Master Thesis

December 15, 2021

Understanding User Reviews

Identifying Clusters of Similar Issues from User Reviews using BERT

Olajoke Oladipo

of Ibadan, Nigeria (17-722-414)

supervised by

Prof. Dr. Harald C. Gall

Assistant

University of Zurich[UZH]

s.e.a.l.

software evolution & architecture lab

Master Thesis

Understanding User Reviews

Identifying Clusters of Similar Issues from User Reviews using BERT

Olajoke Oladipo

University of Zurich[UZH]

s.e.a.l.
software evolution & architecture lab

Master Thesis

Author: Olajoke Oladipo, olajoke.oladipo@uzh.ch
URL: <https://github.com/olajoke
Project period: 15.06.2020 - 15.12.2020

Software Evolution & Architecture Lab
Department of Informatics, University of Zurich

Acknowledgements

First and foremost, I would like to thank God almighty for his countless blessing towards my life and for giving me the required knowledge, and opportunity to be able to accomplish my thesis. I would also like to express my sincere gratitude to my thesis supervisor Prof. Dr. Harald C. Gall, Dr. Pasquale Salza, and Marco Edoardo Palma of the Department of Business, Economics and Informatics for their guidance and encouragement. Thanks a lot. Many thanks to all the Professors, Doctors, Tutors for their support during my time studying at this prestigious University. I extend my thanks to all my colleagues at the university for their encouragement and support, as well as my friends outside the university for their time, advice, and moral support. Finally, a sense of respect goes to my parent Mr. and Mrs. Oladipo and my family for their strong support economically as well as regular encouragement in every step to make me accomplish these great feet.

Ein Produktentwicklungsteam muss seine Anwendungen fortlaufend verbessern, indem es sich mit Fehlerberichten befasst und neue Funktionalitäten einführt, wobei es die Erfahrungen und Bewertungen der Nutzer verstehen muss. Forscher haben zahlreiche Methoden entwickelt, um Entwickler dabei zu unterstützen, relevante Informationen aus Nutzerbewertungen zu gewinnen. Zu diesen Methoden gehören die automatische Extraktion, die Kategorisierung und der Einsatz von Crowd-Sourcing. In einigen Studien wurde versucht, Nutzerbewertungen zu clustern, indem jede Bewertung als Fehlerbericht, Funktionswunsch, Verbesserung und dergleichen kategorisiert wurde. Bei diesen Strategien kommen Methoden des maschinellen Lernens (ML) in Verbindung mit der Verarbeitung natürlicher Sprache (NLP) zum Einsatz, um bedeutungsvolle Informationen aus den Rezensionen zu extrahieren. Die Benutzerrezensionen und deren Absichten effizient zu verstehen, stellt nach wie vor eine wesentliche Herausforderung dar.

Transformationsbasierte Modelle verstehen Nutzerbewertungen auf effiziente Weise und helfen dabei, wertvolle Erkenntnisse zu gewinnen (wie z. B. BERT, ein von Google entwickeltes vortrainiertes bidirektionales transformationsbasiertes Modell). In dieser Masterarbeit werden drei unüberwachte Clustermodelle in Kombination mit BERT verwendet, um die eingehenden Bedürfnisse der Nutzer zu erfassen. In dieser Arbeit soll untersucht werden, ob ein Transformator-basiertes Modell, insbesondere BERT, die Clusterung von Nutzerbewertungen verbessern kann, wenn a priori keine Informationen über die Anzahl von Clustern vorhanden sind. Zusätzlich wurde die Latent Dirichlet Allocation (LDA) und ein Modell zur Textzusammenfassung eingesetzt, um weitere Erkenntnisse aus diesen Clustern zu gewinnen.

Abstract

A product development team must continually upgrade their applications by understanding their users' experience and reviews through addressing bug reports and introducing new functionalities. Researchers have devised numerous methods to assist developers in retrieving relevant information from user reviews. These methods include automatic extraction, categorization and the use of crowd-sourcing. Some studies have attempted to cluster user reviews by categorizing each review as a bug report, feature request, enhancement, and more. These strategies use Machine learning (ML) techniques in conjunction with Natural Language Processing (NLP) to extract meaningful information from the reviews. A significant leap remains in effectively understanding the user reviews and their intentions.

Transform-based models effectively understand user reviews and aid in the extraction of valuable insights (such as BERT is a pre-trained bidirectional transformer-based model developed by Google). This thesis proposes three unsupervised clustering models combined with BERT to capture in-depth user demands. The thesis aims to determine whether a transformer-based model, particularly BERT, can improve the clustering of user reviews in the absence of a priori information on the number of clusters. Additionally, we leveraged Latent Dirichlet Allocation (LDA) and a text summarization model to derive FURTHER? insights from these clusters.

Zusammenfassung

What is Zusammenfassung?

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Maintaining mobile applications (apps) is just as crucial as the initial development of the application. Hence, a product development team must continually upgrade their applications by understanding their users' experience and reviews through addressing bug reports and introducing new functionalities to ensure their solutions remain viable as technology and the market climate evolve [4–6]. A practical method of data gathering for developers to effectively manage their applications is the user feedback system. User feedback is typically composed of user rating system, a quantitative input defined mainly via the use of stars ranging from 1 (lowest) to 5 (highest), and user reviews. These reviews are textual feedback that may convey information relevant to a product development team to manage their apps [7]. For instance, bug reports or defects in an app [8], user experience checks [9], request to implement new features [10] , or overall users satisfaction (i.e., to measure how good or bad the application performs) [11]. Moreover, several studies have established that user reviews collected from online app stores such as Google Play Store or Apple App Store are densely packed with information relevant for developers to enhance their application constantly [4, 12–14].

However, manually extracting meaningful information (e.g., a request for a new feature) from user reviews is quite an arduous task. It can be a time-consuming, labour-intensive and inaccurate task, especially for popular applications (such as Twitter) that receive thousands of feedback in a day. Researchers have devised numerous methods to assist developers in retrieving relevant information from the reviews. These methods include automatic extraction, categorising the reviews into groups to help the developers maintain their program functionality through the implementation of new features, and timely resolution of problems [5, 12, 14–16]. Specifically, some of these studies solved the associated drawbacks using supervised models by preparing the feedback data under ideal situations and categorising each review as a bug report, feature request, enhancement, and more. For Instance, ARDOC tagged reviews using a taxonomy based on the information relevant to developers [12, 17]. URR categorises reviews accordingly into high-level and low-level their taxonomy [18]. CLAP [5] clusters only user reviews defined as bug reports or feature requests. SURF group users reviews based on designed keywords associated with concepts describing a user's intents relevant to a software maintenance task [14]. These methods are inefficient to work with due to the following limitations: (i) Inability to integrate fine-grained details, (ii) challenges extracting only valuable information from a large number of user reviews, (iii) and undermining the wealth of information that other members of a product team can access and benefit during a product release planning. Moreover, all of the above strategies use Machine learning (ML) techniques in conjunction with Natural Language Processing (NLP) to extract meaningful information from the reviews. Asides from this, the studies mentioned above neglect the fickle nature of user reviews. User reviews are submitted by individuals with a variety of backgrounds and regions. Multiple users may have different opinions on bug in a variety of ways and formats. Consequently, we need to understand the context of reviews in order to clus-

ter and efficiently gather insights from these reviews. This approach of understanding context of reviews that focuses on the information or issue discussed in a particular review, regardless of how it is presented. Recently, the unprecedented advancement in Machine Learning (ML), Deep Learning (DL) techniques especially in solving an ample number of tasks (including Natural Language - text comprehension) and generate better performances [19,20].

In many NLP sequence processing tasks, Recurrent neural networks (RNNs) have been the state-of-the-art model [21]. RNNs, on the other hand, are incapable of long-distance modelling associations between sequence tokens. Transformer architectures have recently overcome this constraint and outperformed RNNs in a variety of natural language processing tasks [22]. We believe a transformer-based model is effective at understanding user reviews and aid in the extraction of valuable insights. Transformers models specialise in understanding and recognising the core context of the text [22] as it holds the potential to understand the relationship between sequential elements that are far from each other. This allows the models to understand the text as a whole and its context as well [22]. For this study, we will focus on BERT [2], a popular component of a transformer architecture. BERT is a pre-trained bidirectional transformer-based model developed by Google, trained on Wikipedia data and book corpus, allowing it to understand the diverse nature of text in different language [2]. According to a 2020 survey by Anna et al., BERT is currently the state-of-the-art model in NLP experiments with numerous research studies continually evaluating and developing the it [23].

As a result, this thesis proposes three unsupervised clustering models combined with BERT to capture in-depth user demands and simplify the massive quantity of data app owners get daily from their users.

Below are the three proposed two-stage models:

- **M1: BERT + Kmeans using Elbow Methods.** This model encodes reviews with BERT. Pass the encoded reviews to Kmeans, then choose the optimal K with the elbow method.
- **M2: BERT + Kmeans using Silhouette Method.** This model encodes reviews with BERT. Pass the encoded reviews to Kmeans, then choose the optimal K with the Silhouette method.
- **M3: BERT + Cosine Similarity.** This model encodes reviews with BERT and cluster encoded reviews using the cosine similarity function.

  Further, we investigate these three models to determine which model will best capture similarities in user reviews and produce quality solutions concerning the goal of this thesis.

The primary goal of this thesis is to determine whether a transformer-based model, particularly, BERT can improve the clustering of user reviews in the absence of a priori information on the number of clusters. Further, we also aim to derive in-depth insights from the clusters. We tend to leverage LDA and a text summarising model to derive insights from these clusters.

Research Questions

This thesis is designed according to the following research questions:

- RQ1: To what extent do BERT implementation and cluster algorithms coherently identify similar reviews that describe similar concerns and improve productivity?
- RQ2: To what extent do the tagging (topic modelling) and description (summary) features generated from clusters provide quick and meaningful insights to developers?

The following are the significant contributions of the thesis: (1) The design and implementation of three two-stage unsupervised clustering techniques using the current state-of-the-art deep

learning model to identify and cluster reviews describing similar issues. (2) The inclusion of topics tags through using an unsupervised approach (LDA) on the cluster results, and (3) Generating summaries description to simplify the cluster reviews using an extractive deep learning summarization model The source code and dataset for the model are accessible at .

The rest of the thesis is structured as follows: Chapter 2 describes the state-of-the-art categorising user reviews in supervised and unsupervised settings and text clustering tasks using BERT. Chapter 3 describes the theoretical background of the BERT model and the terminologies associated with our approach. While Chapter 4 describes our approach and specifies the planned methodology to answer the research questions, Chapter 5 discusses the lack of analytical techniques, the preliminary results, and the results from implementing CHANGEADVISOR's data on our algorithm. Finally, chapter 6 concludes with a summary of our approach. It also highlights the contributions of this thesis and an outlook on future research in this area.

Chapter 2

State of art Approach

This chapter presents an overview of the major relevant researches in information extraction from the reviews of mobile app users. It is divided as follows: Section 2.1 states studies to categorizing user reviews using supervised models, whereas Section 2.2 states research categorising user reviews using unsupervised models, and Section 2.3 describes a study that fine-tunes BERT for a text clustering task. Finally, concludes this chapter with a summary highlighting the similarities, dissimilarities, and measures adopted in our approach as well as how we selected the articles for this chapter.

2.1 Clustering Reviews with Supervised Models

- ARDOC

App Reviews Development Oriented Classifier (ARDOC) [17] is a review classifier tool devised by Pancichella et al. a tool that incorporates three different approaches for the automated detection of meaningful feedback included in apps reviews: Natural Language Parsing (NLP), Text Analysis (TA), and Sentiment Analysis (SA). ARDOC categorized reviews using a taxonomy developed to simulate the information requirements of developers proposed from its prior research [12]. The developed taxonomy is represented as Information Giving, Information Seeking, Feature Request and Problem Discovery and others (reviews that do not provide developers with any relevant input). The underlying approach of ARDOC is Parser, which constructs a pipeline with annotations for tokenization and sentence splitting. After separating the texts into sentences, ARDOC extracts 3 distinct types of information from each sentence, that is the lexicon (via TA), the structure (via NLP), and the emotion (via SA). Then, train and classify the extracted features app reviews using a pre-trained machine learning model with the WEKA API (citation) using a collection of 852 manually labelled sentences randomly picked from user reviews of 7 popular applications. ARDOC conducted two experiments involving external validators with expertise in software development and the original developer of 3 real-life applications. ARDOC achieved high precision, F-measure, and recall scores ranging from 84% to 89%, and thus the developers of the tools also validated ARDOC's effectiveness in extracting and classifying meaningful information present in their application particular in maintenance tasks.

- SURF

Summarizer of User Reviews and Feedback (SURF) [14] is a pioneering technique for summarizing the huge amounts of data that developers need to handle. SURF works by summarizing

several hundreds of reviews and generates an interactive, organized, and concise agenda of recommended software using enhanced summarization algorithms. SURF works by first categorizing user reviews based on the defined intention classes motivated by ARDOC [12]. Cluster the result of classification using a predefined set of procedures, to cluster sentences that pertain to the same review topic, and eliminate duplicates in the summary. Then use a scoring technique to extract sentence importance sentences.

Di Sorbo et al. implemented two different experiments (one of which is a controlled experiment) which included 17 real-world applications and 23 participants with varying backgrounds in the area of software development. The first aim focused on assessing the quality of extracted feedback and its usefulness for developers; the second aim is to prove the practical use of SURF's summaries in a work setting. The SURF participant established SURF as a tool for generating relatively accurate, adequate, concise, and assertive summaries. Additionally, SURF is recommended for real-world use [14].

- URR

The User Request Referencer (URR) [18] prototype uses Machine Learning and Information Retrieval techniques to automatically identify reviews according to their taxonomy and indicate which source code files need to be updated. Specifically, URR classified reviews into six high-level *(Compatibility, Usage, Resources, Pricing, Protection Complaint)* and 12 low-level taxonomy *(Device, Android Version, Hardware, App Usability, UI, Performance, Battery, Memory, Licensing, Price, Security, Privacy)*, so that each review is labelled with a list of categories it belongs to from the high and low-level taxonomy [18]. In the first stage, URR uses a preprocessed text that is pre-processed and pre-populated with words that are characteristic of a certain taxonomy category. After that, in addition to the TF-IDF[1], the study applies N-grams based features, which are retrieved from keywords in each review to capture clusters of words that belong to a particular category. URR then trains the Gradient Boosted Regression Trees (GBRT) ML classifier on the features extracted from user reviews (i.e., the N-grams and TF-IDF features). URR evaluated its system to determine how accurately it categories user reviews according to the taxonomy on the reviews and code of 39 mobile applications. With the assistance of two external evaluators with experience in mobile development (from academia and industry), at both the high and low levels of taxonomy, URR established promising results for the majority of categories. The overall precision, recall, and F-measure scores were 83%, 94%, and 87%, respectively, across the high-level categories, and 80%, 94%, and 87% for the overall precision, recall, and f-measure across the low-level categories [18].

2.2 Clustering Reviews with Unsupervised Models

- CLAP

Crowd Listener for releAse Planning (CLAP) [5] is a framework initiated by Villarroel et al to categorize and group relevant user reviews, then automatically prioritize clusters of reviews for developers to easily examine when preparing an app release. CLAP clusters reviews are restricted in two specific categories (i.e., bug report or suggestion for new feature) with DBSCAN [24]. DBSCAN is a clustering algorithm identifying clusters as areas of high element density, assigning the elements in low-density regions to singleton clusters (i.e., clusters only composed by a single element). According to Villarroel et al, to assist with release preparation procedures, clusters of bug report assessments and recommendations for new feature categories are prioritized. Clusters with a high priority are those that CLAP recommends being included in the next app release while a review cluster with a low priority rating indicates features that need to be developed soon [5].

[1] "Term Frequency — Inverse Document Frequency"

Villarroel et al. assessed the effectiveness of CLAP clusters by evaluating it against a manually clustered set of user reviews. They extracted the reviews from android applications: Facebook, Twitter, Yahoo Mobile Client, Viber, and Whatsapp. A total of 40 reviews, 20 bug reports, and 20 feature requests were randomly chosen from each of these applications, all related to the same app's release. Then, they engaged three industrial developers with combined expertise of more than two decades in manually clustering the reviews. Following that, analyzed the separate clustering findings collaboratively and presented a single "oracle" [5]. By comparing the results of the oracle to CLAP's result, an average MoJoFM of 73% on bug reports and 87% on feature requests, indicating a close agreement between clusters constructed manually and that of CLAP [5].

- CHANGEADVISOR

Palomba et al. [15] devised a novel approach, namely CHANGEADVISOR. CHANGEADVISOR aims to automatically extract users' reviews from a maintenance standpoint, identify, and cluster reviews conveying similar user needs, then link the clustering results to the part of the application's source code responsible for changes. Before CHANGEADVISOR cluster users' reviews, it extracts and classifies the reviews using ARDOC [17]. CHANGEADVISOR only examines categorized requests to fix bugs and feature enhancements, which correspond to ARDOC's problem discovery and feature request categories, accordingly. Palomba et al. believed that reviews classified as Information Giving and Information Seeking do not pertain to recommendations for source code modifications or give detailed information on how to fix bugs and feature enhancements. CHANGEADVISOR adopted three topic modelling strategies for the clustering tasks: Latent Dirichlet Allocation (LDA) proposed by Asuncion et al [25], LDA using Generic Algorithm LDA-GA by Panichalle et al [26], and Hierarchical Dirichlet Process (HDP) by Teh et al [27]). They finalized on HDP based on its trade-off between quality solution and execution time [15].

Palomba et al. conducted two human experiments to validate the cohesiveness of its clustering technique on a total of 44683 users reviews of 10 open-source mobile apps extracted from the Google Play Store [15]. The first experiment involved two external validators to express their opinion using a Likert intensity scale (ranging between 1: very low and 5: very high). The result from this experiment established a high accuracy of CHANGEADVISOR's clustering approach given an overall median distribution of 4, while the values are mainly placed between 4 (high) and 5 (very high). The results from the second experiment strengthen the clustering approach of CHANGEADVISDOR as the experiment generated a high level of cohesiveness by the original developer of the applications.

2.3 Fine-tuning BERT for Text Clustering tasks

- Unsupervised Fine-tuning for Text Clustering

According to Huang et al. [3], multiple research have shown impressive results in a variety of language comprehension tasks in a supervised environment; however, only a few studies have examined BERT citedevlin2018bert implementation in an unsupervised environment, notably in text clustering tasks. As a result of this, Huang et al. developed a novel technique for fine-tuning BERT in an unsupervised scenario on a text clustering task. The proposed technique uses a clustering-based loss function to simultaneously learn text representations and cluster assignments. The study experimented on three text clustering datasets (TREC-6, Yelp, and DBpedia) and outperforms the baseline techniques worked with and claimed their approach achieved state-of-art results. The paper does this by fine-tuning the BERT [2] model on a lower task. A masked-language model in which portions of the input tokens are randomly masked and subsequently predicted. The final hidden representations matching to the mask tokens are placed in a softmax layer over the vocabulary. The masked language model loss Lm is optimized by lowering

the negative log-likelihood. They use average pooling to construct the text representation as $z_i = \sum_j^N h_{i,j}/N$, where h_i is the hidden vector of the jth word in sample x_i [3]. Along with the language mask loss, they use a clustering loss with the representation z_i, which is aimed to train representation distributions using an auxiliary target distribution [28]. The clustering loss is defined as the KullbackLeibler KL (see formula **??**) divergence between the distributions P and Q, where Q is the distribution of soft assignment as defined by the Student's t-distribution and P is the target distribution derived from Q [3].

$$L_c = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_i j}{qij} \tag{2.1}$$

Although the text clustering work conducted in this study is unrelated to our objective, which entails categorizing user reviews in to extract valuable information. We adapted our algorithm's technique from this study since Huang et al. also examined two "two-stage clustering algorithms - one that utilizes the k-means algorithm on the pre-trained autoencoder's features (AE+Kmeans) and another that uses the method on the original BERT's average hidden vectors (BERT+Kmeans)" [3]. As shown in Table 2.1 *"BERT+k-means"* exhibits relatively high performance, showing a huge difference in purity compared to other baseline models and importantly, a close difference to Huang et al's method. This demonstrates that a more accurate text representation can be learned from a pre-trained model as BERT for text clustering tasks [3].

Table 2.1: Clustering purity on three datasets experimented by Huang et al. [3]

Dataset	k-means	DEC	IDEC	DCN	AE+k-means	BERT+k-means	Our method
TREC-6	20.87%	23.52%	26.33%	24.63%	21.96%	38.51%	**39.91%**
DBpedia	15.01%	21.77%	20.43%	18.51%	17.15%	63.25%	**66.99%**
Yelp	20.64%	23.12%	23.51%	22.34%	21.53%	30.02%	**33.40%**

Chapter summary In the previous subsections, we discussed important studies that in the context of categorizing user reviews in supervised and unsupervised settings, as well as a study on fine-tuning BERT for clustering tasks. We picked these articles for their relevance to our methodology. Apart from Huang et al study's which strongly influenced the algorithm we developed for our approach, we share common objectives as earlier described in chapter1 with these reasearches [5, 14, 15, 17, 18]. As opposed to our method, ARDOC, SURF, CLAP, URR, and CHANGEADVISOR categorize reviews specifically from a maintenance perspective, restricting the information collected to the application's core developers. Our technique also aim to uncover information that is beneficial to other members of a product team as well. Such as a quality assurance manager who monitors whether products adhere to a company's defined standards. With the exception of ARDOC, CLAP, and SURF, which classify reviews into broad categories such as bug reports, URR offered a fine-grained classification that addressed extensive categorizations such as price and licensing. While the majority of these studies generate clusters of reviews and leave interpretations to the developer, SURF additionally summarizes cluster findings, allowing the developer to quickly comprehend the problem addressed by a cluster. In this thesis, we added both the summary technique and topic attachments as additional features in our approach to aid

quick understanding of user's needs. Finally, compared to the traditional ML model employed by most studies, our strategy included deep learning techniques with the aim of maximizing performance with a DL language model such as BERT.

Chapter 3

Background

3.1 Bidirectional Encoder Representations from Transformers (BERT)

RNNs, the state-of-the-art model for several NLP sequence processing tasks replaced by the Transformer-based models [21]. Due to its inability to model long-distance relationships between sequence tokens as vanishing or exploding gradients begin to occur when RNNs model on long textual data. This means the signal from the initial token becomes weak by the time the RNN reaches the final token [29]. Even though approaches such as attention mechanisms augmented RNNs and addressed this limitation. Yet other limitations remain. Such as handling data sequentially and it being computationally slow to train [29].

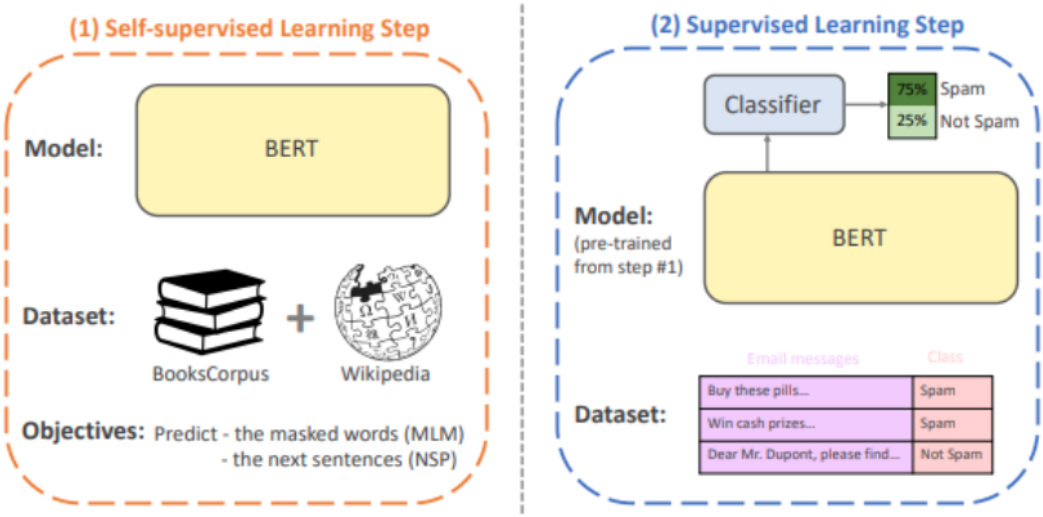

Figure 3.1: Learning steps of BERT (L) Pre-training step: performing self-supervised training on large text (R) Fine-tuning step: performing supervised training with a labelled dataset

On the other hand, Bidirectional Encoder Representation from Transformer (BERT), a component of Transformer-based architecture, is faster to train and deeply bidirectional as it learns contexts at all levels of the representation from both left and right directions simultaneously [2]. BERT is pre-trained on Wikipedia (2,500M English words) and Bookcorpus (800M words). Numerous NLP tasks, such as text classification, benefit significantly by simply fine-tuning the underlying model with a single additional output layer [2]. Figure 3.1 illustrates BERT's pre-training step using the self-supervised training method on a large corpus of wiki data and the fine-tuning step using the supervised learning method on a labelled dataset for a specific task. The architecture of BERT is based on a multi-layer bidirectional transformer encoder [2, 29]. According to Devlin et al (Devlin et al., 2018), BERT contains L-identical transformer encoder layers that are interconnected. A layer of an encoder can be divided into two sublayers. The Multi-head self-attention mechanisms, which keep track of encoded-words as it encodes another word. The feed-forward network (FFN), a basic and fully linked feed-forward network (FFN) applied individually and identically to each location, is also, composed of two linear transformations. The output of each sublayer is normalized using a residual connection in an encoder layer, followed by a layer normalization. While BERT employs the same linear transformations over numerous locations inside a single sublayer, each layer has its own set of parameters. In the feed-forward network, a GELU activation is also employed . Figure 3.2 depicts a 3-Dimensional representation of BERT (Left) and the architecture of a single encoder (Right). Despite the fact that BERT utilizes the same linear transformations across multiple points in the same sublayer, each layer has its own set of parameters.

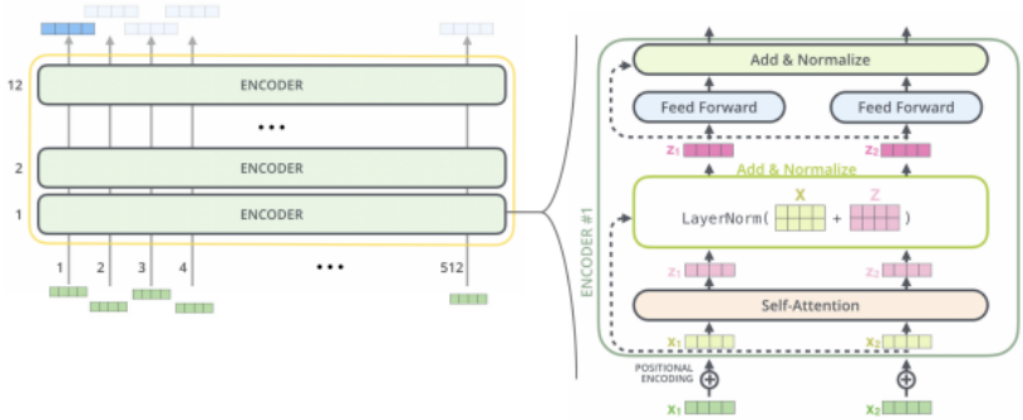

Figure 3.2: BERT-encoderArchitecture of Transformers Encoder Model [1]

BERT-base and BERT-large are the two versions of BERT, with the aim of BERT-large (L=12, H=768, A=12, Total parameter = 110M) enhancing the performance of BERT-base (L=24, H=1024, A=16, Total parameter = 340M) [29]. Where (L) denotes encoder layers, (H) hidden units, (A) attention heads, and the total parameters respectively. Figure 3.3 shows the high level of BERT's architecture and its two versions. Each layer's input and output are represented by the dimensionality of the models, the number of attention heads per layer (A), and the number of hidden units per layer (H) [29].

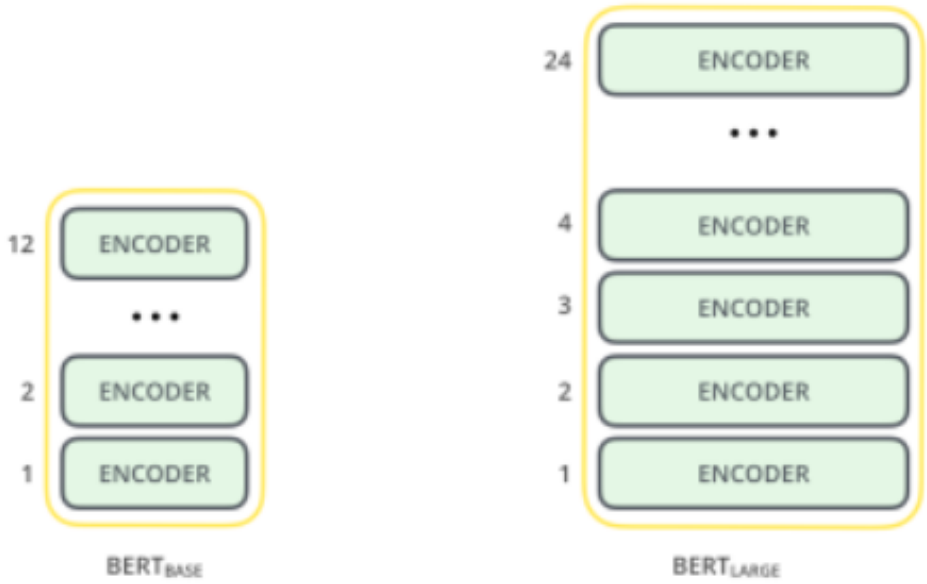

Figure 3.3: Overview diagram of BERT architecture and its two versions [1]

Input Representations

BERT performs a preliminary transformation on a collection of input words (limited to 512 tokens) to create numerical input representations for the model. it is customary to mix three distinct types: a token, segment, and positional embedding as depicted in Figure 3.4 to get an Input representation.

The token embedding function as BERT tokenizes words in the input sequence using the Word Piece embedding (Wu et al., 2016) approach. This vocabulary-building technique employs a predefined set of characters, subwords, and words to construct a vocabulary of a fixed size to fit within a language's corpus. BERT's vocabulary, for example, contains approximately 30,000 of the most commonly used words and subwords (e.g., "es","ed","ly"), along with all English characters and three special tokens: [CLS], [SEP], and [MASK]. The segment embeddings are similar to token embeddings when the vocabulary is size two. When working with sentence pairs, each token has a segment embedding that identifies which sentence it corresponds with. BERT, as the original Transformers, leverages positional embeddings to infuse information about the location of the tokens in the input sequence. To summarize, these embeddings are identical to the token and segment embeddings in terms of dimension. It is computed using the sine and cosine functions with different frequencies as represented in formulas (1) and (2) respectively.

These formulae make use of position and dimension. As a result, the positional embedding's wavelengths form a geometric progression from 2π to $10000{\cdot}2\pi$ [29], with each dimension representing a sinusoid. Vaswani et al. claim that by utilizing this function, the model can simply

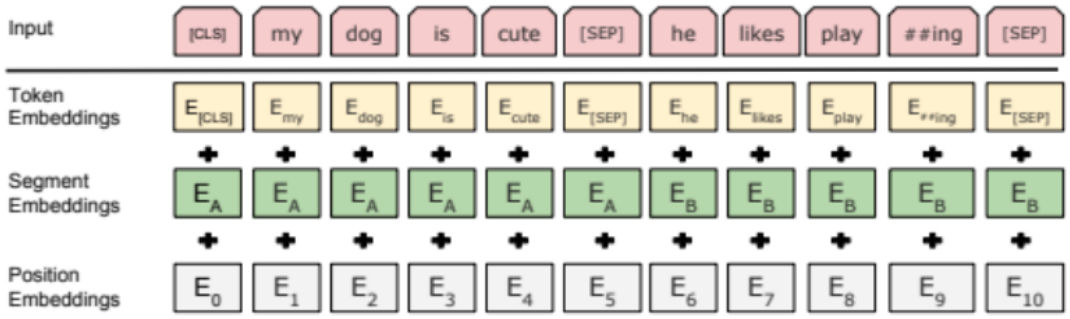

Figure 3.4: Diagram of a BERT's input representation for each token in an input sequence [2]

learn the relative positions of each word since every fixed position can be represented as a linear function of PE_{pos}.

3.2 Clustering Algorithm

Clustering is the process of grouping data points based on their similarity, such that the data points in each group are more similar to one another. Cluster algorithms are unsupervised machine learning techniques that require merely a dataset with data points not labelled. In contrast to supervised machine learning tasks such as classification and regression, which requires a labelled dataset.

Clustering techniques are broadly classified into three categories depending on their corresponding function: density-based, hierarchical, and centroid-based techniques. Density-based algorithms construct clusters by searching for the dense regions in the data while using the low-density regions to determine the clusters' boundaries. centroid-based divides a dataset into a predefined number of groups based on the similarity or distance between data samples. The hierarchical technique generates a tree of clusters via hierarchical methods. The following subsections describe the typical examples (i.e., spectral, agglomerative, and k-means respectively) of the clustering techniques mentioned above

3.2.1 K-Means

K-means clustering is a popular, simple yet efficient unsupervised learning technique. it categorises data points according to the number of clusters k, provided k is known ahead of time. K-means produces an iteratively revised final grouping depending on the number of clusters k chosen by the user and the dataset involved. Initially, k-means picks k as the mean value of k clusters, referred to as the centroids, then seeks the closest data points to the chosen centroids to create k clusters. This procedure is repeated for each cluster until the algorithm finds a single optimal value for each centrifugal point. As a result of using numerical data to determine the means, K-means clustering works best with low-dimensional numerical datasets. The algorithm is implemented as follows:

1. Initialize random points based on the given value of k
2. Create k clusters by calculating the distance between each data point and the nearest centroid. Then get each data point's distance from the initialised centroids using the Euclidean distance
3. Re-calculate centroids by averaging all of the data points allocated to each cluster reduces the overall intra-cluster variation
4. Repeat steps 2 and 3 until the desired outcome is achieved

When the values of the centroids stay unchanged, the sum of the distances between the data points and the centroid of each cluster and the data points assigned to the clusters remain constant. The advantage of K-means is due to its simplicity as it does nothing more complicated than calculating and comparing distances between data points and organising clusters based on those distances. As a result, it has a time complexity of *O(n)*, making it faster to compute than hierarchical clustering [30]. Asides from this, K-mean is quite flexible and applicable to clusters of all forms and sizes, including elliptical clusters [30]. On the other, to operate K-means, one must manually specify the number of clusters k. Similarly, K-means randomly selects the initial centroids of the k clusters. Due to this, outcomes are likely to differ between executions, even when no inconsistent behaviour exists. Finally, K-means are sensitive to outliers or noise in the data, as the outliers may form a cluster [31].

3.2.2 Hierarchical Clustering

Hierarchical clustering methods are used to create a hierarchy of clusters. The process begins with a few small clusters and works its way up to the final output. The algorithm treats each data point as a cluster and uses a specialised proximity-matrix to compute the distance between clusters. There are two types of hierarchical clustering. (1) agglomerative, a bottom-up approach in which initially treats each data point as a distinct cluster, then iteratively merges the clusters until the final cluster has all the data points in the cluster that was generated from the initial cluster. As an alternative to agglomerative clustering, (2) divisive uses a top-down approach that begins with a single cluster containing all of the data points and gradually divides the cluster into smaller ones until each cluster has a single data point. Similar to k-means, an advantage of the hierarchical clustering algorithm, is simple to implement. In some instances, the number of clusters is not required since a cluster tree (dendrogram) can be produced. Figure 3.5 is a graph showing an example of a dendrogram representing the clustering approach of the hierarchical clustering algorithm. The primary disadvantage of hierarchical clustering is its time requirement. In comparison to other algorithms, it is *O(log N)* in time complexity, where n is the number of input data points. Managing clusters of various sizes and convex forms can also provide difficulties [32]. Finally, depending on the distance matrix chosen, it may also be susceptible to noise and outliers.

3.2.3 Density-based Spatial Clustering (DBSCAN)

DBSCAN is a clustering algorithm that does not require a predefined number of clusters. However, there are only two parameters that DBSCAN needs: eps and minimum samples [33]. The maximum distance between two data samples is specified by eps. This term refers to the bare minimum number of samples that must be within a certain distance of one another to be termed a neighbourhood. It is assumed that a cluster is a dense region with more data points than the minimum sample size within the eps range of the core point. Advantages of DBSCAN include operating effectively on a dataset that emphasises high-density clusters over low-density clusters.

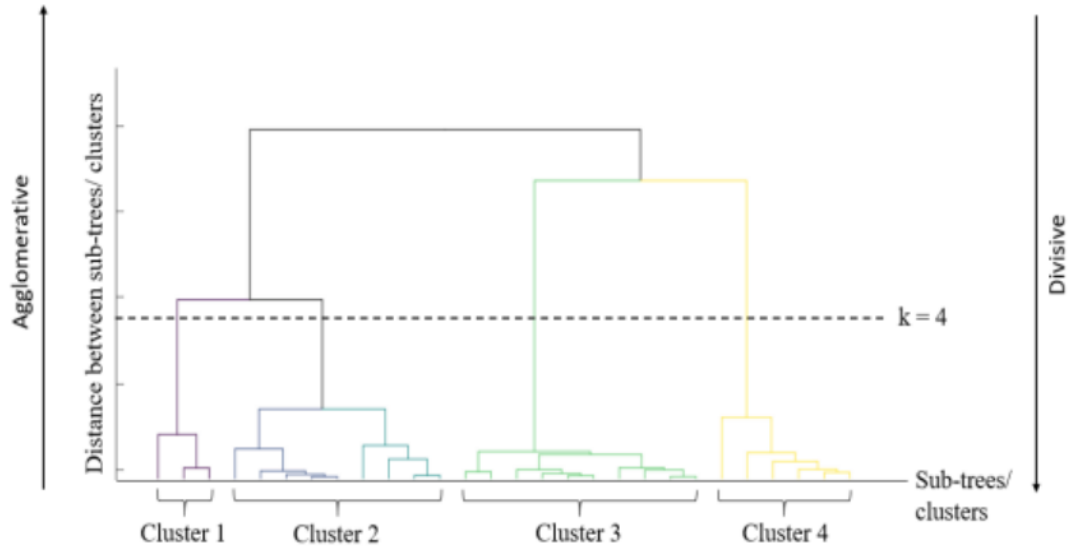

Figure 3.5: A dendrogram illustrating the hierarchical clustering algorithm's clustering approach

It is noise-resistant and capable of dealing with outliers in the dataset. As with k-means, It is not necessary to specify the number of clusters in advance. Despite the fact that DBSCAN does not require a specified number of clusters, it is challenging to estimate the distance between eps when clusters have a variety of densities [33].

3.3 Determining Number of Clusters

3.3.1 Elbow

The elbow method is a technique for determining the best number of clusters. It works by calculating the percentage of the difference between the number of clusters that form an elbow at a given point. This method generates ideas or concepts for selecting cluster values by combining it to form a data model for identifying the optimal clusters. Sum of Squared Error (SSE) is how the elbow approach is expressed. The SSE is calculated as the sum of each point's average Euclidean Distance from the centroid. When the value decreases significantly and the angle becomes smaller, the value of k is obtained. Starting with *k= 2*, and increasing the SSE value incrementally, where *kn = k + 1*, the point with the highest SSE, *kn - SSE Kn - 1* is the ideal k value [34]. Computed in formular 3.1:

$$SSE = \sum_{i=1}^{k} \sum_{X_i \epsilon S_k} \|X_i - C_k\|_2^2 \tag{3.1}$$

Where C_i is *i-th* cluster, k is the number of clusters formed and x is the data in each cluster.

SSE is only applicable as a cluster assessment metric for approaches in which the cluster may be represented by the centroid. Using this metric in conjunction with clusters obtained using other approaches (such as DBSCAN) might result in misleading results [34].

3.3.2 Silhouette

The silhouette approach calculates the degree to which the data is similar to a given cluster. This is calculated by computing the silhouette value for each data point and then averaging the result across the overall dataset. The silhouette measure is effective as it takes into account both the distance between clusters and the distance within the cluster. Kaufman et al. (citation here) established the notion of silhouette coefficient, which encompasses individual and cluster silhouette coefficients. The average distance between a data point and all other data points inside a cluster is $a_(i)$ while the minimal average distance between the data point and other clusters is $b_(i)$. An individual silhouette coefficient for a cluster is determined using the formular 3.2.

$$S_i = \frac{b_(i) - (a_(i)}{max[b_(i).(a_(i)]} \tag{3.2}$$

The silhouette score S_i is bound to a range between -1 and 1. As -1 indicates that no data-point fits to its allocated clusters and 1 indicates that all data-points fit perfectly to its assigned clusters. Similar to the elbow method, the silhouette metric is most effective when used in conjunction with centroid-based clustering algorithms as such Kmeans. For instance, rule-based or hierarchical clustering approaches do not prioritize performing distance minimization as centroid-based methods do. Consequently, silhouette scores cannot adequately represent approaches other than centroid-based methods. Besides that, the silhouette measure takes longer to compute compared to the elbow method [35].

3.3.3 Cosine Similarity

The cosine similarity technique is used to estimate the degree of similarity between two vectors. Formular 3.3 mathematically expresses how the cosine value between two vectors is determined [**?**].

$$¿(Q, D_i) = \frac{\sum_{j=1}^{V} w_{Q,j} \times w_{i,j}}{\sqrt{\sum_{j=1}^{V} w_{Q,j}^2 \times \sum_{j=1}^{v} w_{i,j}^2}} \tag{3.3}$$

Where Q is the Query, D_i is the document *i*, and W_Q, j is the weight of *j* term in Q query, and the weight of *j* term in *ith* document. Significant similarity exists between two vectors if the estimated value provided by the cosine similarity technique is greater than or equal to 1. If

the calculated value is less than or equal to 0, the two vectors are said to have a low degree of similarity. The calculation ranges from 0 to 1. If none of the two vectors used in the computation is identical, the value is 0. The value is 1 if the vectors are identical. [36]

3.4 Topic Modelling

Topic modelling, one of the most effective means of text mining for NLP tasks such as latent data identification and information extraction. It is an unsupervised machine learning approach for identifying and detecting word patterns in text datasets. An ideal topic model technique generates clusters of words (topics) that best describe a collection of related documents in a corpus. According to [37]., the Latent Dirichlet Allocation (LDA) technique is the most commonly used method for topic modelling [37]. LDA is a generative statistical theory that states each word in a document is linked to one of its topics [38]. Each document may be seen as a collection of distinct topics, with each topic typically allocated to a set of words with LDA. LDA is developed on the assumption that both the distribution of topics within a document and the distribution of words within topics are Dirichlet[1] distribution [38]. There are three main hyperparameters that regulate the model, Alpha (α is the parameter of the Dirichlet prior on the per-document topic distributions) and Beta (β is the parameter of the Dirichlet prior on the per-topic word distribution), and the number of topics k. A low αvalue will result in fewer topics and vice versa, and a low β value model suggests a topic with fewer words and vice versa. Thereby making the topics more comparable. The third hyperparameter is the number of topics k, which requires one to manually specify before initialising LDA [38].

3.5 Text Summarization

The widespread availability of data on the Internet has piqued researchers' interest in developing ways for condensing large amounts of data into a relevant summary. Text summarization is a technique for compressing long documents into compact texts (typically less than half the length of the original text), however, preserves the critical information and overall meaning from the original text [39]. Given the massive volume of resources available online, human summarization becomes inconceivable, tedious, and time-consuming. Automatic text summarization can improve the readability of a document by drastically reducing the time to read and understand it. For example, individuals seeking a certain document in a sea of thousands could considerably benefit from a system that automatically exemplifies source texts and collates all necessary details. Thus, a text summary is a vital tool for individuals who require continuous and quick access to essential information such as news headlines, product reviews, and snippet results [40]. In general, there are two methods of text summarization i.e., extractive and abstractive text summarization.

3.5.1 Extractive summarization

In extractive summarizing, significant and exact text snippets from the source material, such as sentences or phrases, are extracted and concatenated together to construct the summary [41]. Extractive summarising aims to determine each text sentence', create a condensed version of the original text that accurately portrays it. There are several ways available to achieve an extractive summarization task, some based on linguistic qualities and others on statistical traits [42]. In the following paragraph, we will examine a few of these strategies in further detail.

[1] Dirichlet distribution is the conjugate prior that encodes the notion that documents contain a limited number of topics and that those topics typically use a limited number of words

The statistical approach. This approach of text summarizing is built on the basis of sentence extraction. Each phrase in a text is given a weight based on the document's word frequency [43]. "Sentences with high frequent occurrence are deemed more suggestive of the subject and should be included in the summary" [43]

The graph-theoretic. This technique represents the structure of a document as an undirected graph. The edges linking the texts indicate their relationships after the preprocessing processes. Additionally, it is regarded to be evocative of the document's multiple topics due to the subgraphs generated by displaying relationships between words. Similarity scores are utilized to determine the common words between phrases in order to create an edge (connection) between nodes (sentences) [41, 43].

Neutral Network. The implementation of deep learning models in text summarization tasks has increasingly become popular since neural networks have proven to be successful [44]. Before a model is set for a summarization task, it must be trained on large datasets to understand the patterns and features of individual words. Ultimately, the model will learn which sentences should be included in the final summary by studying the features of the sentences [42, 43].

3.5.2 Abstractive summarization

The abstractive summarization method mimics human-written summaries in a manner that naturally conveys the content and meaning of the source material. This has acquired popularity as a result of its capacity to generate new phrases to that holds critical information from the text sources [45]. Abstractive summaries can be created in two ways: (1) by paraphrasing (rephrasing original idea into shorter and more coherent text)[2] or, (2) by reformulating (rephrasing the original idea to have slightly different meaning)[3] [39]. In general, abstractive summarising is more efficient, however, its implementation requires substantial knowledge of deep learning techniques; hence, the majority of researchers prefer to use or dig deeper into extractive summarising [39, 45].

3.6 Dimensionality Reduction

PCA is a common technique for reducing the dimensionality of a dataset with numerous features while retaining the maximum amount of variation possible. The primary aim of this technique is to easily visualize and analyze data with more than three dimensions. The initial collection of characteristics is linearly reduced to a more manageable set of attributes identified as the principal components [46]. The following are four mathematical steps involved in the PCA process: (1) Determine the data's origin and re-position the data and its origin by averaging the columns and deducting the original data from the average. (2) Create a matrix of correlations (3) Calculate eigenvalues and eigenvectors from the above-generated correlation matrix (4) Calculate the transform values by interpreting the eigenvalues and eigenvectors. PCA is a least-squares approach that allocates large loadings to characteristics with high variation in the PCA output. In this study, PCA is used to reduce the dimensionality of embeddings generated from the deep learning model. Superimposing the two types of plots allows both objects and attributes to be displayed at the same time.

[2] https://dictionary.cambridge.org/dictionary/english/paraphrase
[3] https://dictionary.cambridge.org/dictionary/english/reformulate

Chapter 4

Approach to Evaluating Reviews

4.1 Overview of Approach

The thesis aims to provide meaningful insights and improve performance by clustering user reviews with similar issues, topic tagging, and including a summary. To establish that, we designed a novel approach using BERT as a text representation of user reviews on the first stage, passing the output for clustering using K-means with Elbow (M1), K-means with Silhouette (M2), and Cosine Similarity (M3) on the second stage. We then extracted features from these clusters, by generating topic tags (via topic modelling) and the description (via text summarization) to give a high-level overview of what is going on in a cluster. Finally, we employed a dimensionality reduction approach to creating an interactive visualization of the data. The following involves the steps in our approach:

1. Cluster user reviews with BERT
   (a) Text preprocessing
   (b) Encode Reviews with BERT
   (c) Cluster encoded data using M1 (i.e., BERT + K-Means with Elbow Method)
   (d) Cluster encoded data using M2 (i.e.,BERT +K-Means with Silhouette Method)
   (e) Cluster encoded data using M3 (i.e.,BERT + Cosine Similarity)
2. Add topic tags feature to output cluster data of M1, M2, and M3
3. Add description feature to output cluster data of M1, M2, and M3
4. Perform PCA function on output cluster data of M1, M2, and M3
5. Visualize M1, M2, and M3 on an interactive plot

4.2 Text Clustering with BERT

4.2.1 Text Preprocessing

The review dataset is directly scraped from google play store, having emojis, characters,signs and other irrelevant things in them. We preprocess the data to only extract words, numbers, spaces,

commas and dots from the reviews and exclude rest(puntuation, emoji,). This helps us remove unwanted characters and emojis from the dataset that can cause the model and lda to malfunction. Specifically for LDA we added another layer of preprocessing to remove stop words from it as LDA is proun to stop words occurring frequently.

4.2.2 Encoding text with BERT

As previously indicated, user reviews are textual feedback often provided informally by a variety of people with varying backgrounds and experiences. To help a system better grasp the context of these reviews, we employed BERT, a deep learning and state-of-art language model that has shown impressive results at understanding human language as established by various studies [47]. We used BERT, specifically, the **bert-base-uncased**[1] [48] to construct a 768-dimension embedding from the huggingface[2]. An AI and NLP community for hosting open-source libraries and state-of-art models. Our technique further processed these embeddings. Essentially, we employed the pre-trained BERT without fine-tuning due to a lack of considerable data and resources that may have resulted in more accurate findings. The technique necessitates a substantial volume of reviews data in a compatible format for fine-tuning BERT, which was not accessible. Furthermore, to get a text representation of the data, we pass the raw data (unprocessed user reviews) into the BERT using its default hyperparameter (see the model's paper for detailed information on the hypeparameter [48]) to create a *N x 768* embedded data, where *N* represents the length of the data.

4.2.3 K-Means with Elbow Method - M1

Encoding user reviews in a *(N x 768)* dimensional matrix enables efficient clustering. Our algorithm begins by implementing M1 on the embeddings BERT generated in the initial stage. In this approach, we did not manually choose a k since there is no way to predict the number of clusters to expect from each app review data. As a result, we implement the elbow method of automatically determining the cluster size, i.e., k for the K-means. We initiated K-means with sklearn library with a maximum iteration of 300. We assigned k values between 1 and 50 and saved the corresponding inertia values accordingly. There is no concrete reason for choosing 50 as the maximum value for k. We, however, based on the type of data (mobile reviews) and our domain knowledge, decided on this value. Finally, we locate the elbow in the list of inertias by using the KneeLocater library, a python library that selects a knee point at the point of maximum curvature [49]. Then, using the selected k as the final value of k to train the K-means model and label the reviews according to their relevant cluster.

4.2.4 K-Means with Silhouette - M2

For the second method, M2, we used the same embedded output from the aforementioned BERT-encoder step, which generated the N x 768 matrix. Here, we used the embeddings to implement K-means using silhouette. Similar to the elbow method, Silhouette is another technique for choosing an optimal value of k in centroid-based clustering techniques such as k-means. We utilized the same range between 1 and 50 for the silhouette and maintained the hyperparameter values to synchronize and precisely compare this approach to the elbow. Our approach also uses sklearn to initialize the silhouette to calculate the corresponding scores for k-means across the specified range. Moreover, we also adopted the Kneelocater library to determine the knee-point based on

[1]This model is uncased, which means the distinction between English and English is irrelevant
[2]https://huggingface.co/bert-base-uncased

the silhouette scores generated. After that, we train the model on the selected k value and label the reviews according to the relevant cluster.

4.2.5 Cosine Similarity - M3

For this, we use the cosine similarity method, i.e., M3 of our approach. To define the clusters, we constructed a cosine similarity matrix using the originally obtained vector embeddings from BERT. In contrast to M1 and M2, the models do not require specifying a range of values for the cluster size. Based on the embeddings, this method determines the optimal number of clusters. The method processes all vector embeddings generated from the sentences. It begins by generating a matrix of size pxp, where p is the number of embeddings. The matrix duplicates the similarity of each embedding to other embeddings. Following that, embeddings with a similarity higher than 70% are clustered together and presented as distinct clusters, also known as local communities (sets of highly similar sentences). Embeddings that do not share a threshold of similarity with any other embeddings are coerced into a cluster of single points.

4.3 Topic Modeling - Topic Tags

From the cluster results of the methods mentioned in the preceding subsections, that is, M1, M2, and M3, we have successfully generated a cluster of similar reviews. In this step, we briefly describe one of the added features, topic tags. Therefore, this step aims at extracting insights from user reviews with topic tags to understand the major topics discussed in a cluster. To achieve this, first, we initialised the LDA model, a topic modelling technique (described in section-name), on each cluster to extract the top 5 keywords based on their frequency rate from the cluster. We implemented LDA using the Python Genism library[3], preprocessed the data by removing stopwords and lemmatizing the words, created a bag of word models such as bigrams from the sentence, and trained the LDA model. In the end, for each cluster, the top five topics are generated.

4.4 Text Summarization - Description

Text summarisation, as defined in section-name (), is the practise of compressing a large text into a short, and cohesive text for easy consumption. Leveraging on the importance of text summarising, we incorporated a summary feature to the cluster review generated by our models to facilitate a quicker comprehension of the issue a cluster depicts. We utilised a deep learning extractive summarization model in this stage to extract the key sentences and words that best summarise each cluster. The extractive summarization model we adopted is also a Hugginface model [48]. Precisely, "paraphase-MiniLM-L6-v2". It works by transforming the phrases and paragraphs to a 384-dimensional vector space before passing them to the output to construct the summary. The length of sentence created is determined by factors that are predefined. In our method, we chose 3 sentences to display.

4.5 Visualization with PCA

To graphically display the results of cluster analysis, we created a visual representation of the reviews along with the topic tags to conceptually aid understanding. As mentioned earlier, PCA

[3]https://radimrehurek.com/gensim/

is an approach for reducing high-dimension data into a low-level dimension, primarily for easy visualization. Hence, in this step, we reduced the *N x 768* to a *N x 2* vector space data using the sklearn PCA library *(note, we specified number components = 2)*. We then use an interactive plotly [4] scatter graph to show the clusters and topics.

4.6 Validation

To understand the approach's effectiveness in (a) grouping reviews according to the aforementioned technique, (b) topic modelling and text summarization per cluster. This section describes the planned manual analysis, a human experiment that unfortunately could not happen due to limited time resources and other means of evaluation metrics we explored.

4.6.1 Manually Analysis

This subsection describes how to manually develop a collection of reviews into groups representing similar issues that more closely reflect the unique and actionable problems associated with mobile applications.

To create these groups, on a subset of the dataset's reviews, we use an iterative content analysis strategy similar to that suggested in [50]. We set up a validation dataset by manually grouping the reviews into a fine-grained granularity suitable for our algorithm. First, we aim to collect the reviews from the latest versions of the apps as we believe such reviews will address the most current issues in the mobile app. Thus, we excluded apps with less than 500 reviews, as some apps may have fewer reviews to guarantee that a small number of users did not skew the categorised reviews we evaluated. We tend to collect reviews by randomly selecting a considerable number of open-source apps from the Google Play Store and Apple Store, spanning multiple categories such as games, news, and finance. We developed a web scraper tool to extract corresponding reviews of the apps; however, for iOS apps, reviews are obtained from AppComments [5]. The web scraper will collect.review data such as the app name, the review title, and the textual feedback [50].

Second, we planned to study the statistical representation of the whole dataset. Hence, we will generate the statistical sample and randomly choose and select reviews for the task using a confidence level and interval of 95% and 5%, respectively. For instance, if 1000 reviews were pulled for an app, the statistical sample size would be 278 at a 95% confidence level and an interval of 5%. This signifies that the app's result has a ±5% margin of error[6].

Finally, the main task, which involves grouping the reviews using an iterative method. It starts with an empty list of categories and, as one reads each review, extra categories are included if the review is worthy of a new group. Each time a new class is created, relevant keywords and a brief description are included. When there are no more reviews left, the overlapping groups will be combined. The reviews will be subjected to another iteration to assign the most appropriate groups to each review from the final list [50]. It is important to note that a review can only belong to a unique group . This is especially important since our solution is concerned with clustering reviews that point to similar issues. In the dataset of reviews, there are some reviews with no meaningful or not beneficial comment from a developer's perspective, for example, "this is a bad app" or "It's an awesome app!" such reviews are labelled "Less Informative" and put together these reviews in separate clusters. Even though such reviews are less informative to the developer, other members of the product team, such as the quality assurance manager, get

[4] https://plotly.com/python/
[5] http:// appcomments.com, a web service that gathers user reviews for all iOS applications.
[6] https://surveysystem.com/sscalc.htm, a Sample Size Calculator website

beneficial information from these clusters [51]. Moreover, we did not build our algorithm to filter out these reviews like other studies [5, 15] since we have no target classes available in our data. Therefore, we expect our algorithm to have these set of reviews included in the clusters.

4.6.2 Validation Methods

We planned to conduct a human experiment using at least two external inspectors with a combined experience of at least five years to validate our results. We tend to ask the participants to rate (1) the cohesion of our cluster algorithm for each app in our study, and (2) how relevant are the keywords and summary generated for each cluster. Then validators will express their opinion using a Likert scale intensity scale with values ranging from 1 to 5, with 1 denoting very low and 5 denoting extremely high [52]. However, as stated previously, this evaluation phase did not occur due to a lack of sufficient time resources. In the following paragraphs, we describe other ways to justify our cluster's performance.

In general, there are two viable methods for evaluating cluster analysis's performance: internal and external measures. The internal measure checks cohesion and the distance between the resulting clusters. The typical types of this are the elbow and silhouette methods that we already described and implemented in the section above (I will mention the exact section later) to select the best K Value for the K-means cluster. These two internal measures are relevant metrics because they are also used to optimise clustering methods such as K-means.

The eternal measure compares the result of the cluster to the ground-truth labels. One popular type of this measure we explored is the cluster purity. "Purity is a statistic for evaluating the quality of groups constructed from labelled samples. If the purity value is one, it is termed pure" [53].This suggests that the whole labeled set is appropriately categorised. Similarly, when a cluster's value is close to or equal to zero, it is said to be impure as it indicates that each item with a label is a member of a different class [53]. To determine a clustering's purity, one requires a target variable that was not used throughout the clustering process, and unfortunately, no target variable is available in our dataset. Therefore, we cannot also cannot perform external validation. However, we investigated further the techniques previous studies employed to label their dataset. We tend to consider the strategies and then complete the cluster purity thereof. In the discussion section, we critically analysed the techniques described in the previous studies and stated why we could not employ those techniques as well.

Chapter 5

Discussion and future work

5.1 Preliminary Result

The initial aim of this research is to cluster the user reviews using different techniques to generate meaningful insights from the clustered user reviews. As a part of the methodology section of our research, we used three clustering techniques: BERT + Kmeans using Elbow Methods (M1), BERT + Kmeans using Silhouette Methods (M2), and BERT + Cosine Similarity (M3). All these three techniques are expected to generate clustering results with high efficiency and precision. By using these deep learning clustering techniques, we also expect that our implemented algorithm generates better results and gives better performance when compared to previous literature studies and their implemented techniques.

5.2 Experiment Results

We experimented with the technique proposed in our approach using Changeadvisor's dataset. That is the 10 open-source mobile applications which were retrieved from the Google Play Store [15]. We applied our compiled Python script to analyze the text dataset, which is accessible on https://github.com/olajoke. In the following we present and analyze the findings of our experiment in the form of tables and graphs.

After implementing the three models (M1, M2, and M3) on the data from CHANGEADVISOR, Table 5.1 displays all the Apps included in the data and the matching number of clusters created for each of the three clustering methods. As we can see, M1 detected a minimum of 13 clusters and a maximum of 17, M2 identified a minimum of 15 clusters and a maximum of 10, while M3 created a minimum of 8 clusters and a maximum of 28 clusters. Indeed, the three approaches create widely differing numbers of clusters.

Figures 5.1 - 5.3 and tables 5.2 - 5.4 are representation of each of our cluster techniques result upon successful completion of a modeling process on for each app reviews. Each plot displays all reviews from a particular dataset, where the data points represent the user reviews and the colors represent the unique cluster class with the LDA keywords corresponding to each cluster is shown on the right side of the graph. For the tables, LDA topics, the frequency, and the summary of a unique cluster class (a subset of the clustered review represented as cluster sentences) upon successful completion of a modeling process given an app's review data

The table 5.2 above shows the results of clustering using K-means with elbow algorithm technique. We have selected four cluster topics randomly from all the generated clusters. The first cluster listed in the table 5.2 signifies suggestive reviews from users about the apps. As shown

Table 5.1: Number of clusters generated by M1, M2 and M2 for each app reviews

App Name	M1	M2	M3
ACDisplay	13	6	13
FrostWire	11	8	23
K-9 Mail	12	8	23
Solitaire	13	10	24
Shortyz Crosswords	13	6	8
SMS Backup +	15	5	22
Focal	14	7	9
Cool Reader	17	8	19
FB Reader	16	5	25
WordPress	16	5	28

Table 5.2: M1 results on the LDA topics, the frequency, the summary, and a subset of the corresponding clustered sentences (i.e reviews) of “FB Reader”.

LDA topics	Frequency	Summary	Cluster sentences
Option, library, set, left, book, ability, select,	239	It only shows the first 27 pages of the book I was reading.Easy to find books neat having a bookstore built in a wonderful look. I just wish I could read mobi format n convert PDF to smaller footprint easy read text. Had to move to Moon reader because it has sync, hopefully it will be implemented but I won't hold my breath.	Best reading app in Google Play Store. No comments. Plenty of customization, including fonts, colors, size, and UI. Well done. Pretty awesome reader. faster , smaller efficient than other readers
Can read, suck ,zoom, fbreader, epubbud, forever	48	Except I cant delete books from the library once read them. Plus it dow play .mobi eitha GLXYS2GB.cant do without it. Cant find my folders.	Best reader app on my devices. Need sync utility to follow my Android tablet and WP8 phone, though. Really easy to use. need PDF and docx support though. Also it freezes sometimes when there are a lot of books in a folder. Please add pdf compatibility
Work well, support, light, efficient, comfortble, smooth	127	Really Good app. Truly effective and efficient. Just what I need. Simple and easy to use. Love this reader. Fast, clean and reliable. Just excellent. Good for what I need..Really nice app.	Simple and easy to use. Display brightness problem it makes reading books super easy
Reader, perfect ,read, love, book, application, good	93	Just the best reader there is. Best reader hands down. Best application for reading books. Best book reader around.	Perfect reader for me Best reader by far. Perfect reader. Love all features

from the listed reviews in the most right column of the table, the topics of this cluster are persistent with the reviews.

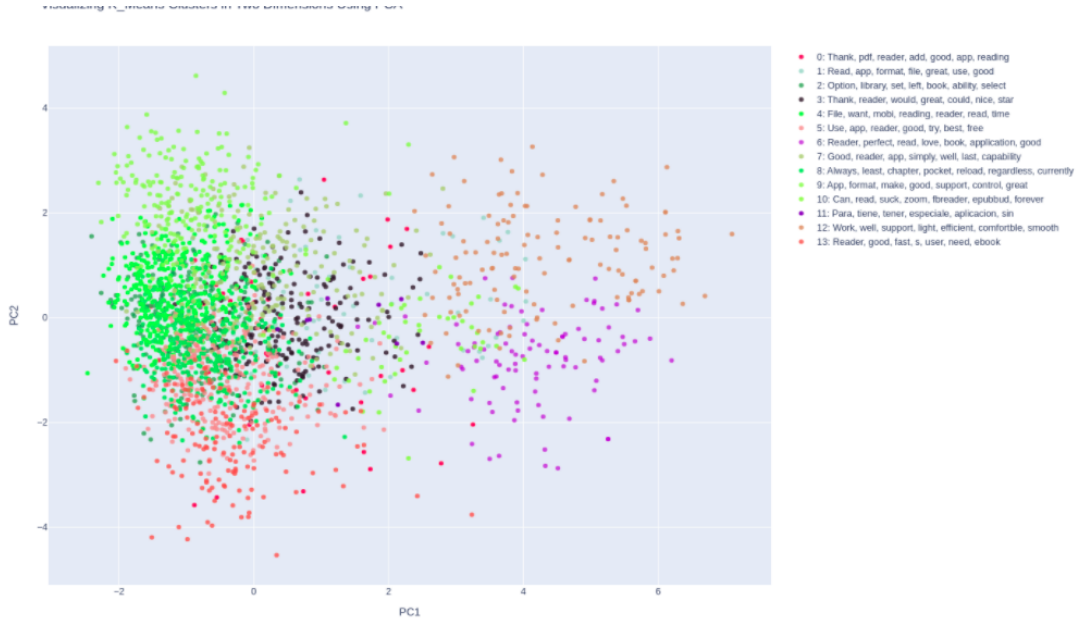

Figure 5.1: A scatter plot representing the clusters generated for “FB Reader” reviews using M1 with the topic tags as the legend.

Table 5.3: M2 results on the LDA topics, the frequency, the summary, and a subset of the corresponding clustered sentences (i.e reviews) of “FB Reader”.

LDA topics	Frequency	Summary	Cluster sentences
Reader, good, best, free, book, use, dictionary	391	My favorite reader app fantastic features and reliability. Thank you for moving the wifi usage to a separate plugin.This is easily the best epub reader on the market, it is fast reliable, and so adaptable.	Best reading app in Google Play Store. No comments. Plenty of customization, including fonts, colors, size, and UI. Well done. Pretty awesome reader. faster , smaller efficient than other readers
Option, feature, read, good, add, device, make,	395	Can't find book style page switching as displayed on screenshots.However, lacks font size control shortcuts. Yet the program doesn't support pdf and djvu formats that are pretty often used for ebooks.	Best reader app on my devices. Need sync utility to follow my Android tablet and WP8 phone, though. Really easy to use. need PDF and docx support though. Also it freezes sometimes when there are a lot of books in a folder. Please add pdf compatibility
Love, problem, thank, job, do, right, absolutely	158	Works fine on my DesireS.Love this thanks.Simple and effective.good application. So reliable n easy to use .It works well.Good for what it does. Does everything I want and more	Simple and easy to use. Display brightness problem it makes reading books super easy
Reader, good, use, ever, far, thank, android,	7	Would have given 5 stars if new books were also made available. Great book reader does what it says. Best reader by far	Perfect reader for me Best reader by far. Perfect reader. Love all features.

The table 5.3 shows the clusters generated from the dataset using K-means with the Silhouette clustering technique. The first and fourth cluster listed in the table shows a positive trend generated using the positive reviews. As we can see, the topics generated from the clusters are not precise as there is a mix of "good" and "bad".

The clustering technique with Cosine Similarity Algorithm generated 25 unique clusters. Out

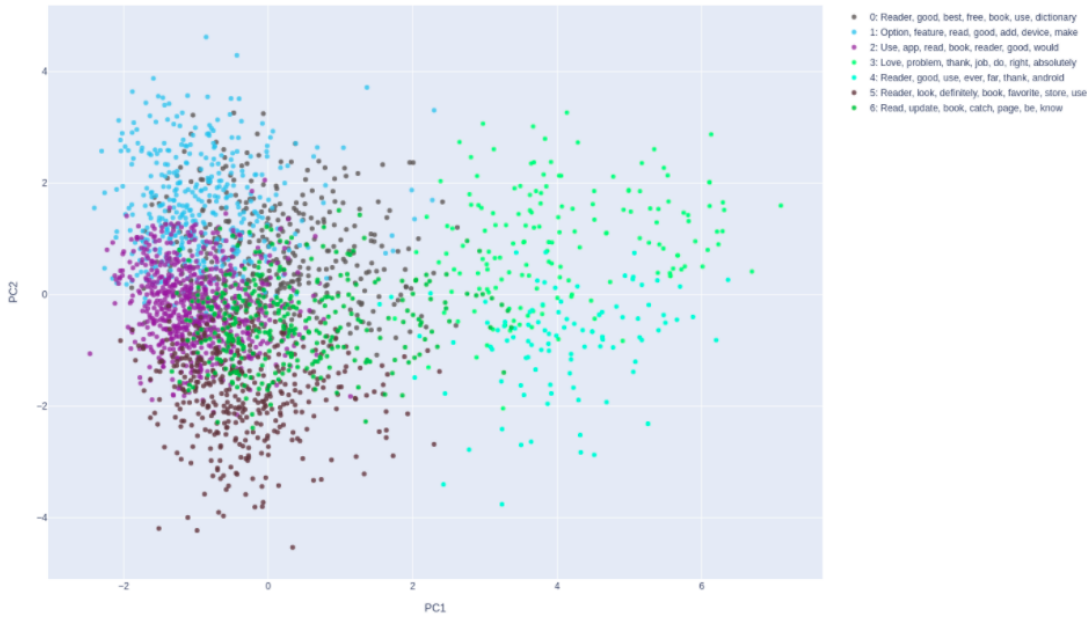

Figure 5.2: A scatter plot representing the clusters generated for “FB Reader” reviews using M2 with the topic tags as the legend.

Table 5.4: M3 results on the LDA topics, the frequency, the summary, and a subset of the corresponding clustered sentences (i.e reviews) of “FB Reader”.

LDA topics	Frequency	Summary	Cluster sentences
Well, work, free, thank, get, use app	2074	Great App. I use this every day to read. Unfortunately, the latest update made it so that the navigation menu will not stay open so I can select anything. Easy to use, lots of personalization options and works with multiple formats, the best ereader. it is the best reader ever with perfect options the best addons.	Best reading app so far. No crashes. I changed my review, great app, so far I really like this reader, but it is broken for Droid DNA. The status bar was taking up a significant part of the screen so I turned it off, now I have no menu button at all.
Problem, update, correct, display, installing, install, cover	9	Unable to install. Update problems. Minor presentation issues only.	poor format of doc Display brigthness problem Update problems
Exactly, need, want, ask, could, excellent, good	7	Good for what I need. Does everything I want it to do.	Good for what I need. Exactly what I need Excellent. Everything I need
Simple, effective, stable ,reader, plain, free feature	48	Simple and it works. Absolutely love using it. Very usable and handy. Comfortable, simple, fast..Easy to use.Great features, simple and very stable.	Really powerful and nicely done Very nice and light reader Simple and perfect

of these 25 clusters, we randomly selected four clusters as shown in table 5.4 above. The first cluster is not defined as one unique topic; instead, it is a mix of two, not a refined one. Another major flaw in this technique is that it gives an empty summary for some clusters.

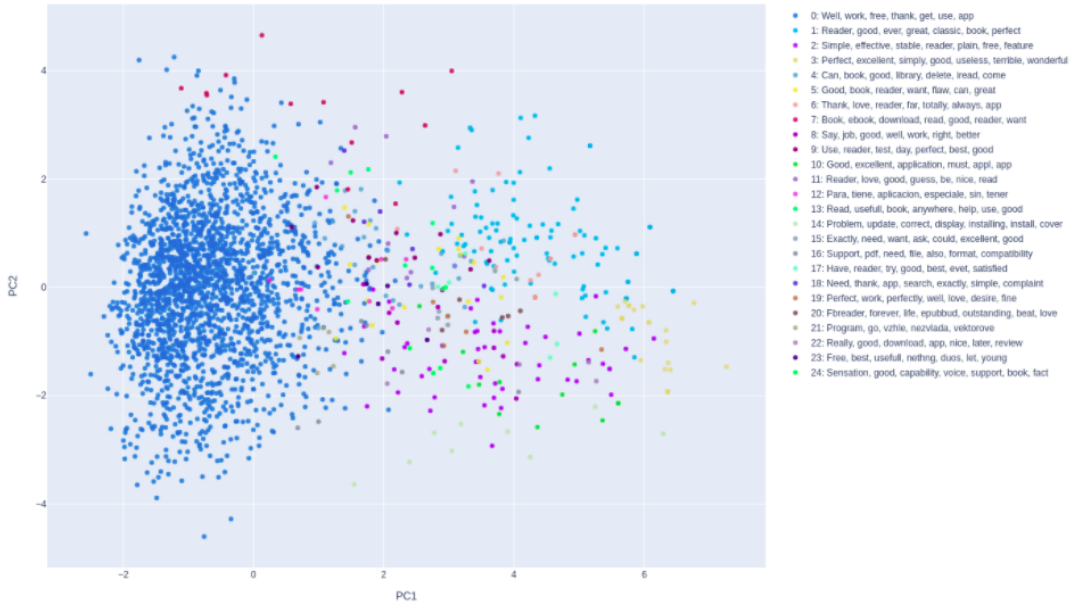

Figure 5.3: A scatter plot representing the clusters generated for “FB Reader” using M3 with the topic tags as the legend.

Chapter Summary We experimented with three clustering techniques to generate valuable insights from user reviews on the dataset used in CHANGEADVISOR’s study. During this process, we did not use any analytical technique to either qualitatively or quantitatively analyse the result of our clustering techniques. One main reason behind not using any analytical technique is the deficiency of data experts and resources. We need data experts who can generate the clusters manually for an analytical process. Moreover, generating manual clusters is time-consuming and requires adequate time and planning. Hence, shortage of time is another reason why this research implied no analytical techniques.

Chapter 6

Conclusion

We primarily carried out this research to generate valuable insights from a dataset of user reviews. We implemented three clustering techniques; Clustering with Cosine Similarity, K-means with elbow and K-Means with Silhouette. The clusters generated using this technique were refined, somewhat accurate, directly persistent topic tags and simplified summary with the user's reviews. With this, there is a likelihood of generating clusters from user reviews that can be helpful for future code recommendation and release planning thereof. The major limitation in our study is the lack of time resources as this remarkably caused manual and Human experiments to occur. Hence, no concrete evidence and analytical techniques to support the goodness of our model. Another limitation is no availability of a huge dataset that can help us fine-tune the BERT model on user reviews.

As mentioned earlier, the primary limitation in our study is due to lack of time resources. Due to this, a future recommendation is to carry out manual analysis techniques and human-evaluation methods to analyze the performance of the generated results. This includes generating clusters manually and comparing the results generated with the deep learning techniques used in this research to analyze the performance of these techniques. We used the base version of BERT in the current methodology, which was trained on a general dataset not specific to our use case. In essence, generating a review dataset and fine-tuning BERT on user reviews can increase the model understanding of reviews and improve the overall performance.

Bibliography

[1] Jay Alammar. The illustrated bert, elmo, and co. *How NLP Cracked Transfer Learning*, page 38, 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. 2018.

[3] Shaohan Huang, Furu Wei, Lei Cui, Xingxing Zhang, and Ming Zhou. Unsupervised fine-tuning for text clustering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5530–5534, 2020.

[4] Fabio Palomba, Mario Linares-Vásquez, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. *Crowdsourcing user reviews to support the evolution of mobile apps*, volume 137. Elsevier, 2018.

[5] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. *Release planning of mobile apps based on user reviews*. 2016.

[6] Anthony Finkelstein, Mark Harman, Yue Jia, William Martin, Federica Sarro, and Yuanyuan Zhang. App store analysis: Mining app stores for relationships between customer, business and technical characteristics. *RN*, 14(10):24, 2014.

[7] Venkata N Inukollu, Divya D Keshamoni, Taeghyun Kang, and Manikanta Inukollu. Factors influencing quality of mobile apps: Role of mobile app development life cycle. *arXiv preprint arXiv:1410.4537*, 2014.

[8] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *2013 21st IEEE international requirements engineering conference (RE)*, pages 125–134. IEEE, 2013.

[9] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162. IEEE, 2014.

[10] Laura V Galvis Carreno and Kristina Winbladh. Analysis of user comments: an approach for software requirements evolution. In *2013 35th international conference on software engineering (ICSE)*, pages 582–591. IEEE, 2013.

[11] Xiaozhou Li, Boyang Zhang, Zheying Zhang, and Kostas Stefanidis. A sentiment-statistical approach for identifying problematic mobile app updates based on user reviews. *Information*, 11(3):152, 2020.

[12] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *2015 IEEE international conference on software maintenance and evolution (ICSME)*, pages 281–290. IEEE, 2015.

[13] Claudia Iacob and Rachel Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *2013 10th working conference on mining software repositories (MSR)*, pages 41–44. IEEE, 2013.

[14] Andrea Di Sorbo, Sebastiano Panichella, Carol V Alexandru, Junji Shimagaki, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. What would users change in my app? summarizing app reviews for recommending software changes. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 499–510, 2016.

[15] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. *Recommending and localizing change requests for mobile apps based on user reviews*. 2017.

[16] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th international conference on software engineering*, pages 767–778, 2014.

[17] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. Ardoc: App reviews development oriented classifier. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering*, pages 1023–1027, 2016.

[18] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C Gall. *Analyzing reviews and code of mobile apps for better release planning*. 2017.

[19] Hang Li. *Deep learning for natural language processing: advantages and challenges*. 2017.

[20] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. *Recent trends in deep learning based natural language processing*, volume 13. IEEE, 2018.

[21] Seunghoi Kim, Byoung Hun Min, Minjae Kang, and Pavlos Demetriou. Comparative analysis of deep learning-based news topic classification models.

[22] Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 179–183. IEEE, 2020.

[23] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[24] Nishant Jha and Anas Mahmoud. Mining non-functional requirements from app store reviews. *Empirical Software Engineering*, 24(6):3659–3695, 2019.

[25] Hazeline U Asuncion, Arthur U Asuncion, and Richard N Taylor. Software traceability with topic modeling. In *2010 ACM/IEEE 32nd International Conference on Software Engineering*, volume 1, pages 95–104. IEEE, 2010.

[26] Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimilano Di Penta, Denys Poshynanyk, and Andrea De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *2013 35th International Conference on Software Engineering (ICSE)*, pages 522–531, 2013.

[27] Lu Ren, David B Dunson, and Lawrence Carin. The dynamic hierarchical dirichlet process. In *Proceedings of the 25th international conference on machine learning*, pages 824–831, 2008.

[28] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[30] Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(suppl_1):S22–S29, 2001.

[31] Shudong Huang, Yazhou Ren, and Zenglin Xu. Robust multi-view data clustering with multi-view capped-norm k-means. *Neurocomputing*, 311:197–208, 2018.

[32] K-means clustering algorithm.

[33] George Seif. The 5 clustering algorithms data scientists need to know. *Towards Data Science*, 2018.

[34] Purnima Bholowalia and Arvind Kumar. Ebk-means: A clustering technique based on elbow method and k-means in wsn. *International Journal of Computer Applications*, 105(9), 2014.

[35] Duy-Tai Dinh, Tsutomu Fujinami, and Van-Nam Huynh. Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient. In *International Symposium on Knowledge and Systems Sciences*, pages 1–17. Springer, 2019.

[36] Christopher C Paige. Computational variants of the lanczos method for the eigenproblem. *IMA Journal of Applied Mathematics*, 10(3):373–381, 1972.

[37] Alan Ritter, Oren Etzioni, et al. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, 2010.

[38] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[39] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational linguistics*, 28(4):399–408, 2002.

[40] Jason Brownlee. A gentle introduction to text summarization. *Machine Learning Mastery*, 29, 2017.

[41] Vishal Gupta and Gurpreet Singh Lehal. A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*, 2(3):258–268, 2010.

[42] Deepali K Gaikwad and C Namrata Mahender. A review paper on text summarization. *International Journal of Advanced Research in Computer and Communication Engineering*, 5(3):154–160, 2016.

[43] Saeedeh Gholamrezazadeh, Mohsen Amini Salehi, and Bahareh Gholamzadeh. A comprehensive survey on text summarization systems. In *2009 2nd International Conference on Computer Science and its Applications*, pages 1–6. IEEE, 2009.

[44] Yue Dong. A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*, 2018.

[45] N Moratanch and S Chitrakala. A survey on abstractive text summarization. In *2016 International Conference on Circuit, power and computing technologies (ICCPCT)*, pages 1–7. IEEE, 2016.

[46] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[47] MV Koroteev. Bert: A review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*, 2021.

[48] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[49] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a" kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[50] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E Hassan. *What do mobile app users complain about?*, volume 32. IEEE, 2014.

[51] Emanuel AM Mjema, MAM Victor, and MSM Mwinuka. Analysis of roles of it on quality management. *The TQM Magazine*, 2005.

[52] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[53] Noufa Abdulaziz Alnajran. *An integrated semantic-based framework for intelligent similarity measurement and clustering of microblogging posts*. PhD thesis, Manchester Metropolitan University, 2019.