. Conclusions

There are various techniques for requirements prioritization (RP). Traditional prioritization techniques, including the Analytical Hierarchal Process (AHP), Numerical Assignment Technique (NAT), MoSCoW, and Binary Search Tree, have been widely used in the literature. AI-based techniques, including NLP and LLMs, resolve, to a significant extent, scalability and subjectivity challenges. According to recent studies [3,38,50], AI-based

prioritization techniques are more suitable for data-driven software development. AI and LLM techniques need further validation in real-world organizational settings.

This research proposes a Data-driven Requirements Prioritization (DARP) framework, which aims to automate the requirements prioritization process from the user perspective. The DARP framework leverages NLP techniques for data preprocessing and cleaning, as well as topic modeling techniques to extract hidden topics within user requirements. The extracted topics by the topic modeling techniques are then used to support the data-driven requirements prioritization process, thus minimizing manual effort and reducing subjectivity in the prioritization process, which presents a significant improvement in data-driven software product development. The proposed framework leverages topic modeling to support the automation of the requirements prioritization process, especially in the data-driven software development context. DARP framework prioritization scores are reproducible; in addition, they represent the voice of the customer and can save manual effort and time. The DARP framework ensures reproducibility through clearly described preprocessing steps, parameter settings, and methodological procedures. The prioritization scores computed by applying the DARP framework represent the voice of the customer and can save manual effort and time. The proposed framework is developed to complement existing prioritization frameworks, not to replace them.

Although the proposed DARP framework shows enhancements in the requirements prioritization process in the context of data-driven software product development, the research has some limitations. This research does not offer a comprehensive prioritization framework; instead, the scope of the proposed framework is user-perspective prioritization derived from User-Generated Content (UGC), especially mobile app reviews. In addition, the DARP framework was applied to one dataset thus far. A significant challenge was encountered during dataset preparation, which is the uneven distribution of reviews across applications. When reviews were filtered by each application, the resulting subsets frequently contained a relatively limited number of reviews, thereby restricting the full application of the data-driven software development and topic modeling approach. A limitation of this study is the relatively small number of experts involved in the evaluation.

In our future work, more datasets, including requirements from various domains, may be used. In addition, our framework may be applied to datasets of domain-specific applications, such as payment apps, gaming, and others. A larger number of human experts may be involved in the evaluation process, along with more comprehensive agreement analysis.

Author Contributions: Conceptualization, writing—review and editing, F.A.M., G.H.G.-E., E.E.H., and H.M.; methodology, investigation, resources, writing—original draft preparation, visualization; software, validation, formal analysis, F.A.M., G.H.G.-E. and H.M.; software, data curation, F.A.M.; supervision G.H.G.-E., E.E.H. and H.M.; project administration, G.H.G.-E. and H.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AHP	Analytical Hierarchical Process
AI	Artificial Intelligence
BERTopic	Bidirectional Encoder Representations from Transformers and Topic Modeling
DARP	DAta-driven Requirement Prioritization
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
LLMs	Large Language Models
NLP	Natural Language Processing
RE	Requirements Engineering
Std	Standard Deviation
UGC	User-Generated Content

References

1. Triantafyllou, I.; Drivas, I.C.; Giannakopoulos, G. How to utilize my app reviews? A novel topics extraction machine learning schema for strategic business purposes. *Entropy* **2020**, *22*, 1310. [CrossRef]
2. Dąbrowski, J.; Letier, E.; Perini, A.; Susi, A. Analysing app reviews for software engineering: A systematic literature review. *Empir. Softw. Eng.* **2022**, *27*, 43. Correction in *Empir. Softw. Eng.* **2022**, *27*, 58. https://doi.org/10.1007/s10664-022-10135-4. [CrossRef]
3. Bai, S.; Shi, S.; Han, C.; Yang, M.; Gupta, B.B.; Arya, V. Prioritizing user requirements for digital products using explainable artificial intelligence: A data-driven analysis on video conferencing apps. *Future Gener. Comput. Syst.* **2024**, *158*, 167–182. [CrossRef]
4. Crowder, J.A.; Hoff, C.W. MDRE: The Requirements Engineering Process. In *Requirements Engineering: Laying a Firm Foundation*; Springer: Berlin/Heidelberg, Germany, 2022.
5. Carlshamre, P. Release planning in market-driven software product development: Provoking an understanding. *Requir. Eng.* **2002**, *7*, 139–151. [CrossRef]
6. Ruhe, G.; Saliu, M.O. The Art and Science of Software Release Planning. *IEEE Softw.* **2005**, *22*, 47–53. [CrossRef]
7. Jamasb, B.; Khayami, S.R.; Akbari, R.; Taheri, R. An Automated Framework for Prioritizing Software Requirements. *Electronics* **2025**, *14*, 1220. [CrossRef]
8. Achimugu, P.; Selamat, A.; Ibrahim, R.; Mahrin, M.N.R. A systematic literature review of software requirements prioritization research. *Inf. Softw. Technol.* **2014**, *56*, 568–585. [CrossRef]
9. Grootendorst, M. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv* **2022**, arXiv:2203.05794. [CrossRef]
10. Silva, C.C.; Galster, M.; Gilson, F. Topic modeling in software engineering research. *Empir. Softw. Eng.* **2021**, *26*, 120. [CrossRef]
11. Guzman, E.; Maalej, W. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In Proceedings of the IEEE 22nd International Requirements Engineering Conference (RE), Karlskrona, Sweden, 25–29 August 2014; pp. 153–162.
12. George, L.; Sumathy, P. An integrated clustering and BERT framework for improved topic modeling. *Int. J. Inf. Technol.* **2023**, *15*, 2187–2195. [CrossRef]
13. Chauhan, U.; Shah, A. Topic Modeling Using Latent Dirichlet allocation: A Survey. *ACM Comput. Surv.* **2022**, *54*, 145. [CrossRef]
14. Egger, R.; Yu, J. A Topic Modeling Comparison Between LDA, NMF, Top2Vec, and BERTopic to Demystify Twitter Posts. *Front. Sociol.* **2022**, *7*, 886498. [CrossRef]
15. Jacob, D.; Chang, M.-W.; Lee, K.; Kristina, T. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019.
16. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019. [CrossRef]
17. Ran, X.; Xi, Y.; Lu, Y.; Wang, X.; Lu, Z. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artif. Intell. Rev.* **2023**, *56*, 8219–8264. [CrossRef]
18. Amalina, D.N.; Fauzan, A. A Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) Approach for Identifying Potential Villages in Buleleng Regency. *Knowl. Eng. Data Sci.* **2024**, *7*, 187–199. [CrossRef]
19. Karlsson, J. Software Requirements Prioritizing. In *Proceedings of the Second International Conference on Requirements Engineering, Colorado Springs, CO, USA, 15–18 April 1996*; IEEE: New York, NY, USA, 1996; pp. 110–116.
20. Hudaib, A.; Masadeh, R.; Qasem, M.H.; Alzaqebah, A. Requirements Prioritization Techniques Comparison. *Mod. Appl. Sci.* **2018**, *12*, 62. [CrossRef]

21. Sufian, M.; Khan, Z.; Rehman, S.; Haider Butt, W. A systematic literature review: Software requirements prioritization techniques. In *Proceedings of the 2018 International Conference on Frontiers of Information Technology, FIT 2018, Islamabad, Pakistan, 17–19 December 2018*; IEEE: New York, NY, USA, 2019; pp. 35–40. [CrossRef]
22. Perini, A.; Ricca, F.; Susi, A. Tool-supported requirements prioritization: Comparing the AHP and CBRank methods. *Inf. Softw. Technol.* **2009**, *51*, 1021–1032. [CrossRef]
23. Berander, P.; Andrews, A. Requirements prioritization. In *Engineering and Managing Software Requirements*; Springer: Berlin/Heidelberg, Germany, 2005.
24. Gyani, J.; Ahmed, A.; Haq, M.A. *MCDM and Various Prioritization Methods in AHP for CSS: A Comprehensive Review*; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2022. [CrossRef]
25. Ali Khan, J.; Ur Rehman, I.; Hayat Khan, Y.; Javed Khan, I.; Rashid, S. Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique. *Int. J. Mod. Educ. Comput. Sci.* **2015**, *7*, 53–59. [CrossRef]
26. Kifetew, F.M.; Perini, A.; Susi, A.; Siena, A.; Muñante, D.; Morales-Ramirez, I. Automating user-feedback driven requirements prioritization. *Inf. Softw. Technol.* **2021**, *138*, 106635. [CrossRef]
27. Barbosa, P.A.M.; Pinheiro, P.R.; Silveira, F.R.V.; Filho, M.S. Selection and Prioritization of Software Requirements Applying Verbal Decision Analysis. *Complexity* **2019**, *2019*, 2306213. [CrossRef]
28. Morales-Ramirez, I.; Munante, D.; Kifetew, F.; Perini, A.; Susi, A.; Siena, A. Exploiting User Feedback in Tool-supported Multi-criteria Requirements Prioritization. In *Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017*; IEEE: New York, NY, USA, 2017. [CrossRef]
29. Pattyn, F. Identifying Key Requirements Prioritization Criteria for Software Startups Survival: A Systematic Literature Review. In *Proceedings of the 2025 IEEE/ACM International Workshop on Software-intensive Business (IWSiB), Ottawa, ON, Canada, 27 April–3 May 2025*; IEEE: New York, NY, USA, 2025.
30. Santos, D.; Frota, J.R.; Albuquerque, A.B.; Pinheiro, P.R. Requirements prioritization in market-driven software: A survey based on large numbers of stakeholders and requirements. In *Proceedings of the 10th International Conference on the Quality of Information and Communications Technology, QUATIC 2016, Lisbon, Portugal, 6–9 September 2016*; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2017; pp. 67–72. [CrossRef]
31. Ahuja, H.; Purohit, G.N. Understanding Requirement Prioritization Techniques. In *Proceedings of the 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 29–30 April 2016*; IEEE: New York, NY, USA, 2016; pp. 257–262.
32. Duan, C.; Laurent, Æ.P. Towards automated requirements prioritization and triage. *Requir. Eng.* **2009**, *14*, 73–89. [CrossRef]
33. Lehtola, L.; Kauppinen, M.; Kujala, S. Requirements Prioritization Challenges in Practice. In *Proceedings of the International Conference on Product Focused Software Process Improvement, Kansai Science City, Japan, 5–8 April 2004*; Springer: Berlin/Heidelberg, Germany, 2004.
34. Perini, A.; Susi, A.; Avesani, P. A machine learning approach to software requirements prioritization. *IEEE Trans. Softw. Eng.* **2013**, *39*, 445–461. [CrossRef]
35. Racheva, Z.; Daneva, M.; Herrmann, A.; Wieringa, R.J. *A Conceptual Model and Process for Client-Driven Agile Requirements Prioritization*; IEEE: New York, NY, USA, 2010.
36. Naweer, M.; Aramjoo, B.A.; Rahimiyan, Z. Identification and Prioritization of Factors Influencing Students' Satisfaction with University Services Using the Kano Model (Case Study: Jami University). *Baharestan Sci. Res. Q. J.* **2025**, *2*, 216–219.
37. Yaseen, M.; Mustapha, A.; Ibrahim, N. Prioritization of software functional requirements: Spanning Tree based approach. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 489–497. [CrossRef]
38. Sami, M.A.; Rasheed, Z.; Waseem, M.; Zhang, Z.; Herda, T.; Abrahamsson, P. Prioritizing Software Requirements Using Large Language Models. *arXiv* **2024**, arXiv:2405.01564.
39. Sharma, S.; Kumar, D. Agile release planning using natural language processing algorithm. In *Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI), Kuantan, Malysia, 19-21 February 2019*; IEEE: New York, NY, USA, 2019.
40. Shafiq, S.; Mashkoor, A.; Mayr-Dorn, C.; Egyed, A. NLP4IP: Natural Language Processing-based Recommendation Approach for Issues Prioritization. In *Proceedings of the 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2021), Palermo, Italy, 1–3 September 2021*; IEEE: New York, NY, USA, 2021.
41. Lunarejo, M.I.L. Requirements prioritization based on multiple criteria using Artifnicial Intelligence techniques. In *Proceedings of the 2021 IEEE International Conference on Requirements Engineering (RE2021), Notre Dam, IN, USA, 20–24 September 2021*; IEEE: New York, NY, USA, 2021.
42. Mihany, F.A.; Galal-Edeen, G.H.; Hassanein, E.E.; Moussa, H. Data-Driven Requirements Elicitation from App Reviews Framework Based on BERT. *Appl. Sci.* **2025**, *15*, 9709. [CrossRef]
43. Li, X.; Zhang, A.; Li, C.; Guo, L.; Wang, W.; Ouyang, J. Relational Biterm Topic Model: Short-Text Topic Modeling using Word Embeddings. *Comput. J.* **2019**, *62*, 359–372. [CrossRef]

44. Panichella, S.; Adelina, C. Replication Package for: 'Analyzing Reviews and Code of Mobile Apps for a better Release Planning Organization'. Available online: https://github.com/panichella/UserReviewReference-Replication-Package/tree/URR-v1.0 (accessed on 30 January 2024).
45. Ciurumelea, A.; Schaufelbuhl, A.; Panichella, S.; Gall, H.C. Analyzing Reviews and Code of Mobile Apps for Better Release Planning. In *Proceedings of the 4th IEEE International Conference on Software Analysis, Evolution, and Reengineering, Klagenfurt, Austria, 20–24 February 2017*; Institute of Electrical and Electronics Engineers Inc.: New York, NY, USA, 2017.
46. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* **2020**, arXiv:1802.03426. [CrossRef]
47. Sedgwick, P. Spearman's rank correlation coefficient. *Br. Med. J.* **2014**, *349*, g7327. Correction in *Br. Med. J.* **2018**, *362*, k4131. https://doi.org/10.1136/bmj.k4131. [CrossRef]
48. Eltehiwy, M.; Abdul-Motaal, A.B. A new Method for Computing and TestingThe significance of the Spearman Rank Correlation. *Comput. J. Math. Stat. Sci.* **2023**, *2*, 240–250. [CrossRef]
49. Farea, A.; Tripathi, S.; Glazko, G.; Emmert-Streib, F. Investigating the optimal number of topics by advanced text-mining techniques: Sustainable energy research. *Eng. Appl. Artif. Intell.* **2024**, *136*, 108877. [CrossRef]
50. 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. *Inf. Softw. Technol.* **2022**, *144*, 106798. [CrossRef]