INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Navigating the Mobile App Galaxy: Harnessing Textual Metadata for App Categorization

Pedro Afonso Marques d'Oliveira

Master in Computer Engineering

Supervisor:
Doctor Ricardo Daniel Santos Faro Marques Ribeiro, Associate Professor, Iscte – Instituto Universitário de Lisboa

Co-Supervisor:
Doctor Fernando Manuel Marques Batista, Associate Professor, Iscte – Instituto Universitário de Lisboa

September, 2024

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Navigating the Mobile App Galaxy: Harnessing Textual Metadata for App Categorization

Pedro Afonso Marques d'Oliveira

Master in Computer Engineering

Supervisor:
Doctor Ricardo Daniel Santos Faro Marques Ribeiro, Associate Professor, Iscte – Instituto Universitário de Lisboa

Co-Supervisor:
Doctor Fernando Manuel Marques Batista, Associate Professor, Iscte – Instituto Universitário de Lisboa

September, 2024

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my supervisors, Professor Ricardo Ribeiro and Professor Fernando Batista. Their invaluable support, insightful suggestions and scholarly contributions have been instrumental in the completion of this thesis. Thank you for sharing your knowledge and guiding me through this journey.

I would also like to thank the team at INESC-ID for providing access to their servers, which greatly increased the computational power available for my research.

Last but not least, I am deeply grateful to my family for their unwavering support. To my parents, João and Cristina, my sisters, Vania and Inês and my brothers-in-law, Edgar and Nelson, thank you for your unconditional support, comforting words and encouragement. Your belief in me has been a source of strength and motivation throughout this endeavor.

Pedro Afonso Marques d'Oliveira

Resumo

Este estudo efectua uma análise comparativa dos métodos de representação de texto e de extração de características para categorizar aplicações móveis em categorias predefinidas. A categorização eficaz melhora a capacidade de descoberta das aplicações, a experiência do utilizador e a organização do ecossistema de aplicações.

Para desenvolvermos uma abordagem automática para categorizar as aplicações, utilizámos Word2Vec, Labeled Latent Dirichlet Allocation (L-LDA), modelos de linguagem pré-treinados e Robustly Optimized Bidirecional Encoder Representations from Transformers Approach (RoBERTa) para gerar representações semânticas numéricas das descrições. Estas representações foram usadas para classificar as aplicações com categorias definidas na Aptoide, permitindo avaliar a eficácia dos métodos. Como estamos a lidar com classificação multi-rótulo, utilizámos Classifier Chains, Label PowerSet, Binary Relevance e Multi-Label Binarizer. O nosso conjunto de dados de 9.163 aplicações foi obtido via APIs da Aptoide.

Os resultados mostram que o nosso melhor modelo de representação de texto, quando devidamente ajustado, é o RoBERTa, que apresenta as pontuações F1 mais altas nas categorias de médias micro, macro, ponderadas e de amostras. É seguido pelo modelo pré-treinado GPT-4o, que também apresenta um bom desempenho, mas fica ligeiramente atrás em comparação.

As futuras direcções de investigação incluem a integração de dados multimodais, a exploração da aprendizagem federada, a adaptação a taxonomias em evolução, o desenvolvimento de sistemas de IA interactivos e explicáveis, a realização de estudos entre línguas e culturas, a criação de modelos de categorização personalizados, a avaliação de implicações éticas, a integração com ciclos de vida de desenvolvimento de aplicações e a utilização de gamificação para aumentar o envolvimento do utilizador.

Palavras-chave: *Classificação de Múltiplas Etiquetas*, *Categorização de Aplicações Móveis*, *Modelos de Representação de Texto*, *API Integração de dados*

Abstract

This study conducts a comparative analysis of text representation and feature extraction methods for categorizing mobile applications into predefined categories. Effective categorization improves application discoverability, user experience, and application ecosystem organization.

To develop an automatic approach for categorizing mobile applications into predefined categories, we used Word2Vec, Labeled Latent Dirichlet Allocation (L-LDA), pre-trained language models and RoBERTa to generate numerical semantic representations of the application descriptions. These representations were then used to classify the apps into predefined categories. Our classification system assigned each app to the same category or categories as it appears on Aptoide, allowing us to evaluate the effectiveness of the methods. Since we are dealing with multi-label classification, we used Classifier Chains, Label PowerSet, Binary Relevance and Multi-Label Binarizer to handle label dependencies and optimize classification performance. Our dataset of mobile apps, consisting of 9,163 entries, was obtained using APIs from Aptoide.

The results show that our best text representation model, when properly tuned, is RoBERTa, which has the highest F1 scores in the micro, macro, weighted averages and samples categories. It is closely followed by the pre-trained GPT-4o model, which also performs well, but falls slightly short in comparison.

Future research directions include the integration of multimodal data, exploring federated learning, adapting to evolving taxonomies, developing interactive and explainable AI systems, conducting cross-language and cross-cultural studies, creating personalized categorization models, assessing ethical implications, integrating with application development lifecycles and using gamification to enhance user engagement.

KEYWORDS: *Multi Label Classification*, *Mobile Application Categorization*, *Text Representation Models*, *API Data Integration*

Contents

List of Figures

List of Tables

List of Acronyms

AI: Artificial Intelligence

L-LDA: Labeled Latent Dirichlet Allocation

LDA: Latent Dirichlet Allocation

RoBERTa: Robustly Optimized Bidirecional Encoder Representations from Transformers Approach

NLP: Natural Language Processing

PRISMA: Preferred Reporting Items for Systematic Reviews and Meta-Analyses

SLR: Systematic Literature Review

LLM: Large Language Model

SVM: Support Vector Machine

KNN: K-Nearest Neighbors

LR: Logistic Regression

DT: Decision Trees

RF: Random Forest

KNN: K-Nearest Neighbors

CNN: Convolutional Neural Network

BR: Binary Relevance

LP: Label Powerset

CF: Classifier Chains

TF-IDF: Term Frequency-Inverse Document Frequency

MLB: Multi-Label Binarizer

CHAPTER 1

Introduction

The exponential growth of the app ecosystem on digital distribution services is changing the way users interact with their smartphones. By 2023, Google Play hosted 3.5 million apps [1], while the App Store contained over 1.8 million [2], a significant increase from the mere 500 apps available at launch [3]. This rapid expansion underscores the evolving nature of mobile technology and consumer demands. The vast selection provides users with a variety of choices to meet different needs and preferences. But while the proliferation of mobile applications has enriched the user experience, it has also created significant challenges in terms of app discovery and organization. As the app ecosystem continues to grow, the need for effective categorization becomes paramount to facilitate seamless navigation and increase user satisfaction.

Categorization plays a critical role in managing this vast and diverse application landscape. The process involves grouping applications based on their primary functions, features and themes, allowing users to easily discover and access applications that meet their specific needs or interests. Both supervised and unsupervised learning techniques contribute to the automated categorization of applications. Supervised learning uses labeled data to train algorithms to recognize patterns and characteristics associated with specific app categories. For example, apps that provide photo editing functionality can be categorized under "Photography", while those that focus on fitness tracking can be categorized under "Health & Fitness". This approach ensures accurate and targeted categorization, making app discovery more efficient for users. Unsupervised learning, on the other hand, adapts categorization to the evolving app landscape. By identifying latent patterns in the data, unsupervised learning techniques help create new categories or refine existing ones based on user behavior and emerging app features. This dynamic approach ensures that the categorization process remains flexible and responsive to the changing dynamics of the app ecosystem.

Developers benefit greatly from the categorization process. Properly categorizing an app increases its visibility and discoverability, optimizing its chances of reaching its intended audience. This targeted exposure not only streamlines user adoption, but also aligns with the developer's marketing efforts.

In the future, advances in machine learning and artificial intelligence will further refine the categorization process. These innovations will ensure that users can effortlessly discover and engage with apps tailored to their evolving preferences. The synergy between developers, users and intelligent categorization mechanisms will continue to shape

the ever-expanding landscape of mobile applications, contributing to a personalized and enriched digital experience for all.

1.1. Context of Work

Today's mobile application landscape is characterized by an unprecedented proliferation of diverse applications, resulting in a digital marketplace that poses significant challenges for users in terms of application discovery and navigation. The exponential growth in the number of available applications has made it increasingly difficult for users to find and access applications that meet their specific needs and preferences. In response to this evolving landscape, decentralized platforms such as Aptoide have emerged to provide users with an alternative approach to finding, downloading and sharing mobile applications. Unlike traditional app stores, the decentralized nature of such platforms fosters a more dynamic and diverse ecosystem, which warrants a closer examination of their categorization mechanisms.

Using Aptoide as a case study, this research seeks to understand the intricacies of its categorization strategy. Effective categorization serves as the backbone for users to explore and discover apps that match their interests and needs. Aptoide's decentralized model requires an in-depth analysis of how its current categorization strategies impact user engagement in a rapidly evolving digital environment. This research is critical because the effectiveness of app categorization directly impacts user satisfaction and the overall user experience. To facilitate this analysis, we will use several APIs provided by Aptoide to obtain up-to-date data about the apps available on its platform. These APIs will give us access to a wealth of detailed information, including app descriptions, categories, user ratings, download statistics and other relevant metadata. By integrating data from these various sources, we will be able to conduct a comprehensive study of Aptoide's current categorization practices. This approach will allow us to assess how these strategies impact user interaction and satisfaction and explore the potential for improving app discoverability in the evolving digital landscape. In addition, by having access to both new and old data, we can make a dynamic comparison over time. This will allow us to analyze trends over time, observe how categorization strategies have evolved and determine their impact on user engagement and satisfaction. The temporal aspect of this study is relevant because it provides insight into the effectiveness of changes and improvements made to the categorization process.

A key focus of this research is the textual metadata associated with mobile applications. Textual metadata, which includes application descriptions, keywords and other relevant information, contains information that can help to understand the category to which it could belong. Leveraging this metadata is essential for developing algorithms that can accurately categorize mobile applications. This approach not only improves the discoverability of apps, but also contributes to a more intuitive and user-friendly app discovery process.

Growing concerns about app discovery and user satisfaction underscore the urgency of addressing the effectiveness of current app categorization mechanisms. Users increasingly demand streamlined and intuitive methods to discover applications that meet their preferences and needs. Ineffective categorization can lead to frustration and disengagement, highlighting the need for advanced solutions that can adapt to the dynamic nature of the app ecosystem.

In light of these considerations, the objective of this paper is to investigate the importance of textual metadata in the classification of mobile apps through the case study of Aptoide. The research analyzes the app's descriptions, keywords, and other metadata and seeks to explain their relevance in app categorization. One of the main objectives is to demonstrate a categorizing method which is suitable for employing this type of information for the purposes of effective and precise app categorization which will benefit users of decentralized app stores. Also, in the course of the study the researchers will formulate a more extensive dataset of mobile applications by integrating several data-collection APIs from Aptoide in order to take into account all notable variables. This enhanced dataset will be beneficial in achieving objectives aimed at training and testing the categorization algorithms, and also provide a basis for subsequent researches on mobile application categorization.

1.2. Research Questions

This subsection articulates the core research questions that guide our study towards understanding and optimizing app categorization through the exploration of textual metadata. The primary questions are formulated as follows:

- How to automatically categorize mobile apps based on their textual metadata?
- How does the performance of text representation models vary with the use of different multi-label classification strategies such as Label PowerSet, Binary Relevance and Classifier Chains?
- How can APIs enable dynamic comparisons of mobile app data over time to improve and maintain accurate app categorizations?

To address the first question, our research will explore into the realm of machine learning algorithms and natural language processing techniques. The objective is to develop a sophisticated classification model capable of accurately categorizing mobile apps by analyzing their textual metadata. This categorization will encompass various aspects, including:

- The overarching structure of categories of the platform, such as Games, Productivity, Entertainment, Education, etc;
- The exploration of the topics or keywords associated with the app, providing deeper insights into its functionality and purpose.

For the second question, we will investigate how the performance of text representation models varies with the use of different multi-label classification strategies such as Label

PowerSet, Binary Relevance and Classifier Chains. We will conduct experiments using these strategies and measure their impact on categorization accuracy. This will help us identify the most effective combinations for accurately classifying mobile applications based on their textual metadata.

For the third question, we will explore the use of APIs to retrieve the latest mobile app data from 2024. This approach will allow us to perform dynamic temporal comparisons between past data and newly acquired data, providing insights into how app categorizations evolve over time and ensuring that our models remain relevant and up-to-date.

In response to these research questions, our study aims to develop advanced algorithms and methods to enable an adaptive categorization system. This system is designed to evolve dynamically based on textual metadata, eliminating the need for manual intervention and allowing app ecosystems to keep pace with the ever-changing landscape of mobile applications. By addressing these questions, we aim not only to improve app categorization, but also to contribute valuable insights to the broader field of app discovery and user satisfaction within the mobile app ecosystem.

1.3. Outline of this Document

This document is divided into six chapters, the first of which is this Introduction. The subsequent chapters are organized as follows:

Chapter 2 unveils fundamental concepts related to Natural Language Processing (NLP), covering topic modeling, word embeddings, and pre-trained language models. Chapter 3 focuses on the state of the art, providing a comprehensive view that includes the research methodology and an analysis of related work from using Systematic Literature Review (SLR) methodology. This section serves as a focal point for understanding the current landscape, the main approaches and the results achieved. Chapter 4 delves into the specifics of the dataset used, offering insight into the foundational elements that underpin the research. Chapter 5 presents the experiments conducted and the results obtained. This section details the experimental setup, the performance metrics used and a thorough analysis of the outcomes to evaluate the effectiveness of different categorization approaches. Chapter 6 peers into future work, providing insight into potential trajectories and avenues for further exploration. This section also serves as a forward-looking reflection, outlining areas ripe for additional research and development.

CHAPTER 2

Concepts

The study of NLP is central to this research, as it provides the basis for understanding and exploiting textual metadata associated with mobile applications. NLP, an interdisciplinary field within Artificial Intelligence (AI), plays a key role in enabling machines to interact using human language in a meaningful and practical way. As we delve into the details of app categorization, NLP serves as an important tool for extracting valuable insights from textual data, thereby increasing the accuracy and efficiency of the categorization process.

NLP encompasses a wide range of techniques that facilitate the analysis of text and speech data. The fusion of linguistics, computer science and machine learning within NLP allows for a comprehensive approach to language understanding. In the field of app categorization, this interdisciplinary nature is particularly relevant, as it enables the development of algorithms that are able to understand and interpret the nuanced textual metadata associated with mobile applications, making it possible to accurately categorize apps based on their functionalities, features and user appeal.

As part of this research, the analysis goes beyond the basic aspects of NLP to explore advanced techniques such as topic modelling, word embeddings and pre-trained language models.

2.1. Topic Modeling

Topic modeling [4] is used to identify patterns of word co-occurrence within a collection of documents. These patterns of word co-occurrence are then conceptualized as hidden "topics" within the collection. Topic modeling has proven useful for classifying documents, easing the retrieval of information and conducting exploratory analysis on sizable collection of textual documents.

Even though computers run topic modeling algorithms on large collection of textual documents, a successful model requires preparing the text and setting algorithm parameters, like the number of topics and their distribution across the documents. Topic modeling can be applied in various domains such as social media analysis, market research and customer feedback analysis. It helps researchers and analysts gain insights into the prevalent themes and trends within a given dataset, enabling them to make informed decisions based on the extracted information. It can also aid in information retrieval tasks by providing a way to categorize and classify documents based on their thematic content. In our case, we will use it for categorizing mobile apps.

Latent Semantic Analysis, also known as LSA, is an early topic modelling algorithm introduced by Landauer, Foltz, and Laham [5]. LSA is based on the idea that words used in similar contexts tend to have similar meanings. It works by creating a matrix of word occurrences in documents and then using a mathematical technique called singular value decomposition to identify patterns and relationships between words and documents. This allows LSA to group similar documents together and extract the main topics or themes present in the collection of documents. This algorithm was an improvement on previous methods of document retrieval because it allowed users to find similar terms through the wider semantic range of the term, rather than having to search for identical words that appeared in related documents.

Blei, Ng, and Jordan [6] created Latent Dirichlet Allocation (LDA), another topic modelling algorithm. In LDA, a document is considered to be a set of observable words, where each word is assumed to be derived from an underlying, unobserved (hereafter "latent") topic. LDA measures both the distribution of a given topic across a set of documents and the probabilistic distribution of a given word across a set of topics. By analysing the patterns of word co-occurrence in a collection of documents, LDA "discovers" the topics. Each topic in the model will have a probability distribution across each document. This algorithm has been widely adopted in various fields, including social science, information retrieval and recommendation systems, due to its ability to handle large datasets and provide interpretable results.

An extension of the standard LDA model is the supervised topic model known as Labeled Latent Dirichlet Allocation (L-LDA), created by Ramage *et al.* [7]. The purpose of topic modelling is to identify abstract "topics" that appear in a set of documents. L-LDA, on the other hand, allows the modelling process to incorporate labels or categories. This can be particularly helpful if you want to focus the topic modelling process on specific topics, or if you have documents that fall into multiple categories. The goal of L-LDA is to learn a set of topics for each label, where each document is associated with one or more labels. In this model, each document is associated with one or more labels and it contains a variety of topics, each associated with a specific label.

2.2. Encoders and Decoders in Natural Language Processing

Encoders and decoders are fundamental components in the architecture of many NLP models. These components are crucial for both understanding and generating text, forming the backbone of various NLP applications. This section explores the details of text representation, particularly focusing on word embeddings and provides an overview of pre-trained language models that utilize these embeddings for various NLP tasks.

Encoders are responsible for transforming input text into a format that models can process and understand. They convert text data into fixed-size vectors that capture the semantic and syntactic information of the input. This process is essential for downstream tasks, as it allows the model to handle variable-length input data and capture meaningful patterns and relationships within the text. Encoders can be implemented using various

techniques, including recurrent neural networks (RNNs), convolutional neural networks (CNNs) and, more recently, transformer architectures.

Decoders, on the other hand, take the encoded representation and generate an output, which can be text, a classification label, or any other desired format. In machine translation, for example, the encoder processes the source language text into a context-rich representation, which the decoder then uses to generate the target language text. Decoders are also crucial in tasks like text summarization, where they produce a condensed version of the input text, or in question-answering systems, where they generate relevant answers based on the encoded input.

2.2.1. Encoders

Word embeddings are the basis of most distributed representation models. They represent words as vectors in a multi-dimensional space, where the proximity and orientation of the vectors capture the similarity and relationships between the words [8]. Word embeddings have become a fundamental tool in natural language processing tasks such as sentiment analysis, machine translation and text classification. These embeddings capture semantic relationships between words, allowing models to better understand the meaning and context of textual data.

There are two categories of word embeddings: static and dynamic [9]. Static embeddings are pre-trained on large collection of text and remain fixed, meaning they do not change based on context. The same word will have the same representation regardless of where it appears, making static embeddings suitable for tasks where word meanings are relatively stable. Dynamic embeddings, however, adapt to the context in which a word is used. They capture variations in meaning based on surrounding words, making them effective for tasks that require understanding context, such as question answering and handling words with multiple meanings.

Mikolov, Yih, and Zweig [10] introduced Word2vec, a static word embedding technique. A family of related models called Word2vec is used to create word embeddings. These models are two-layer shallow neural networks that have been trained to reconstruct word meanings. Using an extensive text collection as input, Word2vec creates a vector space with typically hundreds of dimensions, where each distinct word in the collection is given a corresponding vector. The embeddings generated by Word2vec capture both syntactic and semantic relationships between words, allowing for meaningful operations like vector arithmetic. This method gained significant popularity due to its efficiency and ability to model word similarities, greatly improving performance across various NLP tasks. Many researchers created other static word embedding models in response to Word2vec's impact on the performance of several NLP tasks. FastText from Pennington, Socher, and Manning [11] and Glove from Joulin *et al.* [12] are the most well-known examples. GloVe is a static word embedding technique that builds on the strengths of Word2vec by capturing global co-occurrence statistics of words across an entire collection. It uses the word co-occurrence matrix, which counts the frequency of each pair of words that occur

together within a given context window in the text. GloVe factors this matrix to create word embeddings, ensuring that words that frequently appear together in similar contexts have similar vector representations. This allows GloVe to capture both direct and indirect relationships between words, providing a richer understanding of their associations. FastText, is an extension of Word2vec that addresses some of the limitations of static word embeddings by incorporating subword information. Unlike Word2vec and GloVe, which represent each word as a single vector, FastText represents words as a combination of vectors for character-level n-grams (subword units). This means that FastText can decompose words into smaller components, such as prefixes, suffixes, and roots, and use these to build the final word representation.

However, the issue of word polysemy in context cannot be solved by static word embeddings, Hence, dynamic word embeddings, such as Cove (McCann *et al.* [13]), ELMo (Peters *et al.* [14]), ULMFit (Howard and Ruder [15]), etc. have been put forth.

The transformative breakthrough came with BERT (Bidirectional Encoder Representations from Transformers), a transformer-based machine learning method developed by Google [16]. Introduced in 2018 by Jacob Devlin and the Google AI Language team, BERT marks a paradigm shift in NLP. BERT's key innovation is its bi-directionality, which allows it to consider both left and right context at every level during pre-training. This capability represents a significant advance over previous models, improving the model's understanding of context and semantics. The bidirectional nature of BERT has paved the way for sophisticated NLP models, setting new benchmarks in various tasks, including question answering and language inference. The evolution from static to dynamic embeddings, culminating in the bidirectional capabilities of BERT, exemplifies the continuous progress in the field of NLP to capture and understand the intricate nuances of language. Building on the foundation laid by BERT, RoBERTa [17] was introduced as an optimized version of BERT. Developed by Facebook AI, RoBERTa improves the performance of BERT through several key changes. These include training with larger mini-batches, removing the next sentence prediction goal and training on a larger collection for a longer period of time. These improvements make RoBERTa a robust and versatile model that achieves state-of-the-art performance on various NLP tasks.

2.2.2. Decoders

In this section, we provide an overview of the pre-trained language models used to classify mobile app descriptions into predefined categories.

Pre-trained language models are AI models trained on large amounts of text and then fine-tuned for specific tasks. Given a textual input or context, these models can understand the meaning and predict the next word in a sentence, allowing them to generate coherent and contextually relevant text. By leveraging massive datasets, pre-trained language models can capture the nuances of human language, including syntax, semantics

and contextual relationships between words and phrases. This comprehensive understanding allows them to excel in a variety of NLP applications, such as text classification, sentiment analysis, machine translation and more.

Mistral-7B-Instruct-v0.3 [18], [19] is a state-of-the-art language model developed by Mistralai, consisting of 7 billion parameters. Designed for instruction-based tasks, it is particularly adept at following complex instructions and generating coherent, contextually appropriate responses. This model is well suited for applications that require detailed and structured output, such as the precise classification of mobile application descriptions. Mistral-7B-Instruct-v0.3 uses an advanced transformer architecture to enhance its understanding and generation capabilities, making it a robust tool for various NLP tasks.

Starling-LM-11B-alpha [20], developed by CallComply, is a powerful language model with 11 billion parameters. Trained on OpenChat 3.5 and using a reward model and the Advantage-Induced Policy Alignment (APA) method, this model excels in environments where nuanced language understanding is essential. Its ability to capture intricate patterns in data makes it ideal for complex linguistic tasks.

Meta-Llama-3-8B-Instruct [21] is an 8-billion-parameter model from Meta optimized for instruction-based tasks. Part of the Llama series, known for its balance of performance and computational efficiency, this model provides accurate responses to structured prompts, making it suitable for classification tasks that require detailed instructions. Its architecture supports effective scaling, ensuring robust performance in diverse NLP applications.

Phi-3-mini-128k-instruct [22] is a compact yet efficient model by Microsoft, with 3.8 billion parameters. It is designed to perform well in environments with limited computational resources while maintaining high accuracy. This model is ideal for mobile and embedded systems, where computational efficiency is as important as performance. Phi-3-mini-128k-instruct employs optimized training techniques to deliver reliable performance on various NLP tasks, making it a versatile choice for resource-constrained applications.

Phi-3-medium-128k-instruct [23] is another model from Microsoft's Phi series, featuring 14 billion parameters. It strikes a balance between performance and efficiency, making it suitable for a wide range of applications, from desktop to mobile platforms. This model is designed to handle complex tasks with moderate computational requirements. The Phi-3-medium-128k-instruct model leverages advanced neural network techniques to enhance its understanding and generation capabilities, ensuring robust performance across diverse NLP applications.

Phi-3-small-128k-instruct [24] is another model in the Phi series, featuring 7 billion parameters. This model, like its counterparts, is trained with the Phi-3 datasets that include both synthetic data and filtered publicly available website data with a focus on high-quality and reasoning-dense properties. The model, like the others from his family, has undergone a post-training process that incorporates both supervised fine-tuning and direct preference optimization for instruction following and safety measures. This ensures

that the model performs well on a variety of tasks while adhering to safety standards. Like his counterparts, it uses 128K, which is the context length (in tokens) that it can support.

GPT-3.5 Turbo [25], an advanced version of OpenAI's GPT-3, offers improvements in speed, efficiency, and the ability to handle larger context windows. Designed for high-performance natural language processing, this paid model reflects the advanced capabilities and support of OpenAI. GPT-3.5 Turbo's state-of-the-art transformer architecture ensures high-quality text generation, making it highly effective for complex and demanding NLP tasks.

GPT-4o [26] is a state-of-the-art language model developed by OpenAI that offers significant improvements over its predecessors. Designed for high accuracy and reliability in generating human-like text, GPT-4o is ideal for complex classification tasks. Like GPT-3.5 Turbo, GPT-4o is a paid model, reflecting its advanced features and performance. Utilizing the latest advances in transformer-based models, GPT-4o delivers superior performance across a wide range of NLP tasks, ensuring high quality and reliable text generation.

These pre-trained language models represent the forefront of NLP technology, each offering unique strengths that make them suitable for classifying mobile app descriptions into predefined categories. Their diverse architectures and parameter sizes cater to various computational environments, from high-performance servers to mobile devices, ensuring that robust NLP capabilities are accessible across different platforms.

2.3. Multi-Label Classification Approaches

In multi-label classification tasks, where each instance can belong to multiple classes simultaneously, various methods have been developed to handle this complexity.

Binary Relevance (BR) [27] is a straightforward method where each label is treated as a separate single-label binary classification problem. This approach transforms the multi-label problem into multiple single-label binary classification tasks. Each label is predicted independently and the final prediction consists of the union of the predictions for each label.

Label Powerset (LP) [28], [29] considers each unique combination of labels as a single label in a new space. This method transforms the multi-label problem into a multi-class problem, where each class represents a unique combination of labels. The classifier is then trained to predict these combinations directly.

Classifier Chains (CF) [30] is an ensemble-based approach that takes advantage of label dependencies. It creates a chain of binary classifiers, where each classifier predicts the presence or absence of a single label. The order of the labels in the chain is determined beforehand or by using a specific strategy, such as a randomized order or based on label correlations. The prediction for one label is then used as input for predicting the next label in the chain, resulting in a chain of classifiers.

Multi-Label Binarizer (MLB) [31] is an approach where the output labels are binarized, transforming them into a binary matrix where each column represents a class label and each row corresponds to an instance. Each cell in the matrix indicates whether a particular label is present or absent for that instance. Using MLB, the model is trained to predict the binary relevance of each label independently. However, unlike standard BR, this approach predicts the presence or absence of each label in a more integrated manner, using a single model to handle all labels simultaneously.

CHAPTER 3

State of the Art

This chapter is dedicated to presenting the current state of the art in the area of our research focus. We review the key concepts, theories and related projects that form the basis of our study. In addition, we explain the methodology used to construct this state of the art review, providing a comprehensive understanding of the existing landscape in the field. Through an exploration of relevant literature and projects, this chapter aims to contextualize our research within the broader framework of current advances and theories in the chosen domain.

3.1. Systematic Literature Review

Our research methodology revolves around a two-pronged approach using SLR [32] coupled with adherence to Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) [33] guidelines. Known for its structured and methodical approach, SLR involves systematically searching, appraising and summarizing relevant studies. This is complemented by the PRISMA guidelines, which ensure comprehensive and transparent reporting of systematic reviews and meta-analyses. By combining the rigorous processes of SLR with the structured reporting format of PRISMA, our methodology aims to enhance the credibility, clarity and transparency of our systematic review, providing a robust basis for our research efforts.

Our SLR process begins with the careful definition of the research questions, detailed in Section 1.3. These research questions act as a compass, guiding the trajectory of our exploration into the world of mobile applications and their categorization. With a solid foundation in place, we then proceeded to create a strategic list of keywords, a crucial step in navigating the relevant databases. Keywords play a key role in the efficiency of our search process, allowing us to target relevant information. This strategic selection of keywords ensures a focused search, enabling us to find the most relevant articles and resources in the vast field of mobile application categorization. Skilled use of keywords not only streamlines our search, but also saves valuable time by eliminating irrelevant search results. Since our research questions are closely related to the categorization of mobile applications, our keyword selection revolves around identifying articles in which this categorization process is comprehensively addressed. The following keywords are created:

```
categorization, classification, clustering, mobile, mobile apps,
mobile applications, tagging, apps, labeling, roberta, bert,
transformers, machine learning
```

The research, anchored in the Scopus[1] database, begins with a targeted query using some of the most relevant keywords previously identified. The query

```
"mobile apps" AND (classification OR categorization OR labeling)
AND (text OR description)
```

generated an initial pool of 130 documents. To refine the search, we narrowed the scope by restricting the "subject area" to "computer science", resulting in 96 documents. To ensure relevance, the abstract of each article was carefully reviewed. Emphasis was placed on selecting articles that contributed to the categorization or classification of mobile applications or text. Further review included an assessment of accessibility, with open access articles prioritized for inclusion. Ultimately, this process reduced the collection to 23 articles. The articles were then subjected to a thorough analysis in order to extract valuable information and insights about previous work in the categorization and classification of mobile applications or text. This analysis aimed to uncover common themes, methodologies used and notable findings or trends across the selected articles. In addition, a cascade review of the articles referenced in this selection was conducted to uncover additional information on the categorization of applications or text. The credibility and reliability of each article was paramount in this analysis, ensuring the validity of the information gathered. This comprehensive approach not only delves into the selected articles, but also extends to the broader academic context, enriching our understanding of the methodologies and trends prevalent in the field of mobile application and text categorization.

Besides the previously identified references, we also found it important to explore methods for multilabel classification. Label Powerset, Binary Relevance, and Classifier Chains were identified as potentially valuable approaches for handling multiple labels. These methods offer different strategies to address the challenge of assigning more than one label to each instance. By conducting an extensive ad hoc search of academic databases and relevant literature, we were able to identify six key articles in which these three algorithms were employed and evaluated.

Additionally, our investigation expanded to include the exploration of decoder-based Large Language Model (LLM) and their application in multi-label classification tasks. This included a comprehensive review of articles that assessed the performance of decoder-only models like GPT-3, GPT-4 and other transformer-based models in handling complex classification scenarios. Our objective was to understand how these advanced models, which generate text autoregressively, could complement or enhance text classification.

Moreover, since the Aptoide ecosystem has been used in other research, we found it important to analyze these studies as well. The research has contributed significantly to the understanding and improvement of recommendation systems, focusing on how different algorithms and models can be used to provide more accurate and user-friendly app suggestions.

[1]Scopus: https://www.scopus.com/

3.2. Related Work on the Categorization of Mobile Apps

This section provides a thorough review of the relevant literature, strategically designed to establish the contextual framework essential to the present research project. The primary objective is to position this study within the expansive landscape of scholarly research. By carefully examining the contributions of previous studies in the field, this literature review aims to identify existing knowledge gaps, establish a foundation for the investigation and extract insights that will inform and shape the approach and conceptual framework of the current study.

In order to provide clarity and structure to the complex landscape of related work, this section is divided into three distinct subsections, each focusing on a critical aspect of the research domain. The first subsection examines the methods and advances in mobile application categorization, with a particular focus on the application of topic modeling. It examines how different techniques have been used to group applications based on their features, functions and themes, providing a comprehensive overview of the evolution and current state of categorization methods. The second subsection focuses on using text vectorization and machine learning algorithms for app categorization. It highlights the effectiveness of methods like Naive Bayes and SVM, paired with vectorization techniques such as Term Frequency-Inverse Document Frequency (TF-IDF), in improving the accuracy of app classification. This section highlights the importance of these techniques in improving the accuracy and efficiency of mobile application categorization processes. The final subsection navigates through the dynamic field of word embeddings. It provides insights into how these embeddings capture semantic relationships within text and their relevance in the context of mobile app categorization. This discussion covers different embedding models and their implications for improving mobile application understanding and categorization.

3.2.1. Unsupervised Approaches

Over the past few years, research on the classification of mobile apps has gathered significant momentum, spurred by the enormous growth of app stores for mobile devices. In this context, LDA was used by Mokarizadeh, Rahman, and Matskin [34] to classify Android mobile apps. To be more precise, app descriptions (features) were modeled using LDA [6] and corresponding apps were then grouped together using K-means based on their topic models. Two datasets of Android apps were subjected to the suggested methodology. The findings showed that Google Play's default classification did not put apps on related subjects in one group. Specifically, the analysis revealed that while some categories like "News and Magazines" had high similarity between apps (44.77%), others, such as "Lifestyle," showed very low similarity (5.33%), indicating that Google Play's taxonomy system does not consistently group functionally similar apps together.

Vakulenko, Müller, and Brocke [35] employed topic modeling in order to classify comparable apps in the Apple App Store. The authors found recurring themes in app descriptions by using LDA. The apps were then divided into 66 categories, which were modified

from the categories and subcategories of the Apple App Store according to their topic models. The outcomes demonstrated that the extracted topics added to the App Store original categories and offered comprehensive analyses of the information within each one.

Nayebi *et al.* [36] explored the use of clustering techniques to enhance the analysis of mobile apps. The researchers applied the DBSCAN clustering method to a dataset of 940 open-source mobile apps from F-Droid, aiming to gain clearer insights by focusing on apps with similar characteristics. The study clustered apps based on attributes such as app size, downloads, reviews and ratings. The researchers also used LDA to group apps by their descriptions. The results indicated that DBSCAN, when applied to market attributes, was more successful at forming meaningful clusters compared to using topics extracted from app descriptions. This demonstrates that market-based data was more effective for clustering than topic modeling.

Al-Subaihin *et al.* [37] conducted an experimental comparison of text-based mobile app similarity measurement techniques and assessed how well they detected app-feature similarity. These techniques included topic modeling (LDA) and keyword feature extraction techniques [38]. The evaluation made use of 12664 descriptions of mobile apps that were downloaded from Google Play. The outcomes showed that, in terms of quantitative cluster quality, LDA-based solutions performed better than other strategies.

Gorla *et al.* [39] presented CHABADA, a technique for identifying discrepancies between the advertised and actual behavior of Android apps. The authors used Latent Dirichlet Allocation (LDA) to extract topics from app descriptions, which were then used to cluster apps using the K-means algorithm. Within each cluster, sensitive APIs requiring user permissions were identified. One-Class Support Vector Machines (OC-SVMs) were used to detect anomalies in API usage that were considered potential indicators of malicious behavior. CHABADA was evaluated on a dataset of over 22,500 Android apps, where it successfully identified several anomalies and detected 56% of new malware without relying on known malware patterns.

Che and Sun [40], developed a recommendation system based on deep learning techniques to help mobile app designers select appropriate models and features and to provide tailored recommendations based on the app's input description, reducing development barriers and increasing efficiency. The system consists of three main parts: text classification, text similarity matching and recommendation degree calculation. Techniques such as Labeled-LDA text classification and BERT text similarity matching are used to provide accurate and relevant suggestions for features and models for mobile applications on the Google App Store. The system demonstrates its potential value in helping mobile application developers make informed decisions about feature and model selection, ultimately improving the overall app development process.

Fuad and Al-Yahya [41] offer a thorough analysis of Arabic mobile apps available in the Google Play store. The authors want to know how technical and non-technical aspects of Arabic apps relate to one another as well as how well the predefined Google Play app

categories currently in place represent the different types and genres of Arabic mobile apps. The authors employed topic modeling (LDA) to examine the textual content of Arabic apps available in the Google Play store in order to accomplish their goals. They used the app description to classify the apps after extracting and analyzing data from a sample collection of Arabic apps. Additionally, the authors created a new, long-lasting categorization scheme for Arabic apps in the Google Play store that works with both new and old Arabic apps.

Cao *et al.* [42] present a novel approach to mobile application recommendation that improves both accuracy and effectiveness by combining factorization engines with a topic attention mechanism. The proposed method uses Latent Semantic Analysis (LSA) to extract global topics from mobile application descriptions and Bidirectional Long Short-Term Memory Networks (BiLSTM) to capture local semantic information. An attention mechanism further refines the model by assigning varying importance to words based on their contribution to the app's content representation. After classifying apps using these mechanisms, the factorization machine model integrates multiple app features and ranks them according to user preferences. Experimental results show that this method achieves superior performance in terms of accuracy, MAE, RMSE, and AUC compared to baseline methods, demonstrating its effectiveness for mobile app recommendation.

Al-Subaihin *et al.* [43] introduced an innovative technique for app clustering based on textual features. Their approach integrated information retrieval with ontological analysis, where features were extracted from app store descriptions. Specifically, they used NLTK's N-gram Collocation Finder to identify frequent bi- or tri-grams, referred to as "featurelets". These featurelets were then utilized to group apps using Agglomerative Hierarchical Clustering (AHC). WordNet was employed to calculate the semantic similarity between feature words. To validate their method, they mined 17,877 apps from the BlackBerry and Google Play stores. Following that, human judges assessed the clusters cohesiveness and the results demonstrated that the proposed method significantly improved the default categories provided by the app stores.

120 Apps were grouped according to their functionalities using unsupervised machine learning, as reported by Lulu and Kuflik [44]. To be more precise, professional blog content was added to app features that were taken directly from the app store descriptions. Subsequently, TF.IDF-weighted vectors of words were then used to represent the apps features. Using WordNet, synonymy relationships were resolved. The authors then created hierarchies of apps with related functionalities using hierarchical clustering.

3.2.2. Classical Supervised Approaches

Supervised machine learning approaches have been widely utilized in text classification tasks to categorize textual data into predefined classes. These methods rely on labeled datasets to train algorithms that can predict categories for unseen data based on learned patterns.

Qiu, Wu, and Zhang [45] investigated the challenge of using machine learning algorithms to categorize natural language descriptions into 61 genres. Numerous machine learning algorithms, such as Naive Bayes, SVM, Random Forestand KNN are used in the study. TF-IDF and word frequency were employed to represent the descriptions in the study. The findings demonstrate that the Naive Bayes algorithm outperforms the others in terms of performance and requires the least amount of time to train and test. With an optimized Naive Bayes algorithm, word frequency achieves an accuracy of roughly 60.276%. Additionally, the study discovers that rather than overfitting, the low accuracy in the first text is caused by the redundant dataset. Specifically, the dataset contained overlapping and ambiguous genres, such as "Music" and "Music & Audio," which led to confusion during classification. After merging redundant genres, the accuracy improved, confirming that the dataset's structure, rather than model overfitting, was the main issue causing lower accuracy.

A method for classifying mobile apps into 50 categories was presented by Berardi *et al.* [46]. The writers obtained meta-data about apps, such as their names, ratings, categories, descriptions and sizes, by searching Google Play and the Apple App Store. The extracted features were then utilized to train an SVM classifier. BM25 was used as the weighting function for each of the features that were chosen [47]. With 5,792 apps analyzed, the suggested method yielded an F1 score of 89%.

A machine learning strategy was put forth by Sanz *et al.* [48] to detect malicious apps and organize the Android market were the goals. The suggested method made use of attributes that were taken from the apps source code, along with their requested permissions and meta-data, such as size, permissions advertised and ratings. A dataset consisting of 820 apps that were selected from seven distinct Google Play categories were then classified using a variety of classification algorithms. With an Area Under the Curve (AUC) of 93%, the results demonstrated that Bayesian networks performed better than other algorithms, such as Decision Trees (DT), Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

Zhu *et al.* [49], [50] proposed an automated method for categorizing mobile apps in the Nokia Store. Their method used data from user device logs, including app usage patterns and search engine interactions, such as with Google. A Maximum Entropy model was then applied to combine these features and train an app classifier. The authors tested their method on a dataset of device logs from 443 users, covering 680 apps and found it to outperform other methods based on word vector analysis and topic modeling.

3.2.3. Embeddings-based Supervised Approaches

Ebrahimi, Tushev, and Mahmoud [51] propose a new approach for classifying mobile applications based on their app store descriptions using word embeddings. The proposed approach generates numeric semantic representations of app descriptions, which are then classified to generate more cohesive categories of apps. The method of choice makes use of GloVe [11], a count-based word embedding model. The results are validated by

analyzing a dataset that is dedicated to sharing economy apps and 12 human participants assess the results. The findings of this study demonstrate that the integration of GloVe and SVM produces app classifications that show a significant degree of agreement with human-generated classifications.

The TRedBert model, as used by Buqing Cao *et al.* [52], is a multi-modal feature fusion approach designed for mobile application recommendation and classification. It integrates image features from mobile app logos, extracted using the RedNet convolutional network [53], with text features derived from BERT. These two sets of features are combined through vector splicing, and the model uses an attention mechanism to dynamically assign importance to different modalities (image and text). This attention mechanism ensures that the model gives appropriate weight to each feature type during classification. In comparison to other models like TResbert, Redbert, Resbert, and Bert, TRedBert achieves higher accuracy and macro-F1 scores, demonstrating superior performance in mobile app classification tasks.

Qorich and Ouazzani [54] presented a Convolutional Neural Network (CNN) model that uses word embeddings to classify Amazon reviews text sentiment. The goal of the article is to discuss the difficulty of evaluating vast volumes of textual data generated by online users who are expressing their opinions. The proposed CNN model classifies sentiments of the text reviews as positive or negative and uses word embeddings to represent text data as low-dimensional vectors. The experiments findings demonstrate that the proposed CNN model with random initialization, data preprocessing and stop words vocabularies outperforms other models word embedding representations and achieves good accuracy on large-scale datasets. It had an accuracy of 90% on the Amazon dataset. They also applied the model to three other data sets: Rotten Tomatoes, Twitter and IMDB. The accuracy of these three data sets was 74.77%, 89.0% and 78.0%, respectively.

In the context of text classification using the KNN algorithm, Putra, Gunawan, and Hidayat [55] compared two feature extraction methods: word2vec and TF-IDF. The purpose of this comparison is to assess how these techniques affect computational efficiency and classification accuracy. The study's findings demonstrate that, in comparison to TF-IDF, word2vec can generate data with fewer dimensions and have higher accuracy values. With the KNN algorithm, the maximum accuracy value of TF-IDF in the 70% training and 30% testing with 8133 features was 73%, taking 312.5 seconds. In the 90% training and 10% scenario with 300 features, the Word2Vec algorithm's maximum accuracy value using the KNN algorithm was 74%, only taking 18.0 seconds.

3.3. Related Work on Multi-label Classification

This section provides a comprehensive review of the major contributions and advances in multi-label classification. The goal is to highlight the different approaches, methodologies and results proposed by researchers and to highlight the evolution of techniques in the

field. By critically examining these developments and addressing key challenges in multi-label classification, this review serves as a central resource to inform and guide future research efforts.

Amer and Elreedy [56] proposed a multi-label classification problem for driving behavior that incorporates road conditions and traffic data. To that end, the authors compare techniques such as binary relevance, classifier chains, label powerset and RAkEL. The study shows that multi-label approaches improve the performance of the classifiers, with label powerset achieving the best F-measure performance when combined with the Random Forest (RF). The experiments demonstrated the importance of feature selection in sensor-based aggressive behavior detection, improving the macro F1 measure of the SVM and Random Forest classifiers by 9.94% and 8.68%, respectively.

Yadav, Rao, and Mishra [57] propose a multi-label prediction method using decision tree-based algorithms, binary relevance, label powerset and classifier chain methods. They also address the quality of the dataset through z-score normalization and chi-square feature selection. The study demonstrated the effectiveness of the proposed method in detecting code smells with only seven features, significantly reducing the computational time and outperforming other approaches.

Won, Chi, and Choi [58] developed multi-label classification models for ground surface types using UAV-collected images. The best performing model was the BR model with the ResNet architecture, which achieved an average F1 score of 0.83. This research contributes to the construction industry by introducing a novel ground surface type model that reduces the cost and time of on-site ground surface management. The model also addresses the limitations of manual monitoring on construction sites, improving safety and accuracy.

Kumar *et al.* [59] aimed to create a generic framework for movie genre classification using problem transformation techniques, text vectorizers and machine learning classifiers. The method extracts features from movie synopses and plots and classifies movies into genres based on textual information. The best performing model, consisting of label powerset, TF-IDF and SVM, achieved an accuracy of 0.95 and an F1 score of 0.86 on the IMDb dataset.

Shah, Kumar, and Shashank [60] focus on multi-label news category. The researchers achieve high accuracy, precision and F1 scores by combining problem transformation approaches, text vectorization techniques and machine learning models. The optimal combination involved the SVM, label powerset and TF-IDF, resulting in an accuracy of 91% and an F1 score of 92%.

Nahak and Saha [61] aimed to improve the diagnosis and treatment of cardiac diseases by using hand-crafted features and the label powerset technique. The model shows promising performance in separating different classes of arrhythmias, even for sparsely occurring classes such as LBBB (left bundle branch block) and STE (ST-segment elevation). The ensemble features provide discriminative information, which improves the

classification performance. The overall F1 score for single and multi-label recordings are 88.45% and 86.32%, respectively.

3.4. Related Work on Pre-trained Language models

The advent of pre-trained language models has revolutionized several NLP tasks by leveraging massive amounts of data to learn complex patterns and relationships within text. These models have demonstrated remarkable capabilities in tasks ranging from text classification to text generation and knowledge transfer. This section reviews recent studies that highlight the applications and performance of these models in different domains, demonstrating their impact and potential improvements in specialized tasks.

Divya Venkatesh, Jaiswal, and Nanda [62] study aimed to understand the alignment between human reasoning and AI models in text classification tasks. They compared the performance and explainability of text classification performed by non-expert humans, a traditional machine learning (ML) model and a LLM using eye-tracking data. Eye-tracking data provided insight into the specific words humans fixated on during classification, offering a direct comparison with the word importance ranked by the models. The study used a domain-specific dataset of 204 injury narratives, each categorized into one of six injury cause codes. The results showed that the ML model achieved the highest overall recall (84.3%) compared to both humans and the LLM, especially for complex narratives and less distinct injury codes. Conversely, humans and the LLM performed better with less complex narratives and more unique injury codes.

Nfaoui and Elfaik [63] study evaluated the effectiveness of LLM in Arabic multi-label emotion recognition. The research aimed to assess whether LLM, specifically GPT-3.5 Turbo and GPT-4, could improve the accuracy and reliability of emotion recognition from Arabic text, particularly tweets. The study used three settings for evaluating the models: in-context learning, emotional stimuli prompts and fine-tuning. The aim was to determine if these approaches could enhance the performance of LLM in accurately classifying emotions in Arabic text. The results showed that the fine-tuned GPT-3.5 Turbo model achieved a new benchmark for Arabic multi-label emotion recognition, with an accuracy of 62.03%, a micro-averaged F1-score of 73% and a macro-averaged F1-score of 62%. The study concluded that fine-tuning and advanced prompt designs significantly enhanced the models' ability to capture the emotional context of Arabic text. The findings highlighted the potential of LLM to outperform traditional models, especially in low-resource language settings and underscored the importance of further research to optimize these models for practical applications.

Rouzegar and Makrehchi [64] aimed to improve text classification by integrating LLM and active learning strategies. The researchers aimed to reduce the financial burdens of manual data annotation by employing active learning techniques, particularly uncertainty sampling, to identify the most informative samples for annotation. They also aimed to automate the annotation process and evaluate the reliability of LLM like GPT-3.5. The study evaluated the performance of a novel framework that integrates human annotators

with LLM within an active learning context, compared the effectiveness of various annotation strategies across multiple datasets and assessed the cost-efficiency of the proposed framework in reducing annotation expenses while maintaining or improving text classification accuracy. The results showed that the proposed framework significantly reduced costs associated with data annotation while maintaining or improving model accuracy. The hybrid model demonstrated notable cost-efficiency while achieving comparable accuracies to human-only annotations. The study introduced the concept of proxy validation to estimate the model's performance at each iteration of the active learning process, providing a reliable indicator of the overall pool quality, guiding the annotation process and optimizing cost-efficiency.

Tao *et al.* [65] evaluated the effectiveness of GPT-4 in querying scientific publications related to HIV drug resistance without human intervention. The research aimed to determine the efficiency of GPT-4 in automating literature reviews and data extraction. The study used four configurations: all questions at once, all questions with an instruction sheet, individual questions and individual questions with an instruction sheet. The results showed that GPT-4 achieved a mean accuracy of 86.9%, recall of 72.5% and precision of 87.4% without an instruction sheet. The multiple-question mode was found to be more effective than the single-question mode, suggesting better performance in bulk. The study concluded that while GPT-4 showed moderately high accuracy, recall and precision, the addition of an instruction sheet did not significantly improve these metrics. This suggests the need for more sophisticated prompt engineering or fine-tuning to handle highly specialized questions in the HIV drug resistance domain. The findings underscore the potential of GPT-4 in automating systematic reviews and data extraction, but need to be addressed for higher accuracy and reliability in specific biomedical contexts.

Qiu and Jin [66] evaluated the performance of ChatGPT and a finetuned BERT model in developing intelligent design support systems. The research focused on definition classification, definition extraction and definition mapping and whether ChatGPT could enhance productivity by automating text processing and generation tasks within a specialized domain. The results showed that ChatGPT exhibited comparable performance to the finetuned BERT model in sentence-level classification tasks but struggled with short sequences. However, its classification performance improved significantly when using a few-shot setting. ChatGPT also demonstrated the ability to filter out unrelated data and enhance dataset quality by assimilating underlying domain knowledge. In terms of content generation, ChatGPT produced informative and readable output for domain-specific questions but included excessive unrelated information, which could burden readers. The study concluded that ChatGPT holds promising potential for facilitating data labeling, knowledge transfer and knowledge elicitation tasks in engineering design.

Arici *et al.* [67] evaluated the effectiveness of LLM, specifically GPT-4 and GPT-3.5-turbo, in automating IT service ticket assignment. The study focused on different prompting techniques and configurations, including zero-shot learning, few-shot learning

and ensemble learning approaches. The results showed varying levels of effectiveness across different approaches. Zero-shot learning achieved a baseline prompt accuracy of 34%, while adding more contextual information improved the performance. Few-shot learning with one example had a performance of 19% and a macro F1 score of 14%, but improved to 56% with ten examples. Ensemble learning led to better performance, with the best configuration (four prompts) achieving an accuracy of 61% and a macro F1 score of 55%. A fine-tuned BERT model outperformed the zero-shot and few-shot GPT models, outperforming the fine-tuned model. The study concluded that while GPT-4 and GPT-3.5-turbo showed promise in automating ticket assignment, the best results were achieved by using an ensemble approach combining multiple zero-shot prompts, highlighting the potential of LLM to perform well even with minimal fine-tuning in real-world applications.

3.5. Mobile App Recommendation

The landscape of mobile app and game recommendation systems has evolved significantly with the advent of advanced machine learning and deep learning techniques. In this section, we review recent studies that explore different approaches to improving mobile app and game recommendations, emphasizing the importance of personalized, efficient and context-aware suggestions to improve user experience and developer outcomes.

Coelho *et al.* [68] aimed to improve mobile app search and discovery by matching user needs, characteristics and context with available apps in the app store. The proposed approaches increase efficiency and satisfaction in app discovery, benefiting developers by improving their proximity to target consumers and increasing commercial success. The study found that character-level embeddings and fine-tuned RoBERTa models outperformed existing retrieval strategies in the database used for evaluation.

Coelho *et al.* [69], [70] delved into enhancing mobile application recommendation and semantic search systems. In their first study, they focused on leveraging semantic similarity within the *More Like This* setting of mobile app recommendation systems. Employing dense representations, they inferred application similarity based on textual fields, surpassing Aptoide's current solution. Results from a user evaluation with 1,262 participants revealed a notable 27% relevance rate for recommendations from their proposed model, contrasting significantly with Aptoide's 17%. Their subsequent work aimed at advancing semantic search in mobile app stores by utilizing transformer-based language models. They proposed a method that fine-tunes neural language models with textual data about known mobile applications, particularly focusing on application names and descriptions. This approach aimed to elevate retrieval strategies beyond lexical-based engines to more refined semantic approaches. Their results demonstrated the superiority of the proposed semantic search approach over all other documented retrieval strategies. User tests further affirmed its effectiveness compared to existing solutions, with users expressing preference for the model developed in their study.

CHAPTER 4

Data

In this chapter, we undertake a detailed exploration of the Aptoide datasets, focusing on comprehensive analyses of the 2021 and 2024 datasets and their respective subsets. Our goal is to provide a thorough overview of these datasets, highlighting their structure, content and the methods used to prepare them for subsequent analysis. We begin by explaining our approach to data management and refinement. This includes the steps taken to ensure the quality of the datasets, such as data cleaning, normalization and criteria for the inclusion and exclusion of data points. By carefully curating the datasets, we aim to ensure their relevance and reliability for our analytical purposes. To facilitate a deeper understanding of the datasets, we will use a variety of graphical representations. These visual tools will help elucidate the complexities within the data, revealing trends, patterns and behaviors that characterize the app ecosystem over time. Through these visual narratives, we aim to provide nuanced insights into the dynamic landscape of mobile apps, capturing shifts and developments from 2021 to 2024.

4.1. Aptoide App DataSet in 2021

This subsection provides a detailed overview of the Aptoide datasets collected in 2021, which include comprehensive information about the mobile applications available on the platform. The analysis uses three primary datasets to understand the attributes, metadata and categorical associations of these applications. Below, we outline the content and significance of each dataset, highlighting the key attributes that will be central to our analysis.

The 2021 Aptoide datasets include:

- Global Application Information (80Mb zip, 441MB uncompressed JSON)
  - Contains the most relevant attributes for each app;
- Individual Application Meta Data (609MB zip, 2.6GB uncompressed JSON)
  - Contains all the metadata;
- Individual Application Category (5MB zip, 126MB uncompressed JSON)
  - Contains the association between Apps and Categories.

The key difference between the first and second datasets is the level of detail: the first dataset contains only selected high-level attributes, while the second dataset contains the full range of metadata for each application. Given the detailed and comprehensive nature of the second and third datasets, we will primarily use these datasets for our analysis. The second dataset, "Individual Application Metadata", provides a rich source of information about specific mobile applications, while the third dataset, "Individual Application

Category", provides insight into the classification and hierarchical categorization of these applications.

The "Individual Application Metadata" dataset consists of 436,969 records and 22 attributes. For the purposes of this study, we focus on the most relevant attributes that provide significant insight into the attributes and characteristics of the applications:

id: Each record in the dataset is uniquely identified by an number. This serves as the unique identifier for each application.

name: The name of the application.

added:: This attribute indicates the date and time when the application was initially added to the Aptoide store.

modified: The most recent date and time when the application's listing was modified in the Aptoide store.

updated: The date and time of the most recent update to the application's content or functionality. This may differ from the "modified" date if the app was updated without changing its store listing.

pegi: The Pan European Game Information (PEGI) rating, which helps users determine the age appropriateness of the app's content.

keywords: A list of keywords associated with the application, which aids in search and discovery.

description: A textual description of the application, providing an overview of its features and functionalities.

prating.avg: The average rating given to the application by users.

prating.total: The total number of user ratings received by the application.

pdownloads: The total number of times the application has been downloaded.

The "Individual Application Category" dataset contains 427,938 records and 14 attributes. We focus on the most relevant attributes to understand the categorization of applications:

id: Each record in the dataset is uniquely identified by an id. This serves as the unique identifier for each application in the categorization context.

title: The category title assigned to the application, indicating its specific classification within the app ecosystem.

parent.title: The parent category of the application, distinguishing whether the app is an "Application" or a "Game".

4.1.1. Visualization

After merging the datasets, "Individual Application Metadata" and "Individual Application Category", we consolidated the data into a single dataset consisting of 427,938 entries. It is noteworthy that this merging process resulted in a reduction of 9,031 entries due to the absence of category information for certain applications in the "Individual Application Category" dataset. This reduction highlights the relevant role of categorical information

in ensuring comprehensive analysis and insight into the characteristics and classifications of mobile applications.

The goal of this merging process was to enrich the metadata by integrating categorical information to provide a richer context for analyzing the applications. By enriching the dataset, we aim to facilitate a more detailed understanding of the application attributes and their classification within the Aptoide ecosystem.

Our analysis revealed the presence of 27,044 duplicate entries within the dataset, corresponding to 12,914 unique applications. These duplicates indicate repetitive or redundant information, which could be due to various factors such as data collection methods, the merging process, or inherent inconsistencies within the datasets. A closer examination of these duplicate entries revealed instances where the same application was categorized under multiple classifications. For example, the application "Underwater Treasures: Cogwheels and Submarines" appeared as both a "Arcade" and a "Simulation", demonstrating the multifaceted nature of certain applications and the complexity of categorization.

Figure 4.1 shows the distribution of apps based on the number of categories to which they are assigned. The data shows a steep decline in the number of apps as the number of categories increases. Over 400,000 apps are associated with just one category, making this the most common scenario by a wide margin. As the number of associated categories increases, the distribution drops. For apps with two categories, the number drops sharply to about 11,917. Apps associated with three categories are even fewer, with a total of only 828. Beyond that, the numbers continue to drop rapidly, with only 135 apps assigned to four categories and only 23 apps assigned to five categories. The trend is even more pronounced for apps with six or more categories: only nine apps have six categories, and only one app each has eight and nine categories.

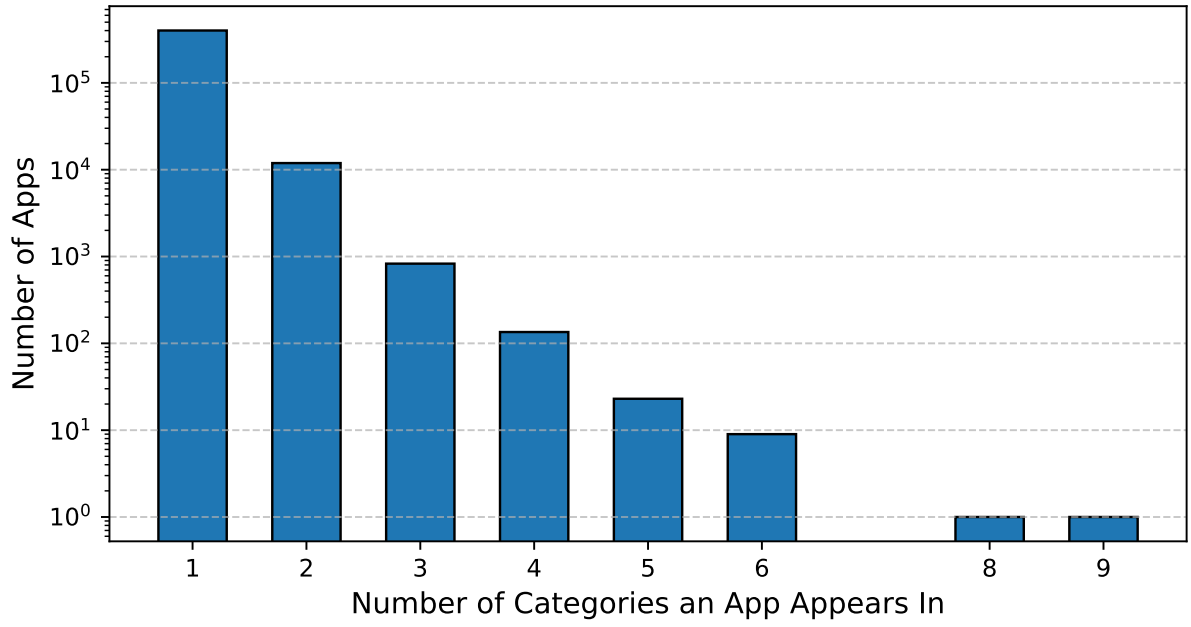

FIGURE 4.1. Number of Categories an Apps in 2021

An in-depth analysis of the categorical distribution of the dataset revealed significant insights into the composition of mobile applications:

- **Games vs. Applications**: 344,456 of the entries were categorized under "Games", while a substantial majority, 83,482, fell under the broader "Applications" category.
- **Subcategory Analysis**: Within the "Games" category, 21 distinct subcategories were identified, with "Puzzle" and "Casual" emerging as the most prevalent. These genres dominate the gaming landscape, highlighting their popularity among users. Conversely, the "Applications" category encompassed 44 subcategories, with "Education" and "Tools" being the most prominent. This diversity underscores the wide range of functionality and user needs addressed by applications in this category.

These findings are illustrated in Figure 4.2, which provides a visual representation of the number of apps by category in 2021. The comparison of app categories between 2021 and 2024, shown in Figures 4.3 and 4.4, further illustrates the evolving trends in app categorization over time.

A notable observation from the data analysis was the high prevalence of applications with a rating of 0, which accounted for 83.27% (344,586) of the entries. However, the majority of rated applications received a rating of 5, with an average rating of 4.21, as shown in Figure 4.5. This finding suggests a wide range of user experiences and highlights the variability in user engagement and satisfaction with applications.

In addition, we observed that a subset of applications, totaling 42,372 entries (10.24%), were not downloaded at all. Of the applications that were downloaded, the majority had fewer than 100 downloads, as shown in Figure 4.6. This indicates a relatively low level of user engagement or visibility for a significant portion of the applications in the marketplace.

Our analysis also revealed that 668 applications (0.161%) had no associated keywords. For those applications with keywords, the majority fell within a range of 4 to 6 keywords, as shown in Figure 4.7. This consistency suggests a standardized approach by developers in describing and classifying their applications.

In addition, only 0.000938% (388) of the submissions lacked descriptive content. Most descriptions were relatively short, with a significant proportion containing less than 200 characters, as shown in Figure 4.8. This brevity may affect how effectively users can understand the purpose and functionality of the application.

Analysis of the PEGI ratings, shown in Figure 4.9, revealed that a significant proportion of applications were rated PEGI-3, indicating that they are suitable for all ages. This was closely followed by PEGI-12 classifications, which are appropriate for users aged 12 and over. This distribution reflects a prevalence of content aimed at general audiences and young adolescents.

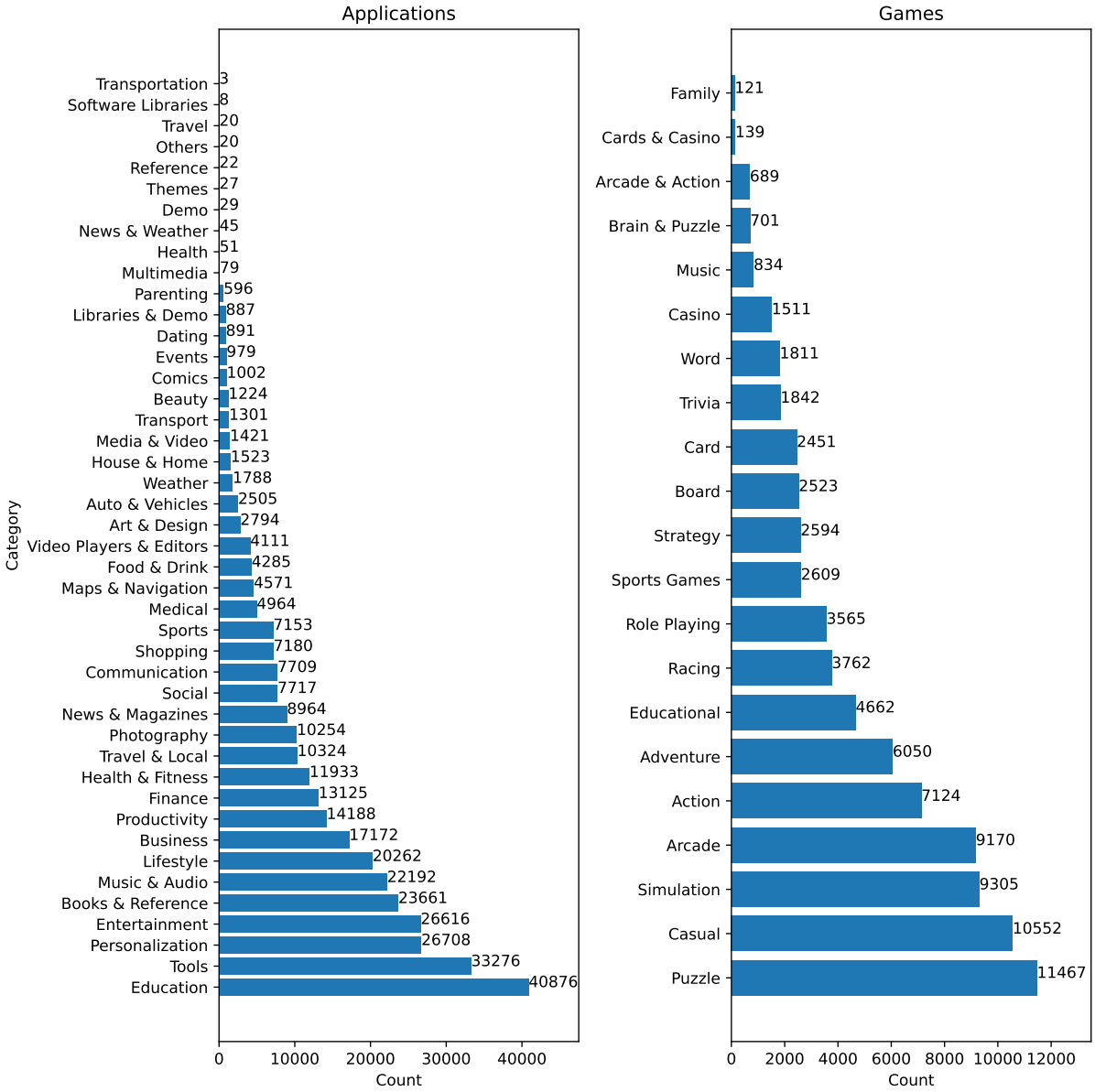

FIGURE 4.2. Number of Apps By Category in 2021

4.1.2. Subset Containing the Most Relevant Apps

To create a smaller subset with the most relevant apps, we applied specific rules to streamline the dataset and ensure consistency.

First, we removed duplicates based on the criterion of retaining only one entry per application. This decision was based on the observation that the primary distinction between duplicate entries was their assigned categories. By consolidating each application into a single entry, we aimed to streamline the dataset while preserving the diversity of categorizations associated with each application.

Following this initial phase of data cleaning, which will be discussed in more detail below, we reintegrated the removed duplicates back into the dataset. This ensured that the dataset retained a comprehensive representation of all the categories associated with

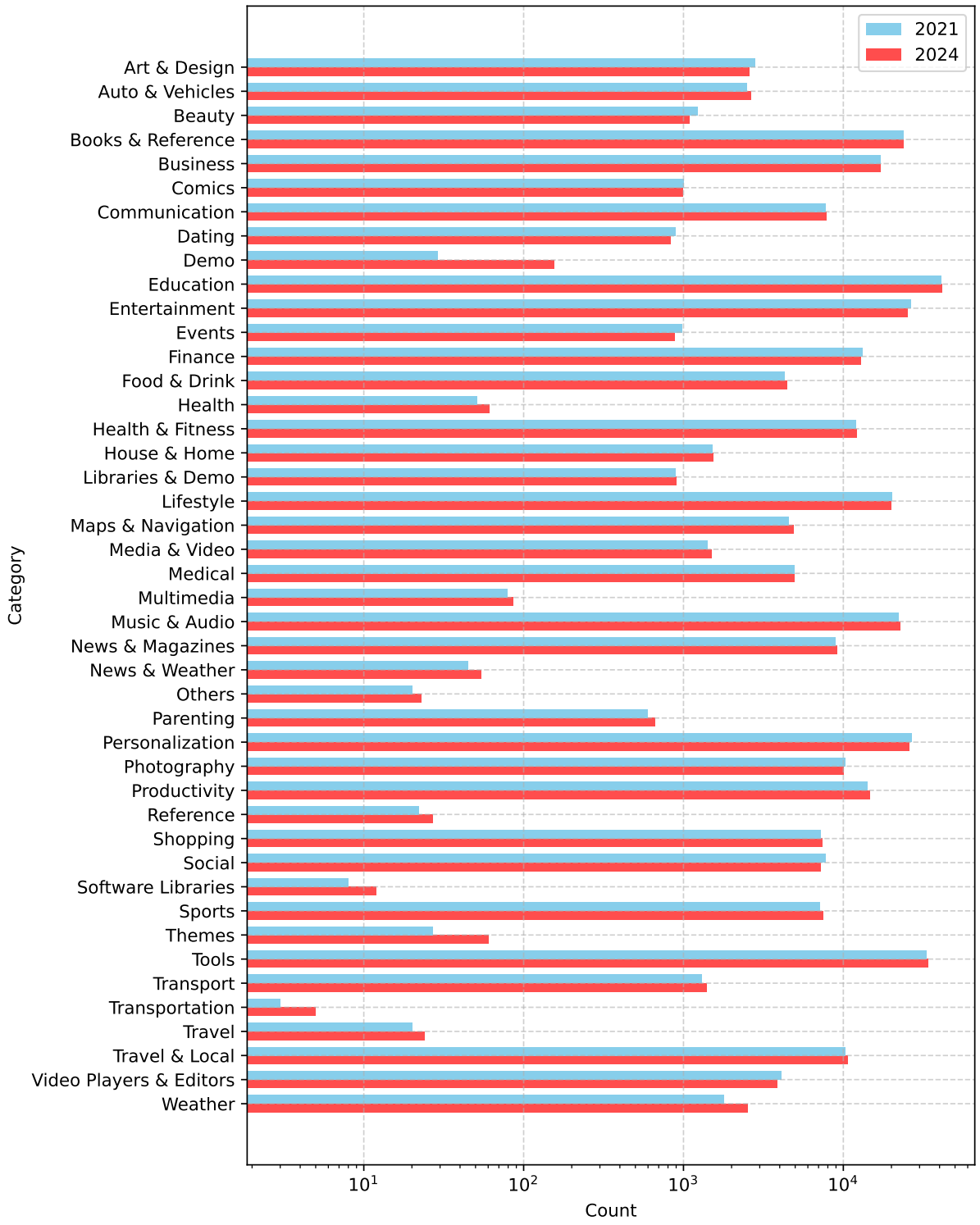

FIGURE 4.3. Number of Apps By Category in 2021 and 2024 (Applications)

each application. By taking this iterative approach, we mitigated the risk of bias from inconsistent categorizations across duplicate entries, thereby improving the overall integrity and utility of the dataset.

To narrow our focus to the most relevant applications, we applied the following five heuristics to identify key applications:

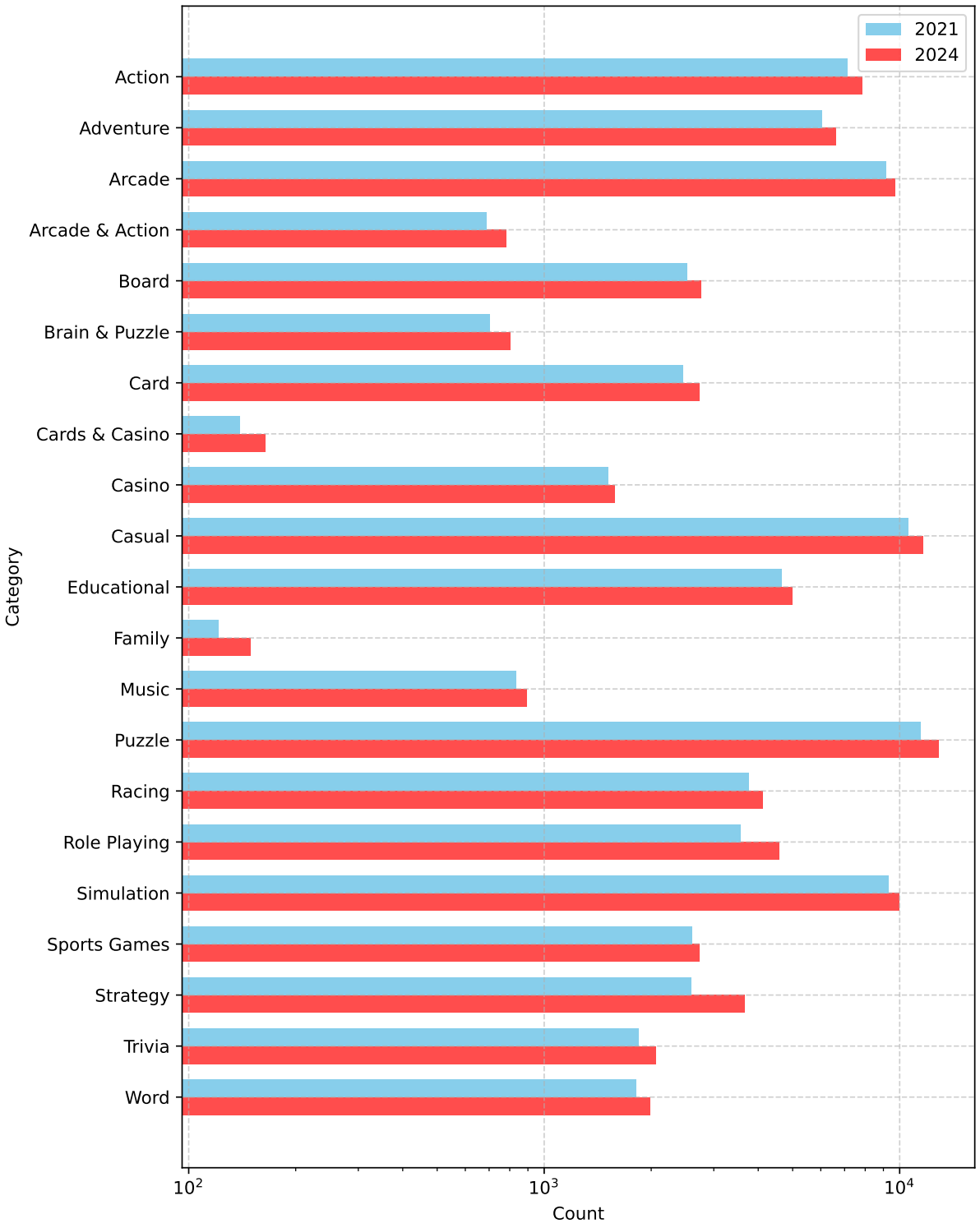

FIGURE 4.4. Number of Apps By Category in 2021 and 2024 (Games)

(1) **Top 7500 downloads with recent updates (last 2 years):** These applications were selected based on their high number of downloads and recent updates within the last two years. This criterion ensures a focus on applications that remain active and relevant over time.

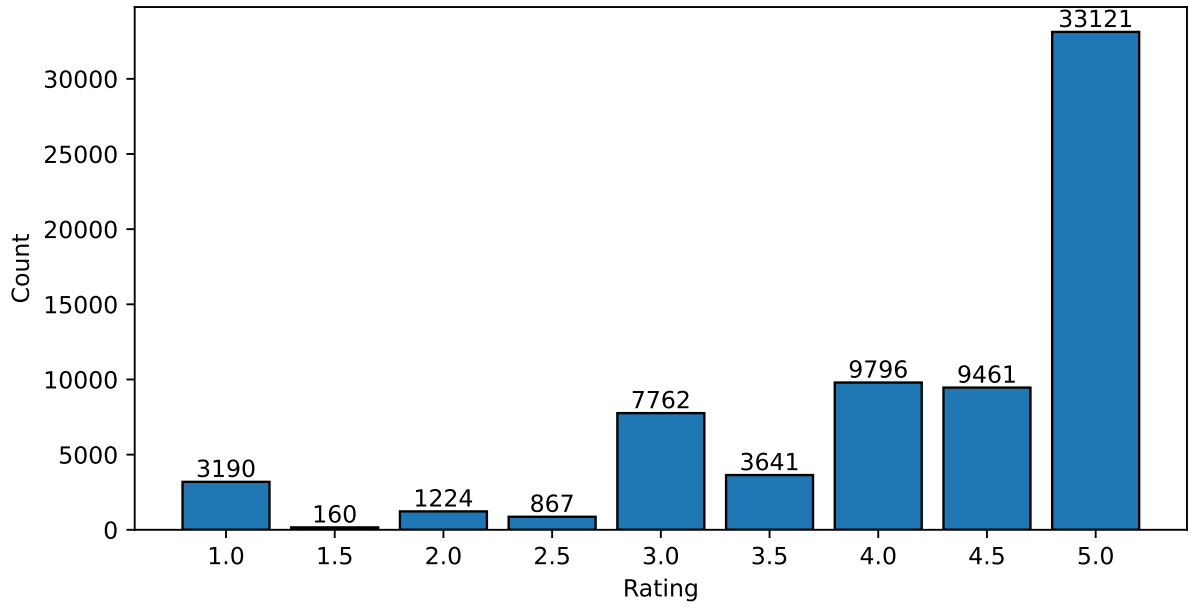

FIGURE 4.5. Rating Vs Frequency in 2021

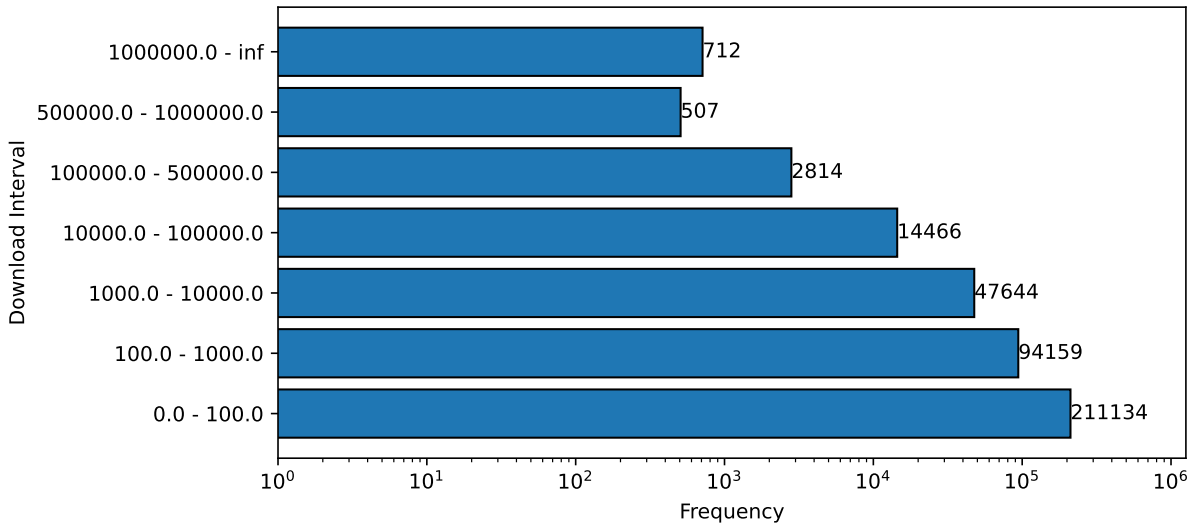

FIGURE 4.6. Download Intervals Vs Frequency in 2021

(2) **Top 7500 downloads with recent updates (last 6 months):** Similar to the previous criterion, this selection identifies applications with a high number of downloads and recent updates within the last six months. This shorter timeframe captures applications with recent activity and potential improvements.

(3) **Top 2500 rated applications with significant user engagement:** This category prioritizes applications with high user ratings. The top 2500 applications were selected based on their ratings, with a minimum requirement of 200 ratings and 1000 downloads, indicating significant user engagement.

(4) **Top 2250 applications with a recent spike in ratings (last 3 months):** This section focuses on applications that have seen a recent spike in user ratings.

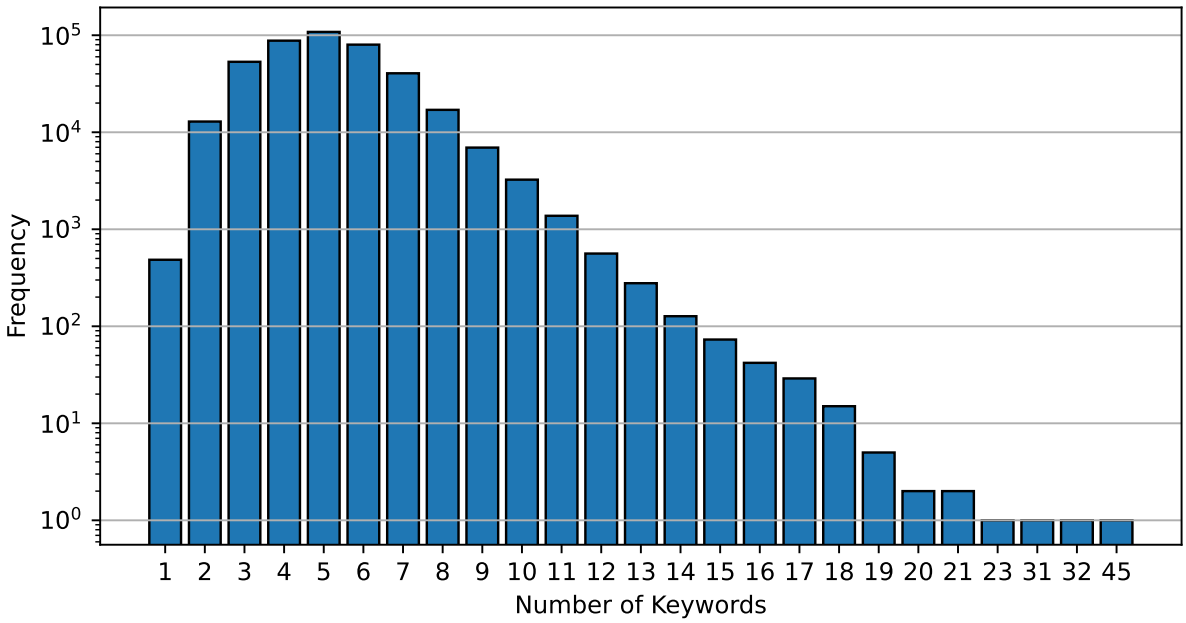

FIGURE 4.7. Number Of Keywords VS Frequency in 2021

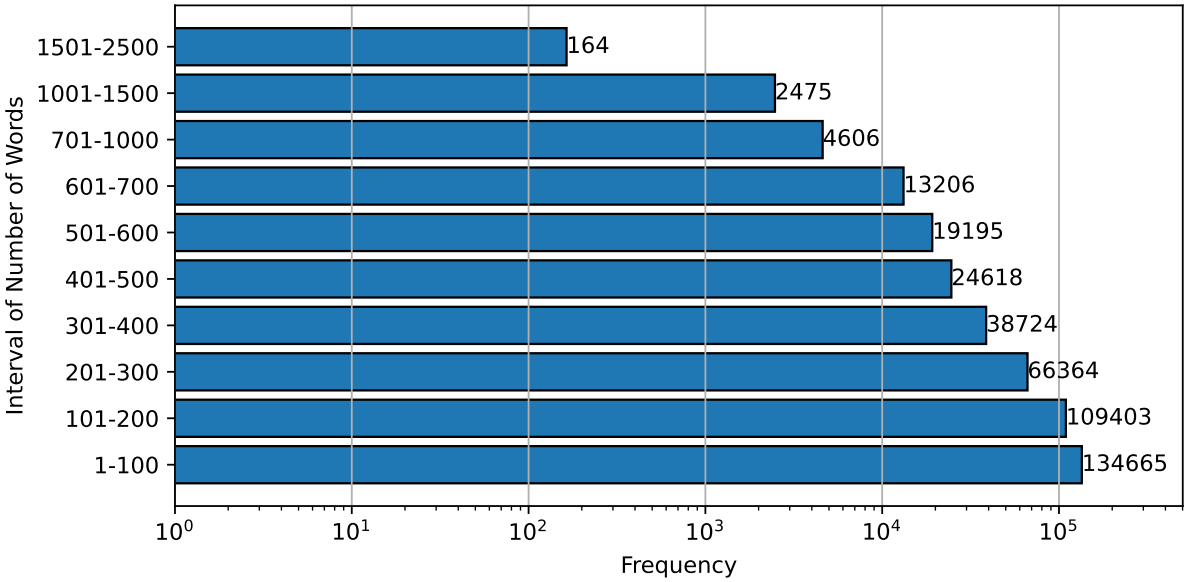

FIGURE 4.8. Words In Description VS Frequency in 2021

The top 2250 applications with the highest number of new ratings were selected, taking into account applications added in the last 3 months and having at least 1000 downloads.

(5) **Top 3000 Applications with the highest monthly downloads:** This category highlights applications that have had consistently high monthly downloads since they were added to the store. The top 3000 applications were selected based on their consistently high monthly download rates.

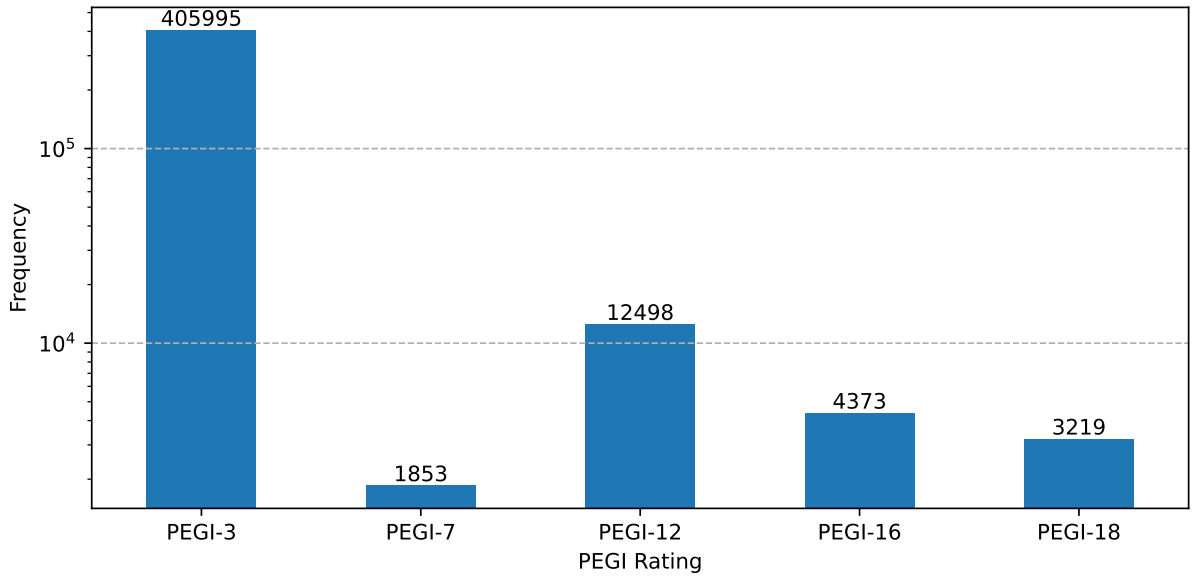

FIGURE 4.9. PEGI Rating VS Frequency in 2021

After implementing these heuristics, which generated a dataset of 10,484 applications, we further refined our dataset by filtering out applications with non-English descriptions using advanced language detection techniques as described by *papluca/xlm-roberta-base-language-detection · Hugging Face — huggingface.co* [71]. The model used for this task is a variant of the XLM-RoBERTa [72] architecture, a transformer-based model enhanced with a classification head. This classification head, a linear layer, is superimposed on the pooled output of the transformer and has been fine-tuned on a language identification dataset consisting of text sequences in 20 different languages. The dataset contains 70,000 samples for training, with 10,000 instances each for validation and testing, achieving an accuracy of 99.6% on the test set. Prior to fine-tuning, the XLM-RoBERTa model was pre-trained on a large collection of filtered CommonCrawl data, totaling 2.5 terabytes and covering content in 100 different languages, as described by Conneau *et al.* [73]. From the 10,484 applications, only 9,729 applications had English descriptions.

To ensure comprehensive coverage, we then reintegrated the previously removed duplicate entries into the dataset. These duplicates represent applications assigned to more than one category. This merge resulted in a final dataset of 10,992 entries. Notably, this inclusion of duplicates expanded the dataset by 1,263 entries to account for instances where an application appeared in multiple categories.

In the analyzed subset of data from 2021, by looking at Figure 4.10 it is evident that 4,936 belong to the "Games" category, which includes 21 different subcategories. Conversely, the remaining 6,056 entries are classified under "Applications", which includes 39 different subcategories. This distribution underscores the significant presence of both gaming and utility-oriented applications within the dataset, being also worth mentioning that an app can appear in both categories. A closer look reveals that certain categories dominate within each domain. Within the "Applications" domain, "Entertainment" and

"Tools" emerge as the dominant categories, reflecting the widespread user demand for leisure-oriented and practical utility applications. Conversely, within the "Games" category, "Simulation" and "Action" are the leading classifications, highlighting the continued popularity of these genres among gaming enthusiasts, as shown in Figure 4.10. Compared to the original dataset, only the "Tools" category has maintained its position within the top two categories. Applications in the other three categories did not adhere to the heuristics used in the data selection process.

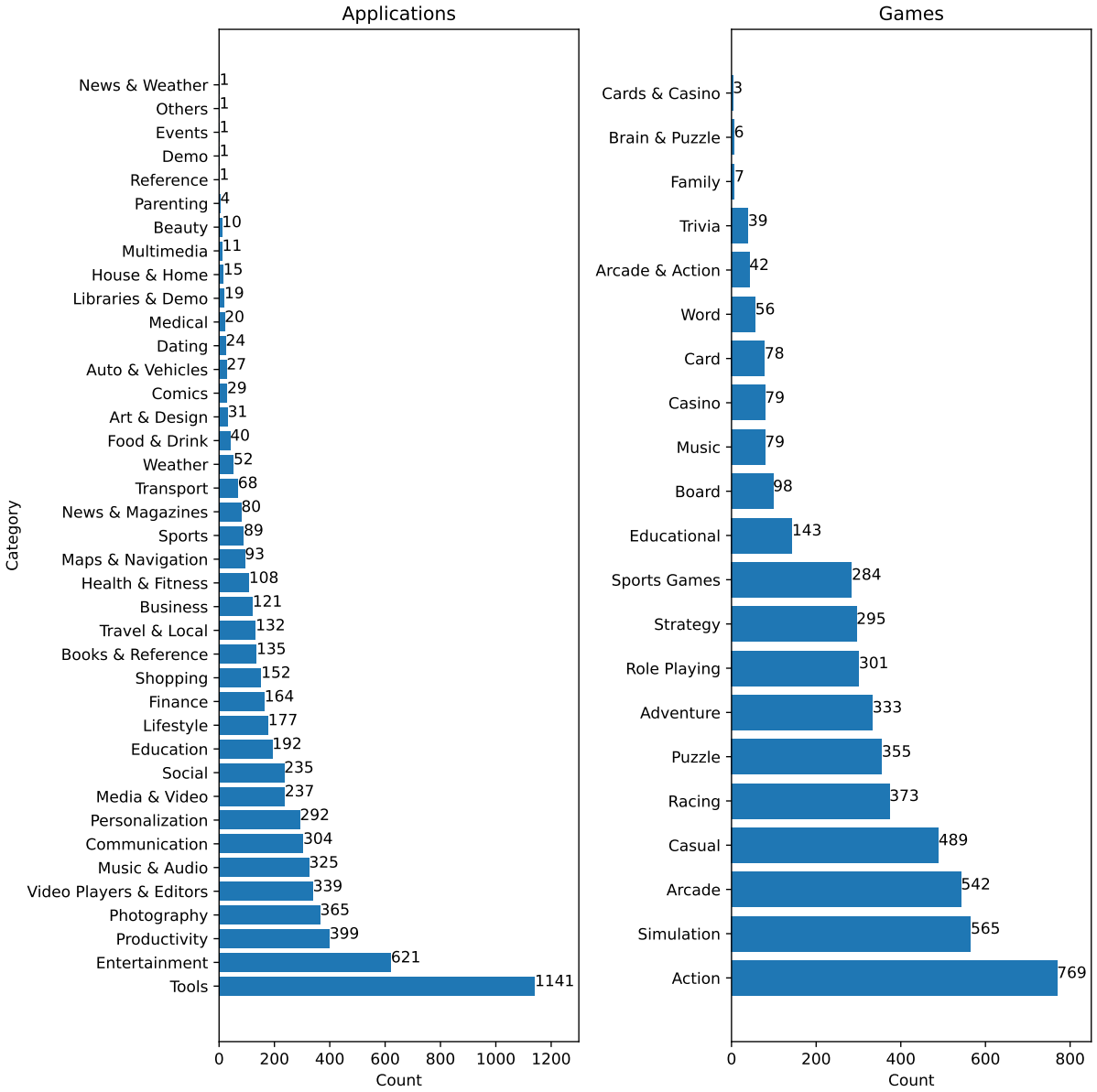

FIGURE 4.10. Number of Apps By Category in Subset 2021

Figure 4.11 shows the distribution of PEGI ratings within the parent categories of "Games" and "Applications". There is a dominant trend in both categories, with the majority of applications being rated PEGI-3, indicating suitability for all ages. PEGI-12 also represents a significant proportion of applications in both areas. However, there are noticeable differences when comparing the "Games" category with the "Applications"

category. There is a noticeable increase in the presence of PEGI-7 and PEGI-16 rated applications within the "Games" category. This phenomenon can be attributed to the inherent nature of games, which are often aimed at a more mature audience than typical applications. The inclusion of content designed for older age groups is in line with demographic preferences and expectations within the gaming sector, where themes, gameplay and interactions may require a higher age rating.

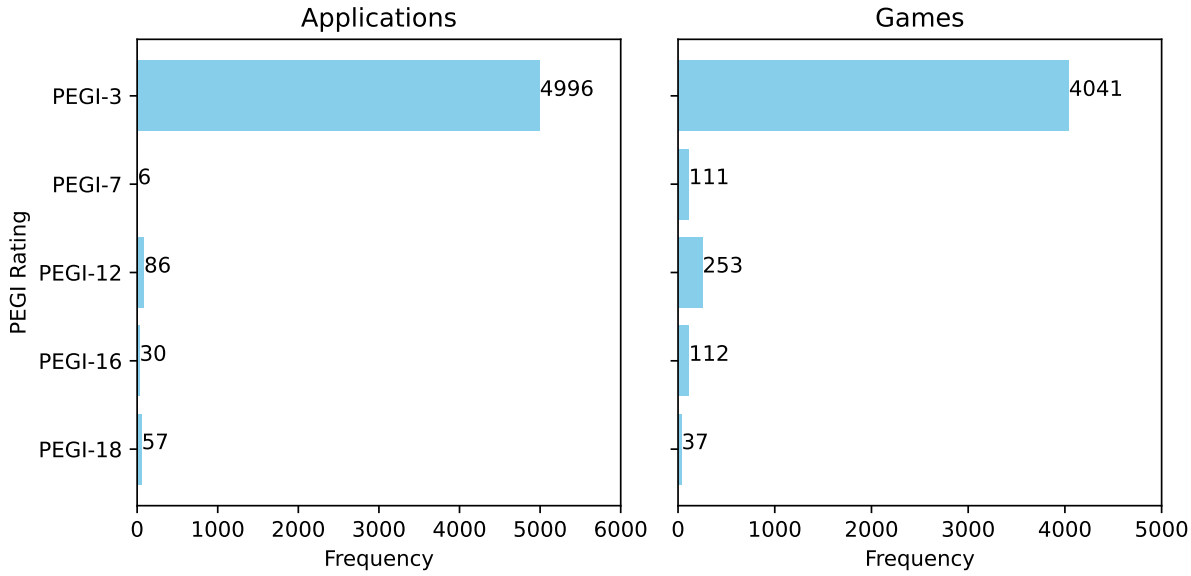

FIGURE 4.11. PEGI Rating By Parent Category in Subset 2021

Figure 4.12 shows the average number of keywords for "Games" and "Applications" across different categories. Upon inspection, it is clear that the average number of keywords across categories is consistently around five, regardless of the parent category. This stability suggests a balanced approach to keyword usage, ensuring relevance and comprehensiveness without overwhelming app descriptions. This balance indicates an optimized strategy aimed at improving app discoverability and facilitating efficient user search and exploration within the digital marketplace.

A detailed analysis of the keywords associated with each category reveals a striking correlation: the category name often corresponds to the keyword with the highest frequency. This correlation reflects a semantic cohesion in which the essence of the category is encapsulated in its most common keyword. This alignment suggests a deliberate and meaningful association between category names and dominant keywords, implying a harmonious representation of application content. Table 4.1 provides illustrative examples of this correlation.

Figure 4.13 presents a comprehensive analysis of the average number of words in descriptions across categories. The average word count for Applications is 298 words, surpassing that of Games, which averages 287 words. Within the "Games" category, all subcategories have an average word count of over 200 words. In contrast, within the

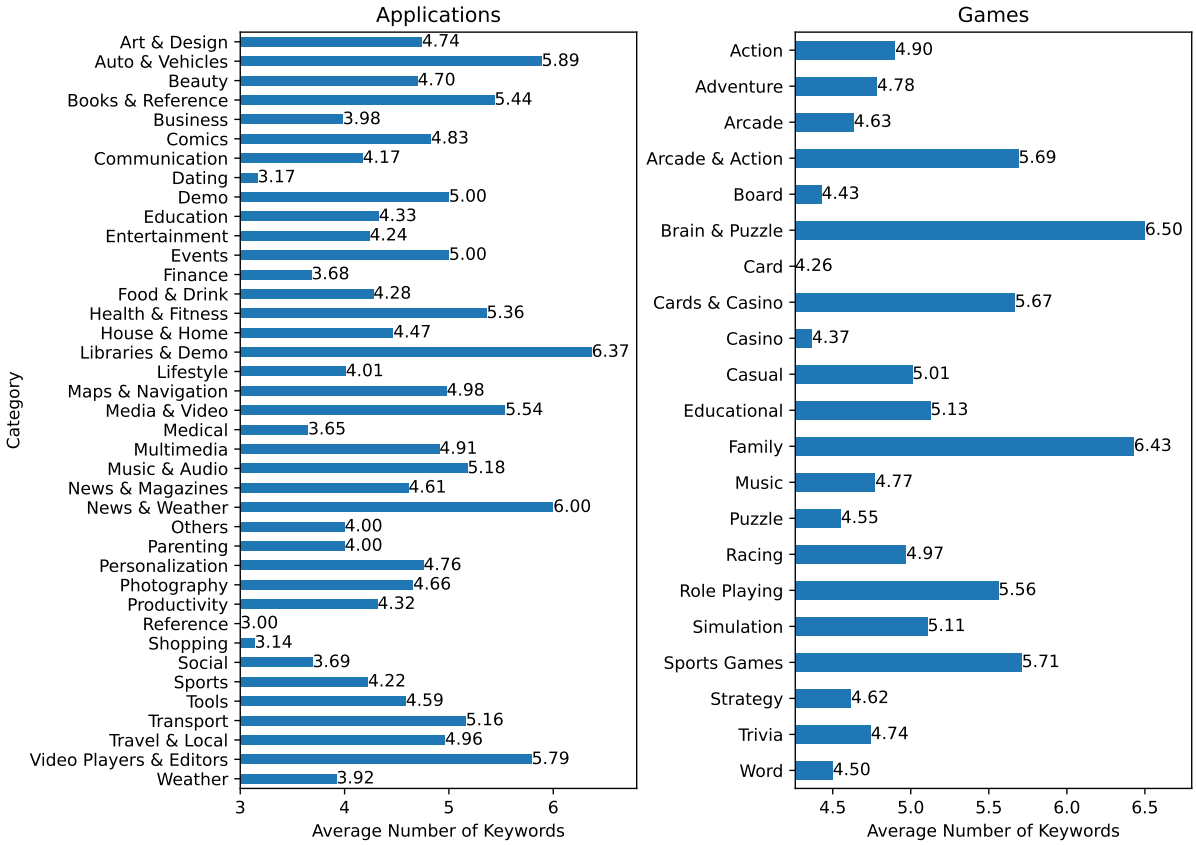

FIGURE 4.12. Average Number of Keywords By Category in Subset 2021

TABLE 4.1. Examples of Categories and Associated Keywords

Category	**Keywords**
Art & Design	design, android, photo
Auto & Vehicles	auto, vehicles, android
Beauty	beauty, camera, makeup
Books & Reference	books, reference, android
Action	action, zombie, game
Adventure	adventure, pigrat, pokemon
Arcade	arcade, action, game
Arcade & Action	arcade, action, lego

"Applications" category, there are two subcategories with an average of less than 100 words and six subcategories with less than 200 words.

4.1.3. Subset Updated with Data from 2024

After analyzing the trends in the 2021 dataset, we extended our comparative analysis to include updated data from 2024. This update was facilitated by an API[1] that allowed the extraction of the apps present in the 2021 dataset. Our script successfully retrieved data for 8,478 of the 9,729 records. Of the remaining 1,251 items, 1,239 returned a 404 error, indicating that they were no longer available in the Aptoide store. Additionally,

[1] https://ws75.aptoide.com/api/7/app/getMeta/package_name={package_name}

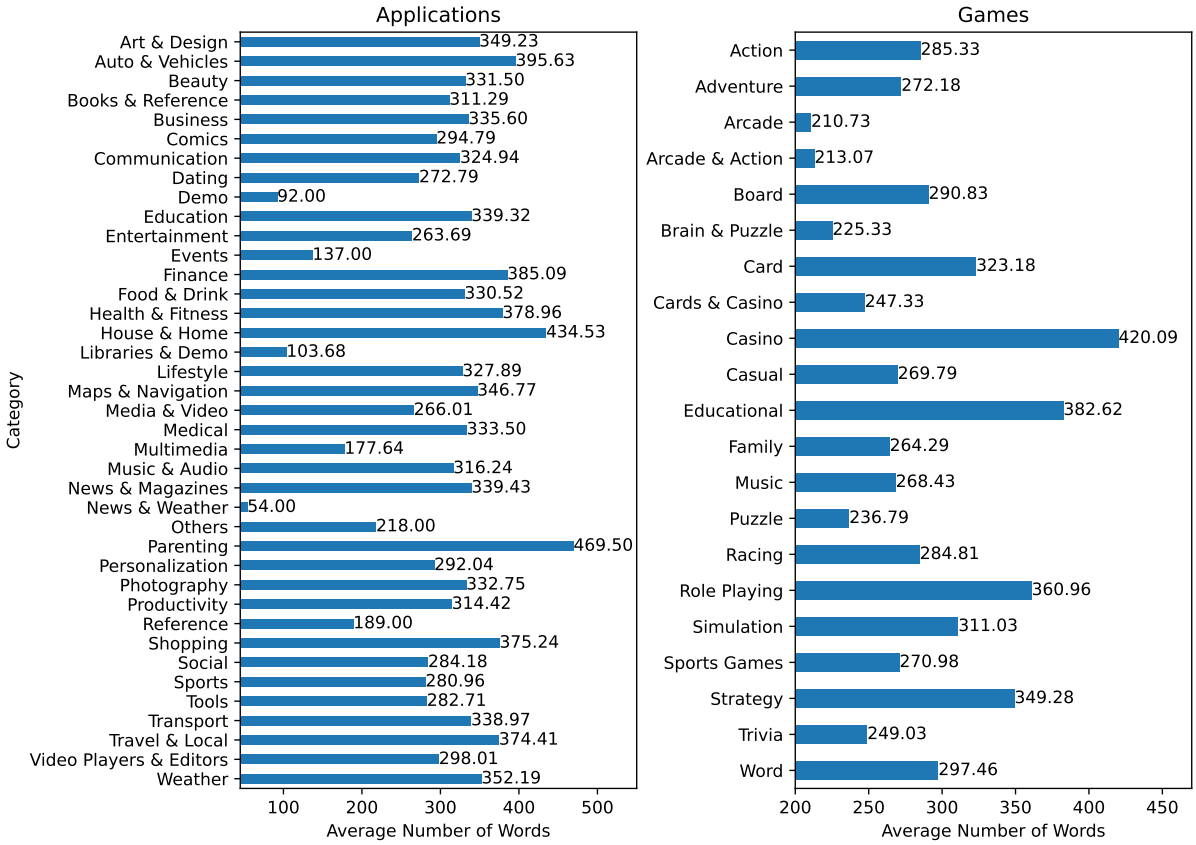

FIGURE 4.13. Average Number of Words in Description By Category in Subset 2021

8 items returned a 401 error due to access permissions and 4 items returned a 410 error indicating that the resource had been permanently removed.

Although the dataset we obtained contained comprehensive information, it lacked details on the category and parent category of each application. To address this, we supplemented our data with a comprehensive dataset of all applications available in 2024, including their categorizations. This additional dataset, which we will discuss in the section 4.2, allowed us to map apps to their respective categories, resulting in a consolidated dataset of 9,808 entries. This increase includes 1,330 additional entries due to some applications being classified in multiple categories simultaneously. For example, "Waze Navigation & Live Traffic" was categorized under both "Maps & Navigation" and "Travel & Local".

In our analysis, as previously mentioned, we identified 1,251 2021 applications that were not available in the 2024 dataset. A significant portion of the missing games belonged to genres such as "Action", "Adventure", "Arcade" and "Strategy", while the majority of the missing applications fell into categories such as "Tools" and "Entertainment".

Looking at the changes in ratings between 2021 and 2024, we found that 2,444 applications experienced an increase in ratings, 2,180 applications experienced a decrease and 3,854 applications remained unchanged. Figure 4.14 illustrates these rating differences. The left graph shows the distribution of rating changes for all apps, with a notable

concentration of apps experiencing modest changes in the -1 to 1 range. A significant number of applications also experienced rating increases, with values ranging from 0 to 5. Narrowing our analysis to apps with at least 10 more ratings in 2024 than in 2021 reduces the dataset to 1,169 apps. The right graph in Figure 4.14 focuses on this subset and shows that there are no apps with rating increases of 5 or more. However, the majority of apps continue to have rating changes in the -1 to 1 range.

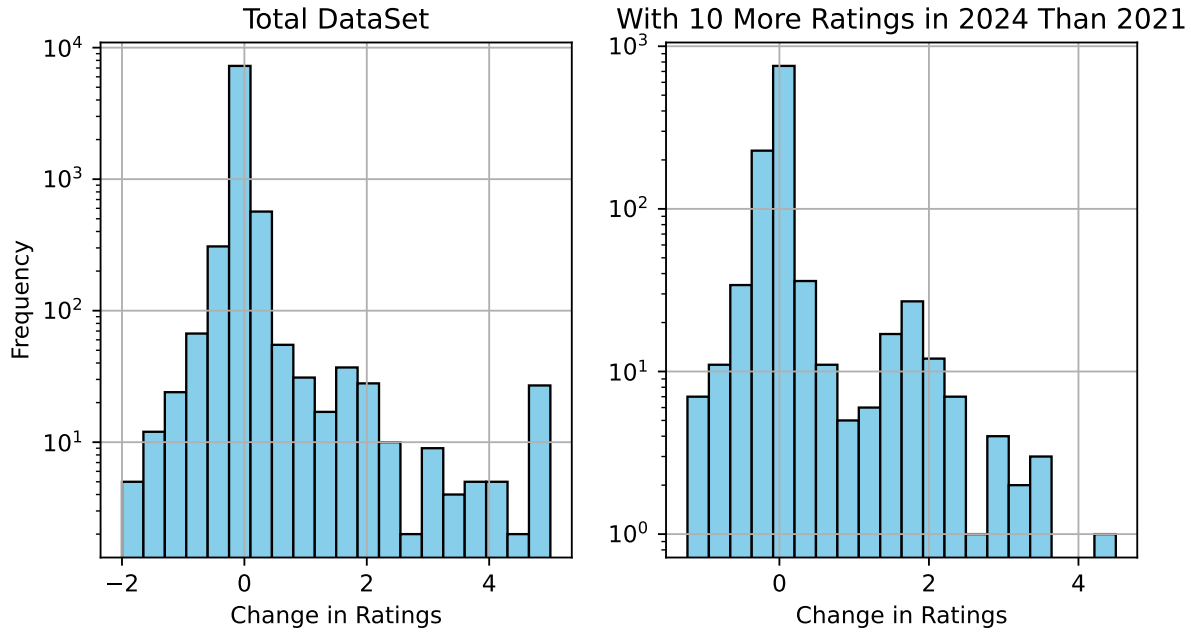

FIGURE 4.14. Comparing Ratings between 2021 and 2024

Figure 4.15 shows the average change in ratings for both "Games" and "Applications" by category. In the "Applications" category, out of 38 categories analyzed, 12 experienced a decrease in ratings, 2 remained the same and 24 experienced an increase. The average change in ratings ranged from -0.34 to 0.25, indicating different rating fluctuations across application categories. In the "Games" category, however, the analysis reveals a more nuanced pattern. Of the 21 categories evaluated, 5 showed a decrease in ratings, 3 were unchanged and 13 showed an increase. The average change in ratings for the "Games" categories ranged from -0.25 to 0.17, suggesting a more clustered distribution compared to the "Applications" category.

Figure 4.16 shows an analysis of the average percentage change in downloads for both "Games" and "Applications". Within the "Applications" category, a wide range of variation was observed across different genres. Notably, the "Reference" genre saw the smallest increase at 10%, while the "Social" category experienced a remarkable 321% increase in downloads. This substantial increase in the "Social" category is heavily influenced by the "TikTok for Android TV" application, which saw an extraordinary 871.69% increase in downloads, skewing the average upward. Excluding this outlier, the increase in the "Social" category drops to a more modest 40.70%. The average increase in downloads across all "Applications" is 61.31%, including the exceptional performance of TikTok.

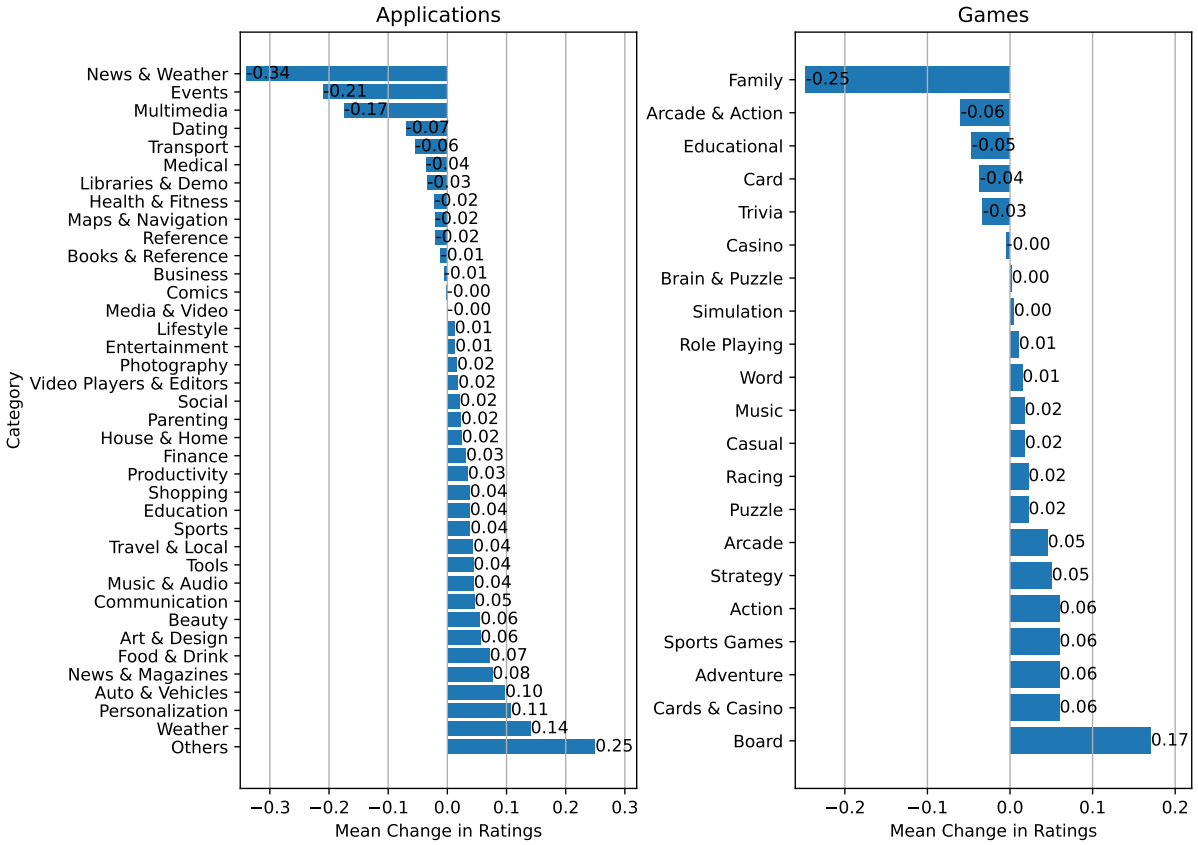

FIGURE 4.15. Comparing Ratings by Category between 2021 and 2024

Excluding TikTok, the average rises to a more representative 49.19%. In the "Games" category, we can see that "Family", despite an increase in downloads, experienced the largest drop in average ratings within the "Games" segment.

In the "Games" category, we see a different pattern with a narrower range of variation in download statistics. For example, the "Card & Casino" genre shows the smallest increase at 8%, while the "Family" genre leads with a 58% increase. This significant growth in the "Family" genre suggests increased user engagement and interest. Despite the variation across genres within the "Games" category, the overall average increase in downloads is 29.22%, which is significantly lower than the increase observed in the "Applications" category. This disparity in growth rates highlights potential differences in user behavior, preferences and market dynamics between gaming and non-gaming applications.

Our analysis shows that 1,048 applications, representing approximately 12.36% of the dataset, underwent a change in PEGI classification. Figure 4.17 provides a visual representation of these transitions. In the "Applications" category, the majority of transitions were from PEGI-3 to other classifications. Transitions to PEGI-12, PEGI-18, PEGI-16 and PEGI-7 were observed, with a significant increase in shifts to PEGI-12. Applications transitioning to PEGI-12 were predominantly in the "Entertainment" genre, suggesting a shift towards content for a slightly older audience. Transitions to PEGI-18 were more common in the "Social" and "Dating" genres, indicating content unsuitable for younger

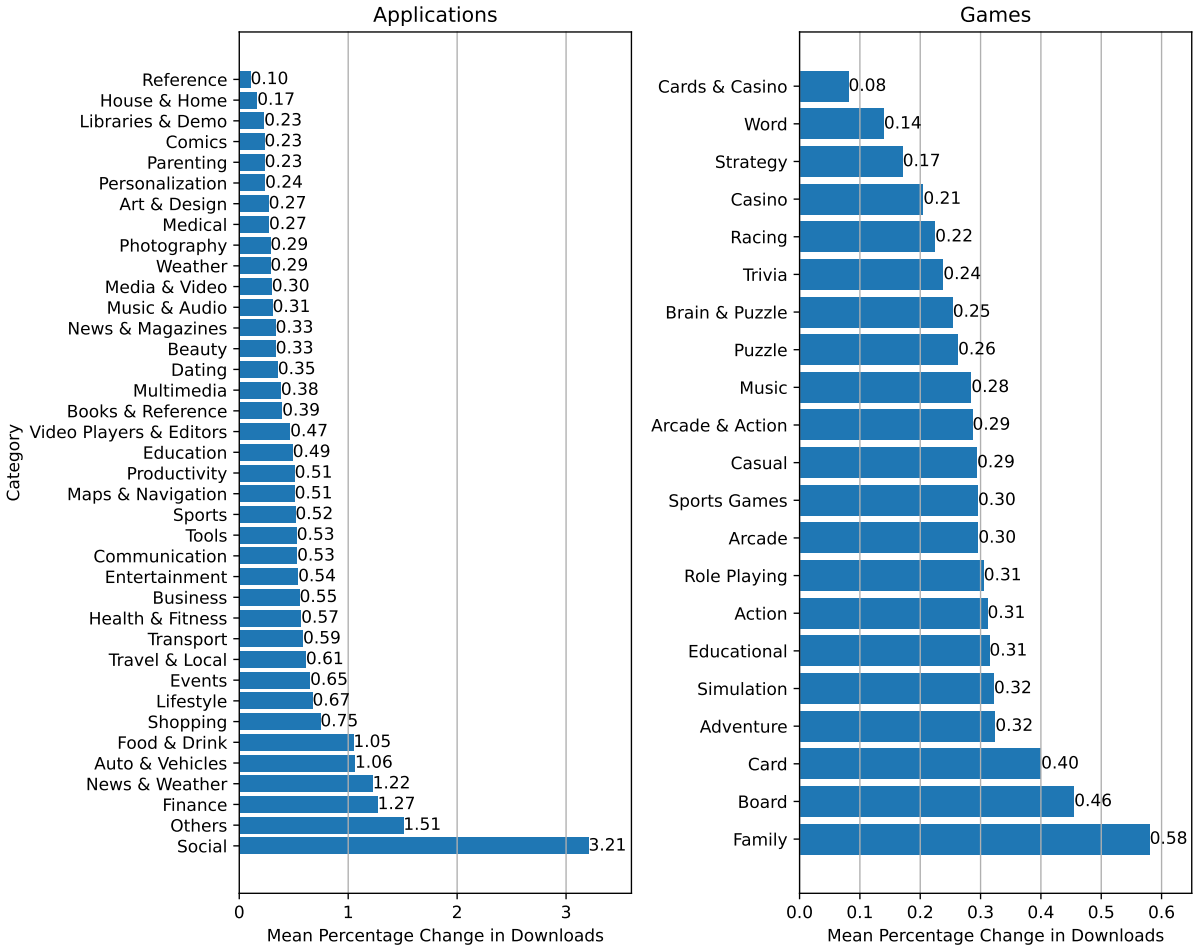

FIGURE 4.16. Comparing Downloads by Category between 2021 and 2024

users. A similar trend can be seen in the "Games" category, with the majority of transitions coming from PEGI-3. These transitions follow a similar pattern to those in the "Applications" category, with shifts to PEGI-12, PEGI-7, PEGI-16 and PEGI-18. Transitions from PEGI-3 to PEGI-18 were mainly in the "Casino" genre, reflecting content inappropriate for children. Transitions to PEGI-12 were common in genres such as "Role-Playing" and "Action", indicating content suitable for slightly older audiences. Figure 4.19 shows a heatmap of the top three categories with the highest number of PEGI rating changes among applications categorized under "Games". A dominant trend is evident, with most applications moving from PEGI-3 to either PEGI-7 or PEGI-12. Notably, some applications in the "Action" category moved directly from PEGI-3 to PEGI-16. In contrast, Figure 4.18 shows a similar heatmap focusing on PEGI rating changes within the "Applications" category. Categories such as "Entertainment" and "Tools" have predominantly moved from PEGI-3 to PEGI-12. However, the "Social" category stands out, with a significant trend of applications moving directly from PEGI-3 to PEGI-18.

In what concerns ratings, Figure 4.20 shows an analysis of rating increases within the "Games" and "Applications" categories, segmented by their respective genres. Within the "Games" category, the "Action" genre stands out with significant increments in total

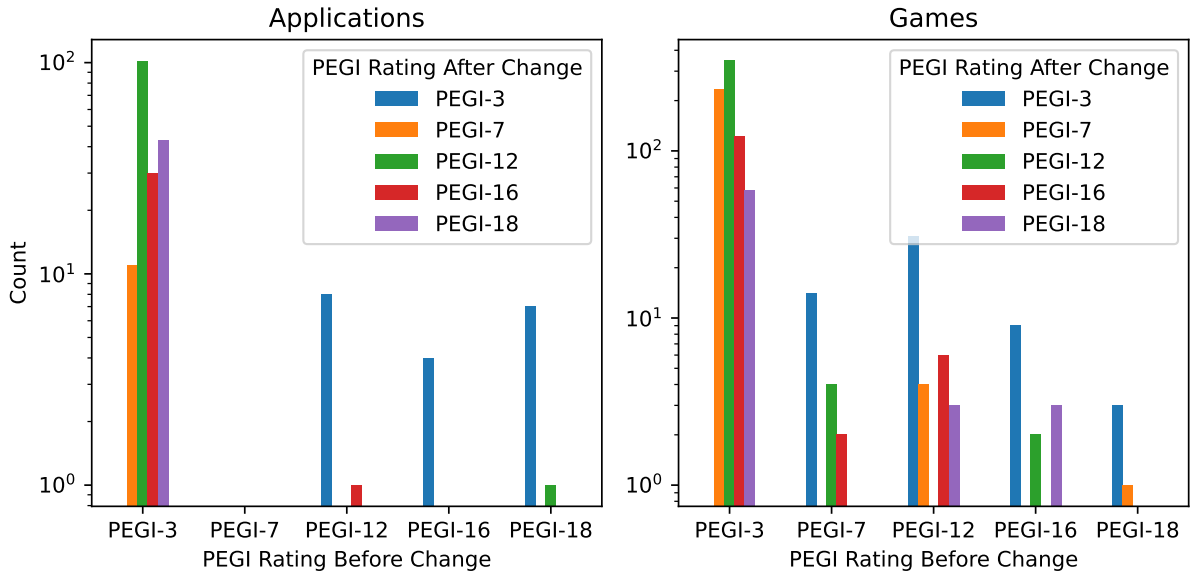

FIGURE 4.17. Comparing PEGI Ratings between 2021 and 2024

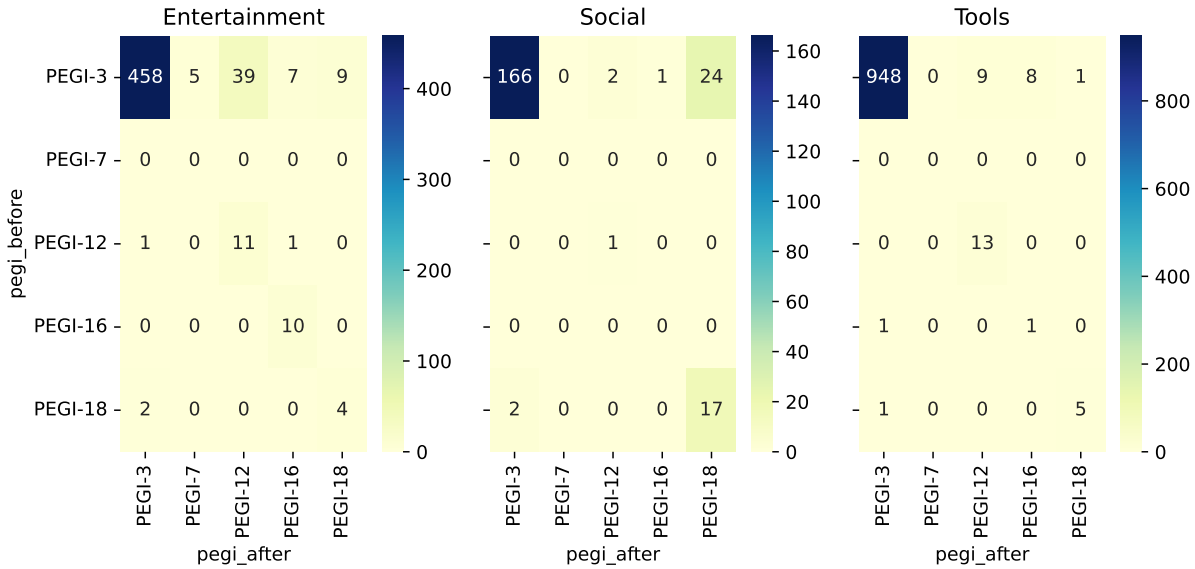

FIGURE 4.18. Heatmaps of PEGI Rating Changes for Top 5 Categories with most changes in Applications Category

ratings. Examples such as "Free Fire MAX" and "Fortnite" show substantial increases of 2,766 and 1,707 ratings, respectively. This growth underscores the continued popularity and widespread adoption of action-oriented games. In the "Applications" category, the "Social" category emerges as the leader, driven by notable applications such as "Facebook", "Instagram" and "TikTok". The "Communication" genre also shows significant growth, led by platforms such as "WhatsApp Messenger" and "Messenger Lite". These findings highlight the high demand and engagement among users for social and communication applications.

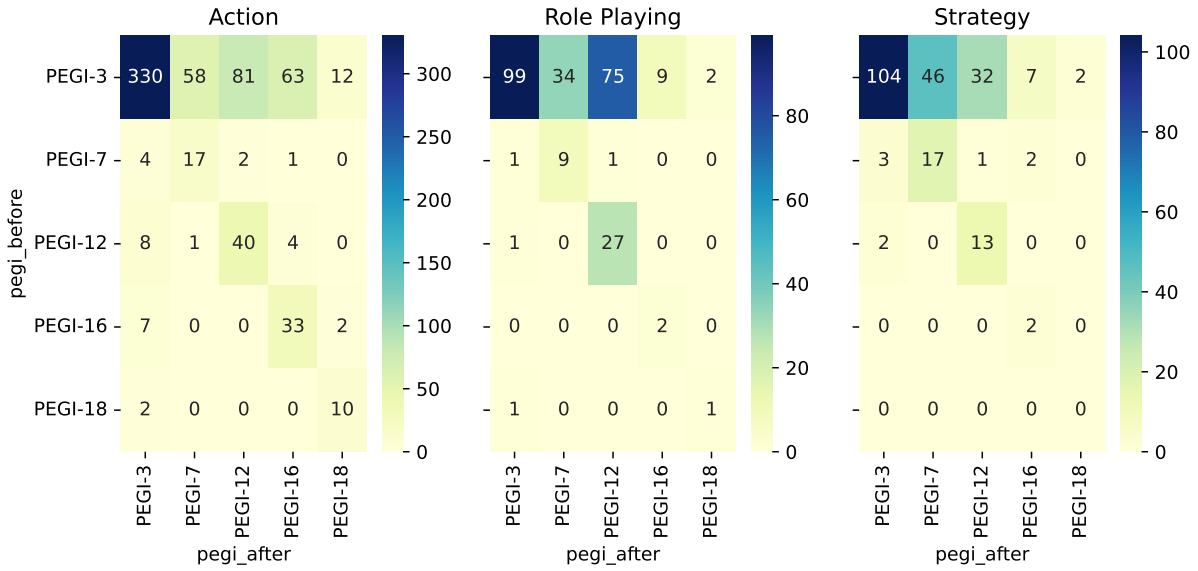

FIGURE 4.19. Heatmaps of PEGI Rating Changes for Top 3 Categories with most changes in Games Category

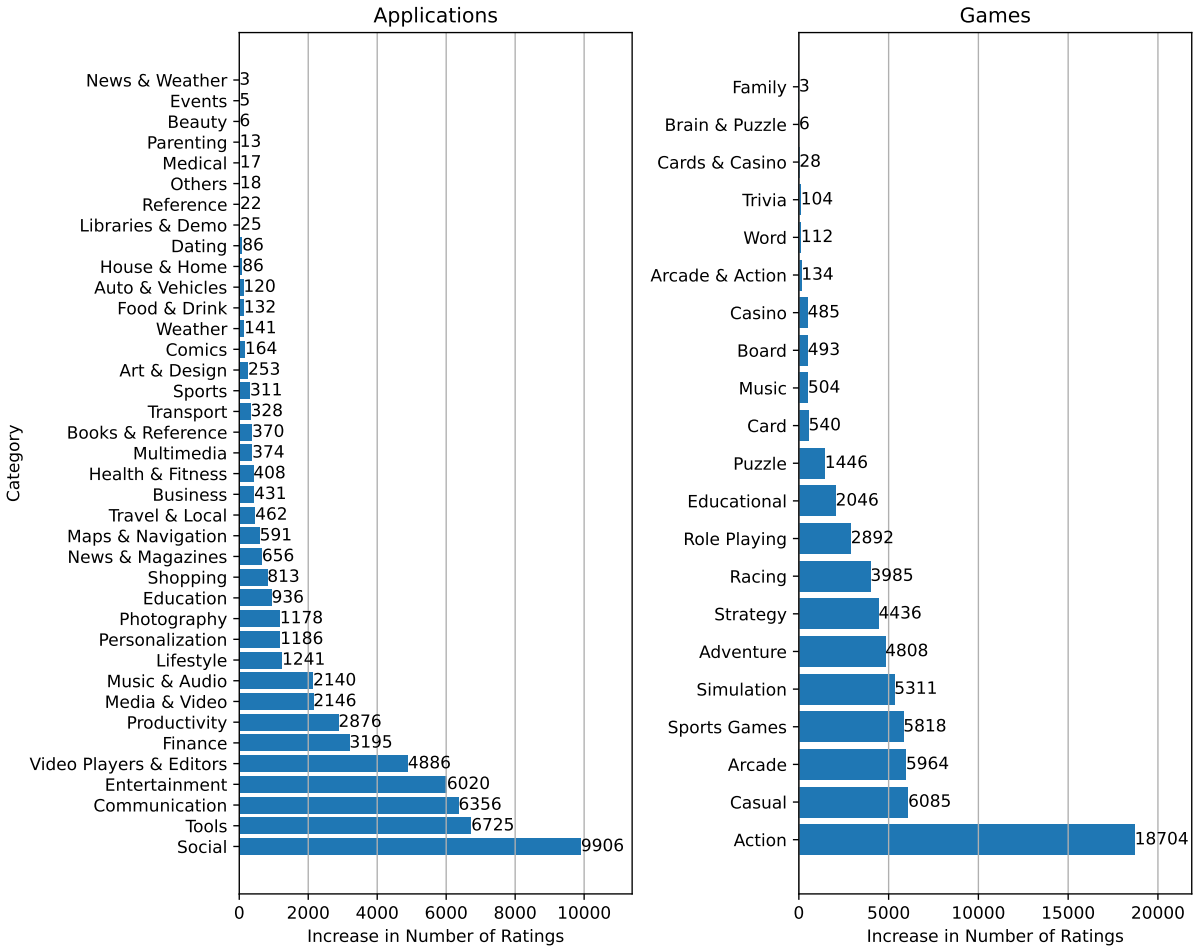

FIGURE 4.20. Increase in the Number of Ratings by Category

Our comprehensive analysis from 2021 to 2024 also examined the dynamic shifts in app categories. We identified 334 apps that experienced a change in categorization. Specifically, 67 apps lost their categorization and 15 experiencing the removal of only one tag.

Conversely, 179 apps gained one or more tags in their categorization, without losing one, contributing to a notable influx of new classifications. In addition, 197 apps retained one of their original 2021 categories, but also experienced changes in other tags, indicating a degree of stability amidst broader categorization shifts. To further complicate matters, 73 apps experienced both a loss and gain of categories, resulting in a complete overhaul of their categorization labels. An analysis of the top five categories with the most significant disappearances revealed that "Tools" was the most affected with 23 instances, followed by "Entertainment" with 17, "Sports Games" with 13 and "Arcade" and "Casual" with 8 each. On the other hand, the categories with the most new entries were "Simulation", "Tools", "Strategy", "Role Playing" and "Casual", with 20, 18, 18, 15 and 15, respectively.

4.2. Aptoide App Dataset in 2024

To build the 2024 dataset, we followed a systematic approach using three different APIs to collect comprehensive app data. The first step involved using the first API[2], which provided information on all available categories. This query yielded a total of 65 different categories, providing a robust taxonomy for organizing the subsequent app data.

We then used the second API[3], where the `group_name` parameter corresponds to each category obtained in the previous step. This API facilitated the collection of application data under each category, resulting in a comprehensive dataset of 438,303 applications. However, upon closer inspection, 39,091 applications were identified as duplicates, representing approximately 8.92% of the dataset. These duplicates were primarily identifiable by their respective categories. However, this API did not return all the available metadata, as it was missing attributes like the description and PEGI rating. To resolve this issue, we used the third and final API[4], where `package_name` serves as the unique identifier for each application, allowing us to retrieve all available metadata for 416,916 unique applications.

After a thorough review of the dataset, information on 229 applications was found to be inaccessible due to restricted access permissions, resulting in a final count of 416,687 applications available for analysis. Further refinement was performed to include duplicate entries based on categorical associations, resulting in a dataset of 438,025 entries. This duplication is primarily due to applications being associated with multiple categories, resulting in 21,338 duplicate entries from 17,663 unique applications.

A detailed inspection of the dataset reveals a notable distribution: 92,756 entries belong to the "Games" category, while the remaining 345,269 entries are classified under "Applications". It is important to note that applications can belong to multiple categories and may appear in both "Games" and "Applications". The "Applications" category includes a diverse range of 44 genres, while the "Games" category includes 21 different classifications. Within the "Applications" category, the top three prevalent genres are

[2] https://ws75.aptoide.com/api/7/apks/groups/get
[3] https://ws75.aptoide.com/api/7/apps/get?group_name={group_name}
[4] https://ws75.aptoide.com/api/7/app/getMeta/package_name={package_name}

"Education", "Tools" and "Personalization", whereas in the "Games" category, "Puzzle", "Casual" and "Simulation" emerge as the primary classifications, as shown in Figure 4.21.

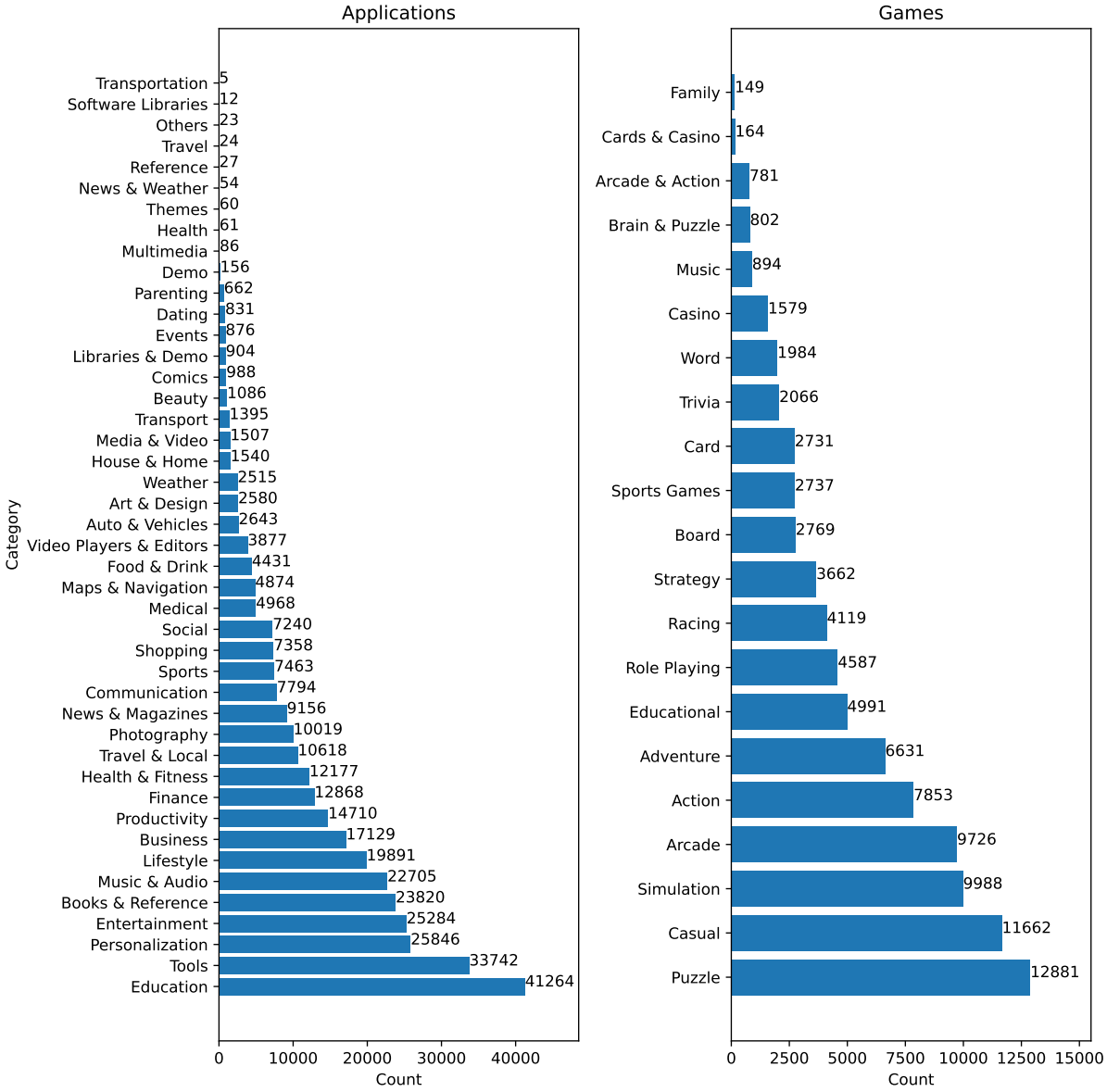

FIGURE 4.21. Number of Apps by Category in 2024

In terms of user ratings, a significant proportion of applications, 80.39% (334,954 entries), have received a rating of 0. Among the rated applications, which account for 81,733 entries, the average rating is 4.2, as shown in Figure 4.22. A detailed examination of the rating distribution shows that the majority of applications with ratings tend to have a rating of 5, indicating a high level of user satisfaction.

An analysis of the download metrics, as shown in Figure 4.23, indicates that 24.03% (10,012 entries) of the applications have never been downloaded. Furthermore, the majority of applications have fewer than 100 downloads, highlighting the prevalence of niche or less popular applications within the dataset.

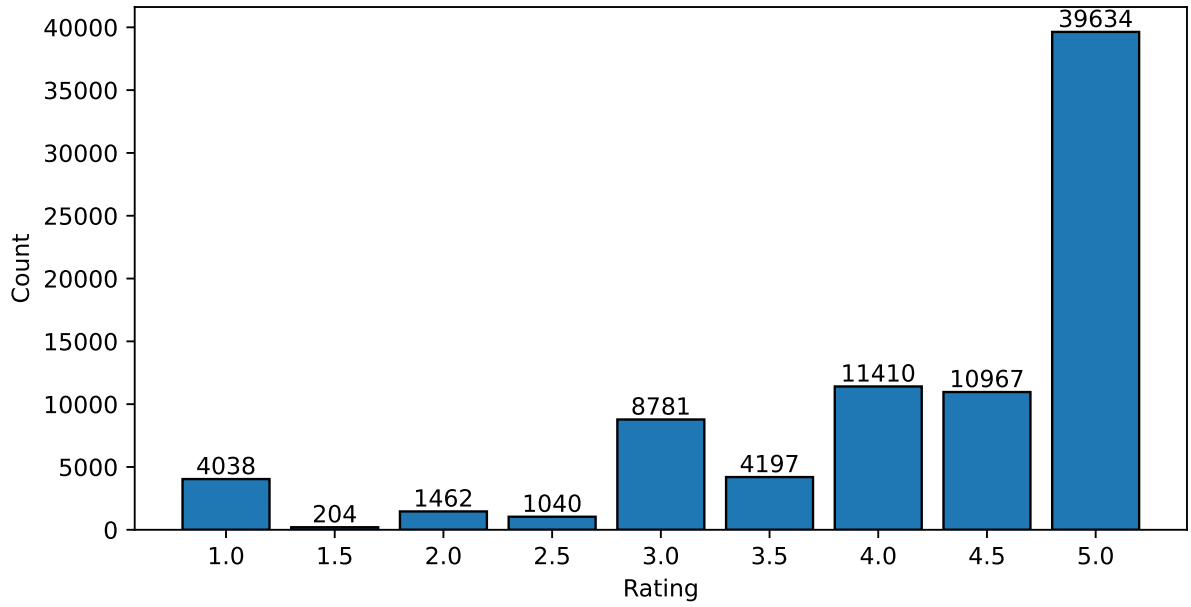

FIGURE 4.22. Rating vs. Frequency in 2024

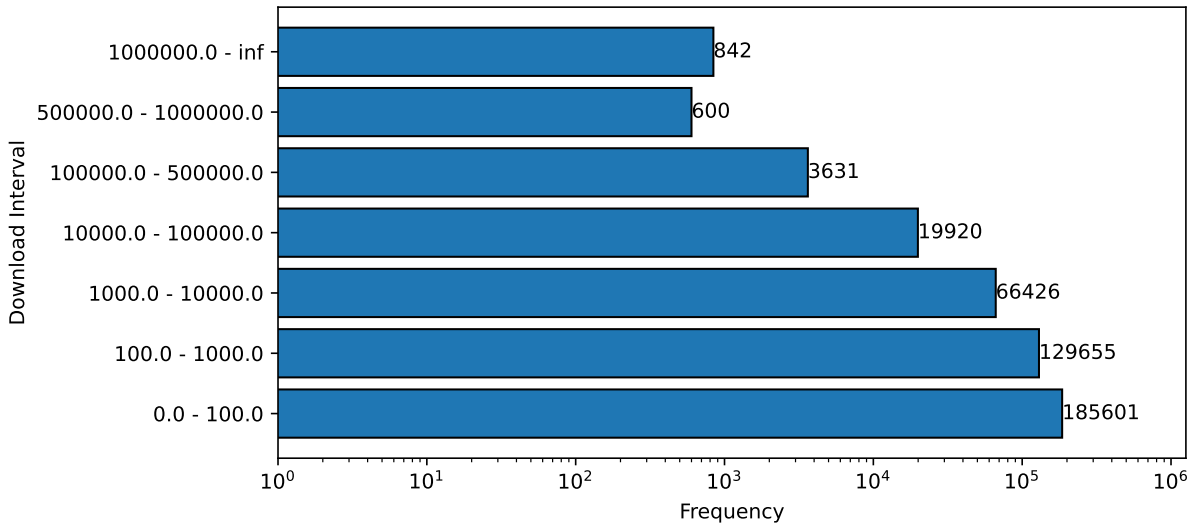

FIGURE 4.23. Download Intervals vs. Frequency in 2024

In the area of textual metadata, it is noteworthy that 309 applications lack a descriptive narrative. As shown in Figure 4.25, the majority of applications have descriptions of less than 200 words. In terms of keyword usage, 795 applications have no keywords at all, suggesting potential areas for metadata enrichment. In addition, a significant portion of applications have between 4 and 6 keywords, as shown in Figure 4.24, underscoring the importance of concise and descriptive keyword assignment for application discovery and categorization purposes.

In terms of PEGI ratings, as shown in Figure 4.26, the majority of applications meet the PEGI-3 guidelines, indicating suitability for all ages. This is closely followed by applications rated PEGI-12, reflecting content suitable for teenagers. This distribution

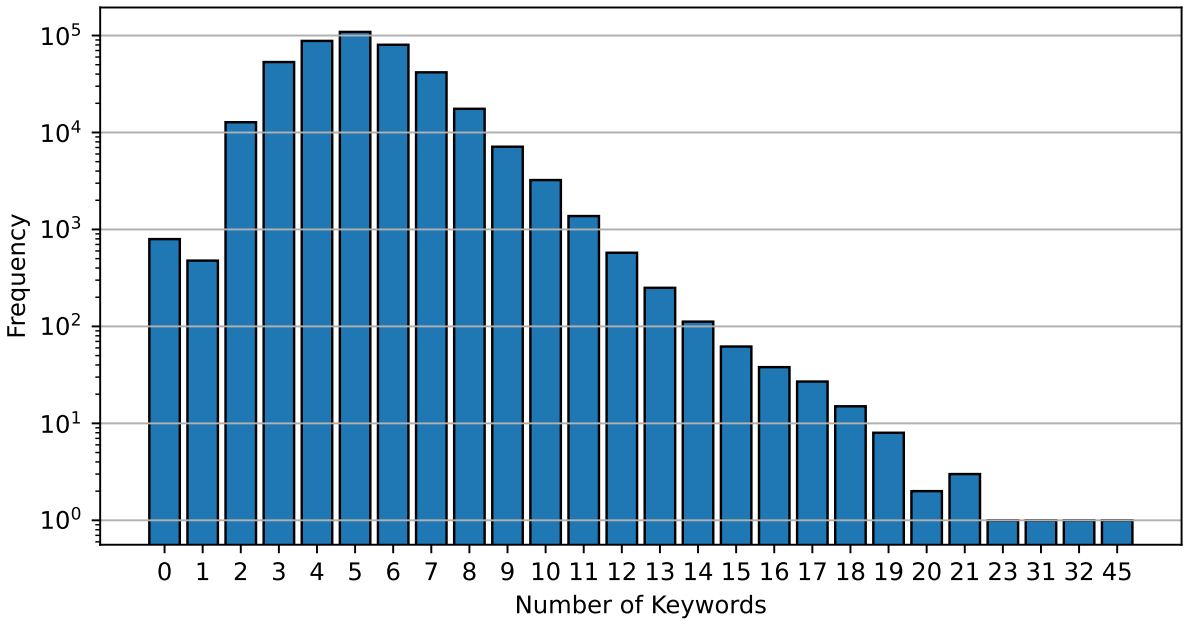

FIGURE 4.24. Number of Keywords vs. Frequency in 2024

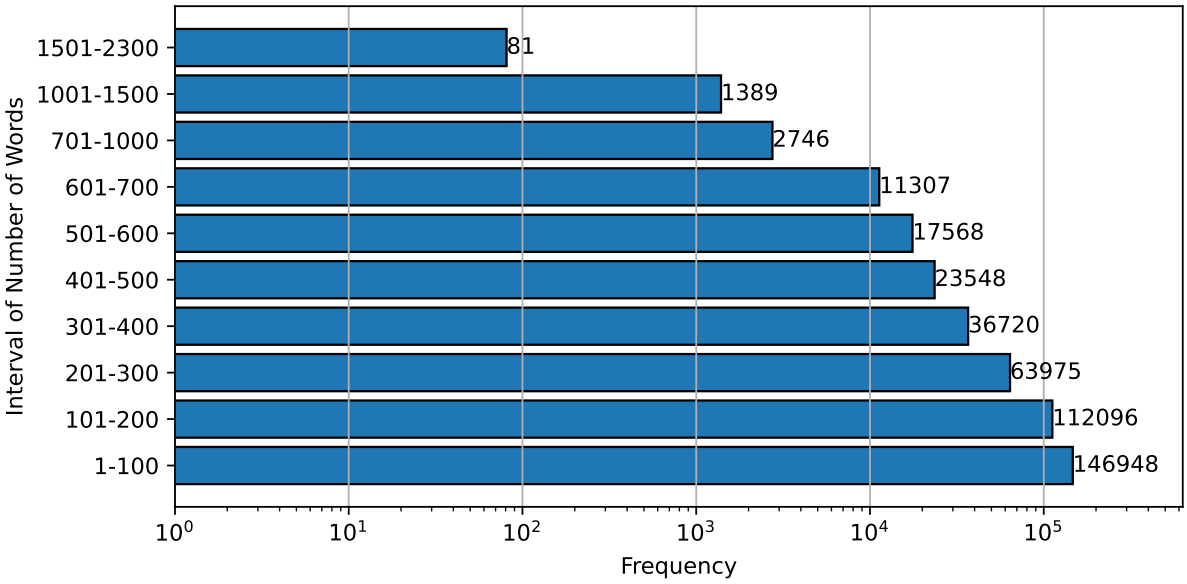

FIGURE 4.25. Number of Words in Description vs. Frequency in 2024

highlights the importance of age-appropriate content in the digital ecosystem, with a significant proportion of applications targeting younger audiences.

4.2.1. Subset Containing the Most Relevant Apps

After the initial exploration of the dataset, we refined our focus by retaining only the most relevant attributes, as described in Section 4.1. This strategic approach was intended to facilitate a focused analysis and ensure that subsequent investigations were anchored in the most relevant variables in line with our research objectives.

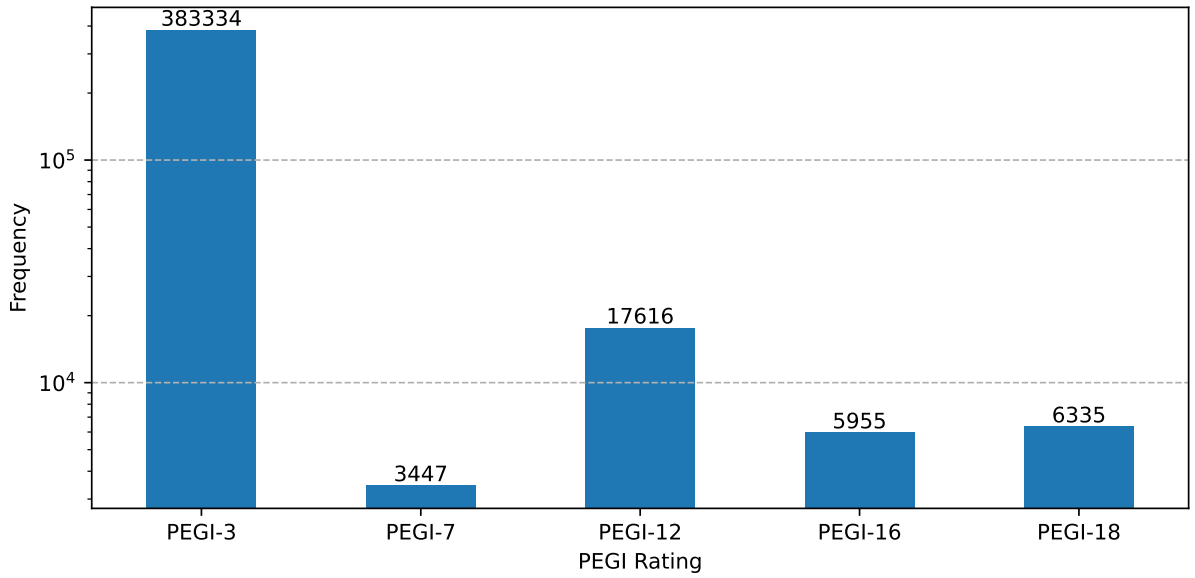

FIGURE 4.26. PEGI Rating vs. Frequency in 2024

To identify the most relevant applications, we applied the set of heuristic guidelines described in Section 4.1.2. After applying these heuristic filters, we used the language detection mechanism, also described in Section 4.1.2, to determine the linguistic composition of the application descriptions. This step was vital to ensure that only applications with English descriptions were retained for further analysis. By standardizing the linguistic composition of the dataset, this process facilitated a more cohesive and interpretable analysis, reducing potential confounding factors arising from linguistic diversity. This process resulted in a subset of 9,163 unique applications. Including duplicates, which are applications classified in more than one category, increased the total to 10,967 entries.

Within this refined subset, there is a nuanced distribution between the "Games" and "Applications" categories. Specifically, "Games" account for 4,743 entries, while "Applications" account for 6,224 entries. This disparity highlights the diversity of digital offerings available to users, spanning entertainment, productivity and utility.

Upon closer examination, the "Applications" category reveals a rich diversity with 38 distinct subcategories catering to different user needs and preferences. Conversely, the "Games" domain exhibits a more concentrated taxonomy with 21 classifications, each offering unique gaming experiences. A deeper dive into these classifications reveals that "Tools" and "Entertainment" dominate within the "Applications" domain, reflecting a significant demand for utility and leisure-oriented applications. Meanwhile, the "Games" landscape is characterized by the prevalence of the "Action" and "Simulation" genres, suggesting a preference for immersive and dynamic gaming experiences, as shown in Figure 4.27.

There is a clear trend in content ratings according to the PEGI system. The majority of applications in both the "Games" and "Applications" categories are rated PEGI-3, suitable for all ages. However, there is a notable divergence in the higher age ratings. Within the

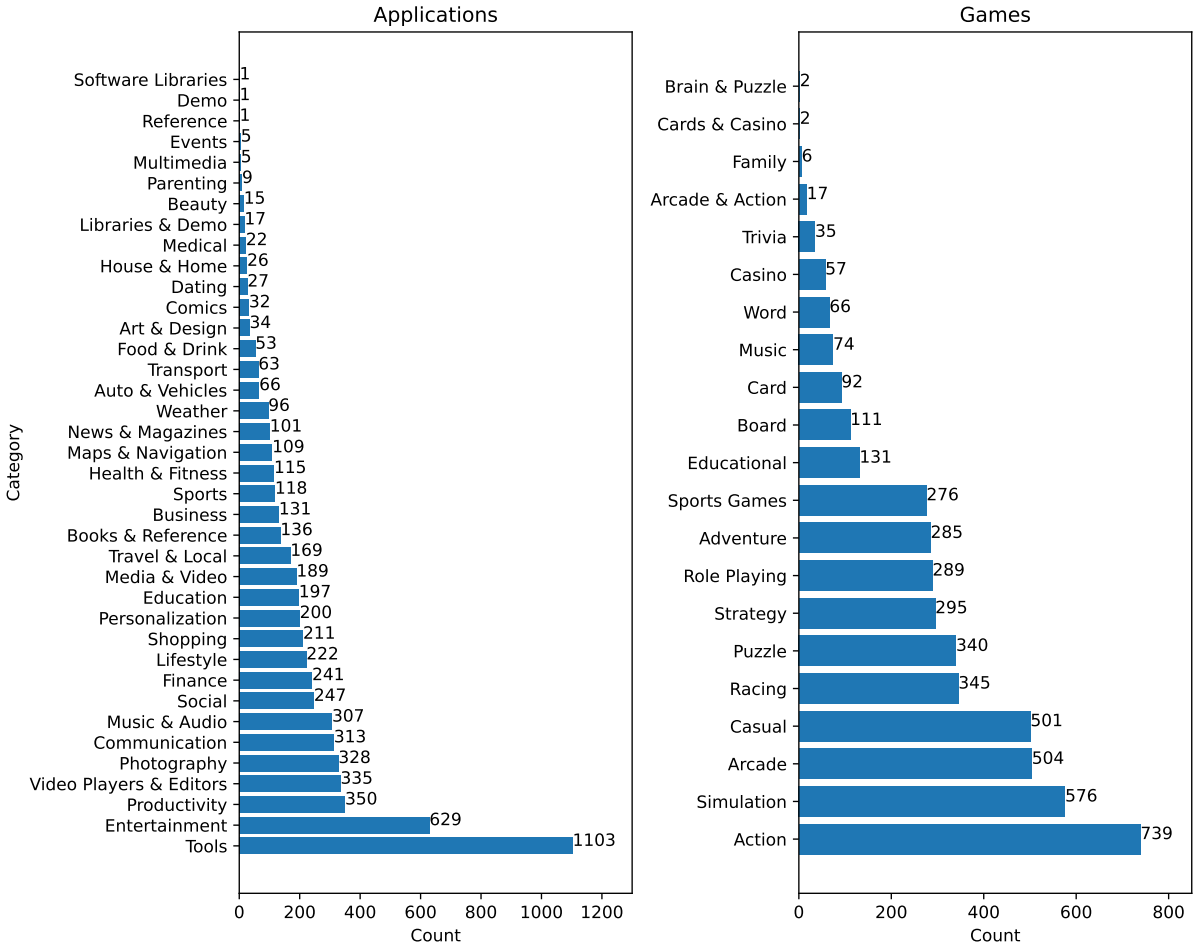

FIGURE 4.27. Number of Apps by Category in Subset 2024

"Games" category, a larger proportion of applications are rated PEGI-7, PEGI-12 and PEGI-16, reflecting content tailored to young adolescents and teenagers. Both categories have a similar proportion of PEGI-18 rated applications, which are intended for mature audiences, as shown in Figure 4.28.

Turning to the analysis of textual metadata, insights derived from keyword density and description length provide valuable information about the characteristics of applications. Across both categories, an average keyword count of five underscores the importance of concise and descriptive application metadata, as shown in Figure 4.29. In addition, the average description length of 307 words, as shown in Figure 4.30, indicates a balance between providing comprehensive information and maintaining clarity and brevity. Consistent with Subset 2021, it is noteworthy that the top three keywords consistently include the names of their corresponding categories. This observation underscores the relevance of category-specific terms in the dataset and suggests a strong association between the most frequent keywords and the categories they represent.

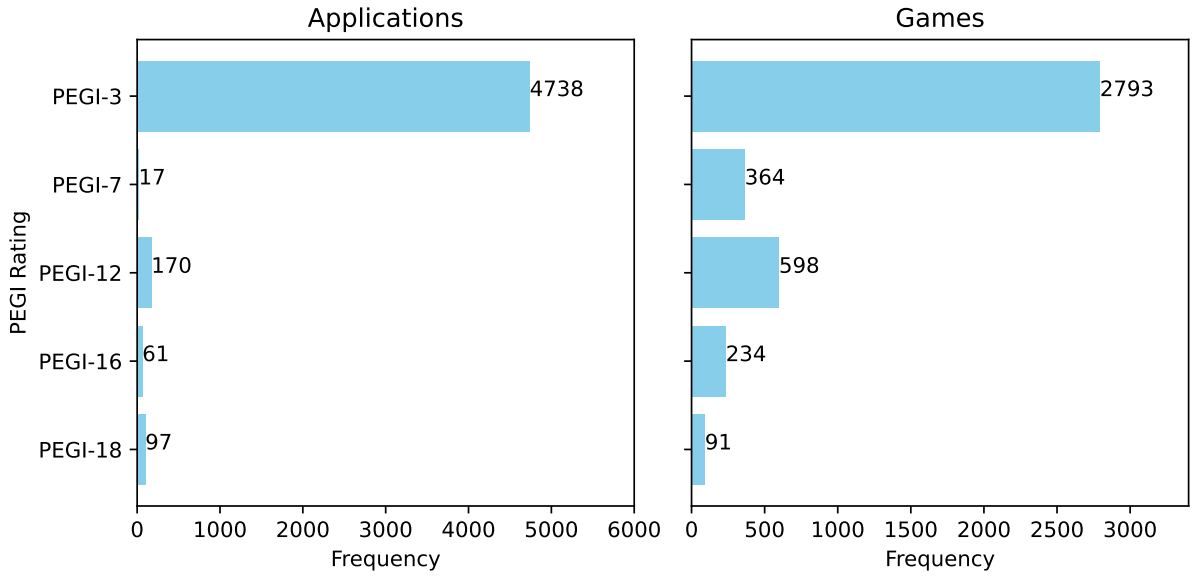

FIGURE 4.28. PEGI Rating by Parent Category in Subset 2024

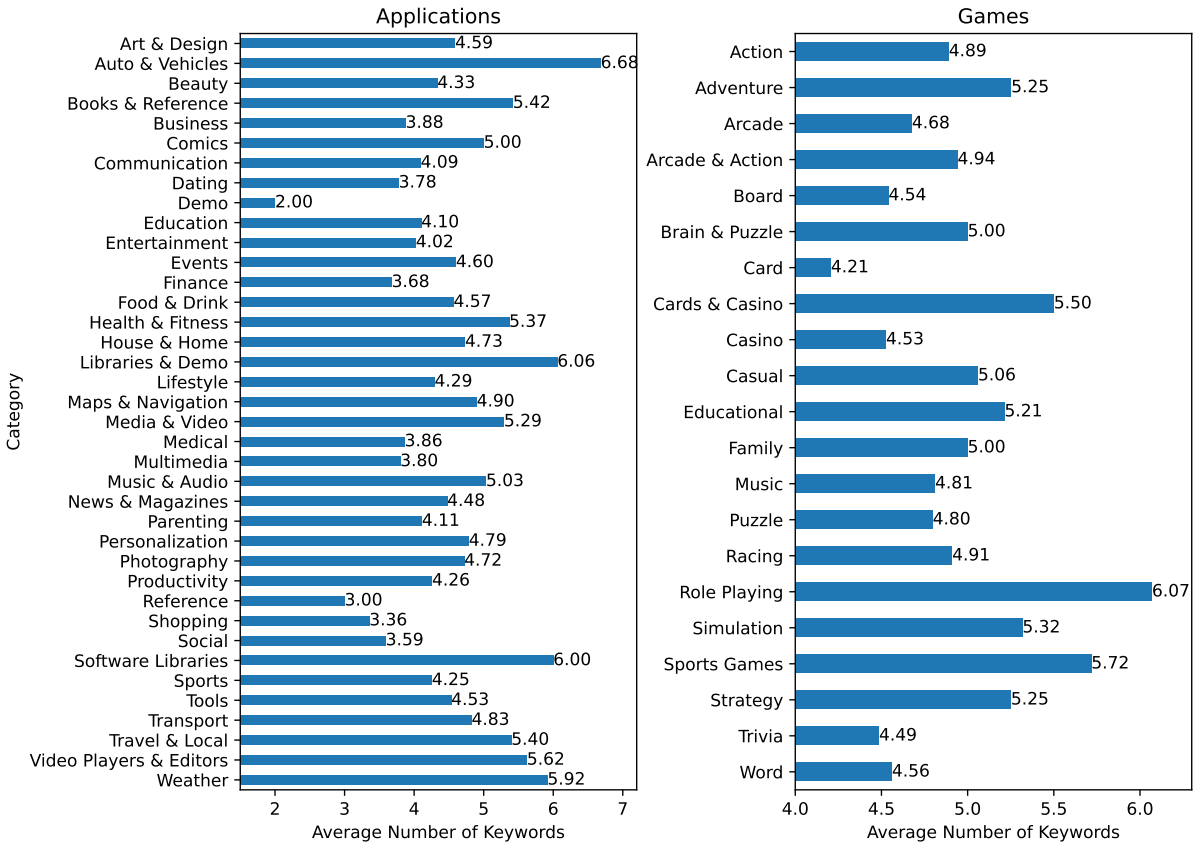

FIGURE 4.29. Average Number of Keywords by Category in Subset 2024

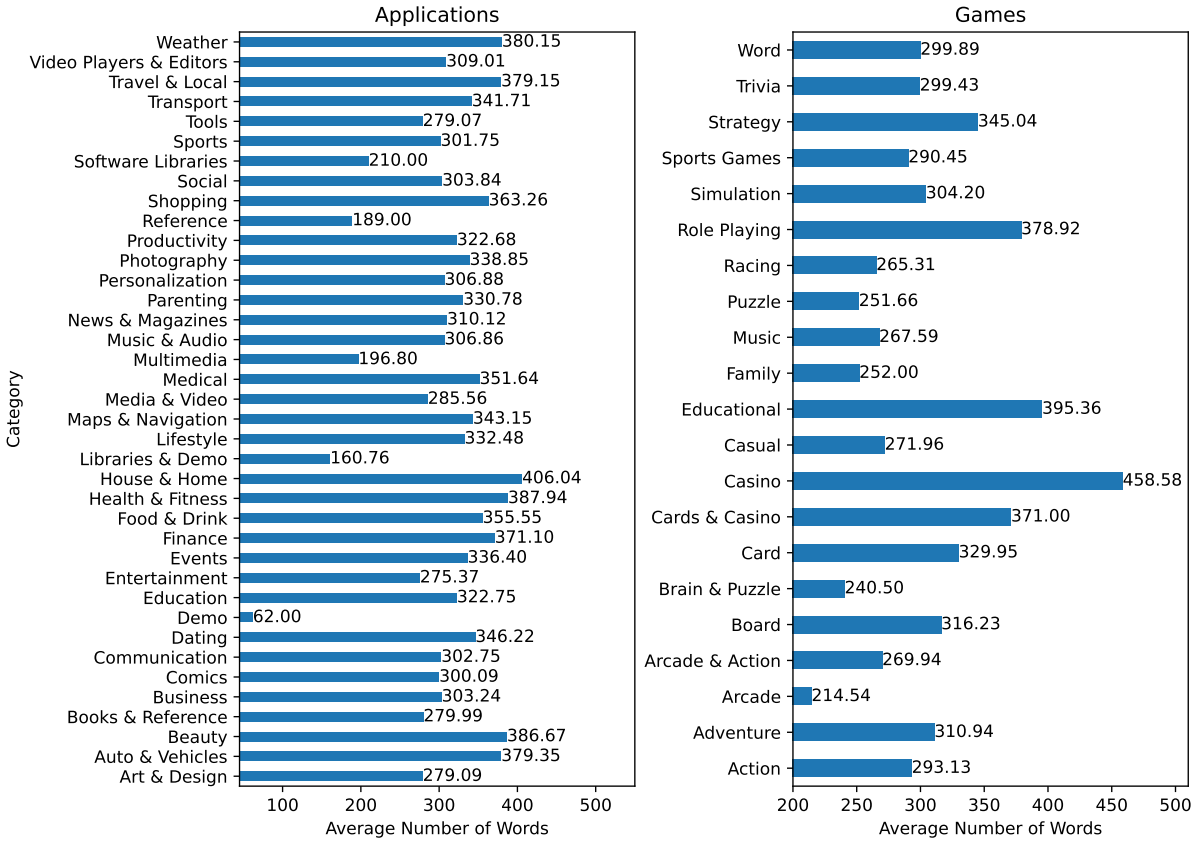

FIGURE 4.30. Average Number of Words in Description by Category in Subset 2024

CHAPTER 5

Experiments

Our primary goal is to perform a comparative analysis of different text representation and feature extraction methods, focusing on their effectiveness in categorizing mobile applications into predefined categories. The categorization of mobile applications is crucial for several reasons: it enhances the discoverability of applications, improves the user experience and helps to systematically organize the entire application ecosystem. In the following sections, we will provide a detailed overview of our text preprocessing procedures and elaborate on the specific analysis methods we intend to use. Each method will be explained in terms of its underlying mechanisms and potential benefits in the context of our study.

5.1. Metrics

For evaluation metrics, we used the following metrics:

Precision: measures the proportion of true positive instances out of all instances predicted to be positive by the model. It represents the performance of the model in correctly identifying relevant instances. Mathematically, precision is calculated as the ratio of true positives to the sum of true positives and false positives:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{5.1}$$

Recall: (also known as sensitivity or true positive rate) quantifies the ability of the model to correctly identify all relevant instances of a given class. It is calculated as the ratio of true positives to the sum of true positives and false negatives:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{5.2}$$

F1-score: is the harmonic mean between precision and recall, providing a balanced measure that considers both precision and recall simultaneously. F1-score is particularly useful when there is an uneven distribution of classes, or when both false positives and false negatives are important. It is calculated as:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.3}$$

When dealing with multiclass classification, especially when there are class imbalances, it becomes important to understand different averaging strategies for aggregating metrics such as precision, recall and F1 score across multiple classes:

Macro-Average: This averaging method calculates the metric independently for each class and then takes the unweighted average of these values. It treats all classes equally, regardless of their frequency in the data set.

Micro-Average: Unlike the macro average, the micro average aggregates the contributions of all classes by taking into account the total number of true positives, false positives and false negatives across all classes. It is particularly useful when dealing with class imbalances, as it gives more weight to the larger classes.

Weighted Average: This method is similar to the macro average, but it accounts for class imbalance by weighting the contribution of each class by its support (the number of true instances for each class). It gives more weight to classes with more instances, providing a more representative measure of overall model performance.

Sample Average: Also known as Instance Average, this method calculates the metric for each instance and then averages them. It treats each instance equally, regardless of class membership.

5.2. Static Embeddings vs Topic Modelling

In the following section, we will perform a comparative analysis between Word2Vec and L-LDA to determine their relative effectiveness for the task at hand. Through systematic experimentation and evaluation, we aim to elucidate the strengths and limitations of each method. By evaluating their performance, we aim to provide actionable insights for selecting the most appropriate approach.

5.2.1. Dataset Preprocessing

To proceed with the dataset at hand, described in Section 4.2.1, a preliminary filtering process was performed to retain only those applications whose category occurred with a frequency greater than 100 occurrences, leaving us with 33 categories. This filtering was implemented to ensure a robust analysis of sufficiently represented categories within the dataset. Figure 5.1 shows the remaining categories. A grouping mechanism was then applied, recognizing that applications may be associated with multiple categories. Applications were grouped based on their "packages", resulting in a consolidation of all categories associated with each individual application. This grouping approach allows for a comprehensive understanding of the multiple categories to which each application belongs. Text preprocessing plays a important role in NLP tasks by transforming raw text into a format suitable for analysis and modeling. There are several key steps involved in this process:

Tokenization: The first step in this preprocessing pipeline was tokenization, which breaks down each text document into individual words. The text was converted to lower case during this process to facilitate consistency in subsequent operations.

Punctuation removal: After tokenization, punctuation was removed from the tokens. Punctuation, such as commas, periods and quotation marks, can introduce noise and hinder analysis.

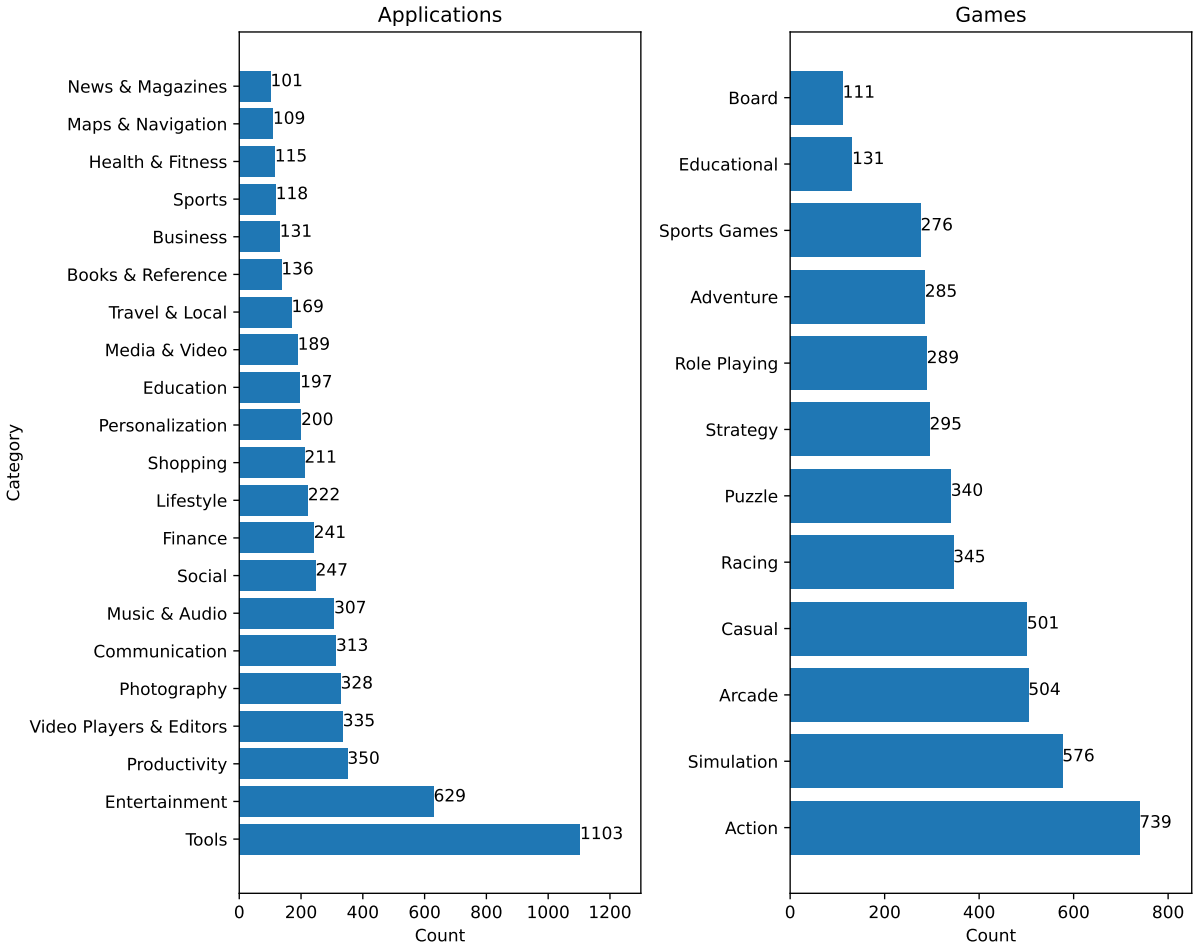

FIGURE 5.1. Number of Apps by Category after preprocessing

Stopword removal: The preprocessing pipeline then incorporated the removal of stopwords, common words that often do not add significant meaning to a sentence or document.

Handling of special characters, digits and URLs: Regular expressions were used to systematically eliminate digits, special characters and remove all URLs beginning with "http" or "https". Additionally, leading and trailing whitespace was stripped from each token to ensure uniformity.

Lemmatization: The final stage of preprocessing involved lemmatization, a linguistic technique that reduces words to their base or root forms, known as lemmas. This step is particularly valuable because it standardizes variations of words, such as different verb tenses or plural forms, to their common lemma representation.

5.2.2. Classification Algorithms

All the multilabel classification approaches can use different classification algorithms. Since we aim at understanding the most appropriate multilabel classification appproaches, for simplicity, our experiments adopted the following classical machine learning classification methods:

- Support Vector Machine (SVM);
- K-Nearest Neighbors (KNN);
- Logistic Regression (LR);
- Decision Trees (DT);
- Random Forest (RF).

5.2.3. Results

In this study, we maintain a stable test dataset throughout our experiments to ensure a consistent and fair evaluation framework. The primary objective of our analysis is to compare the performance of different multi-label classification algorithms in the context of mobile app categorization. Specifically, we investigate the effectiveness of the algorithms indicated in the Section 2.3 in the context of the three adopted approaches for multi-label classification: Label PowerSet, Classifier Chain and Binary Relevance.

Upon careful analysis of the results presented in Table 5.1, obtained using Word2Vec, it is evident that the combination that yielded the highest score in F1-micro avg, F1-macro avg, F1-weighted avg and F1-samples avg was RF with Label PowerSet, closely followed by LR again with Label PowerSet.

Analyzing the results presented in Table 5.2, it is clear that different algorithms combined with the Label PowerSet technique have distinct strengths in multi-label classification tasks using L-LDA document representations. In particular, RF and SVM with Label PowerSet emerge as top performers, considering certain F1 score metrics. RF excels in F1-micro Avg and F1-weighted Avg, demonstrating balanced performance and adaptability to different class sizes. This indicates RF's ability to effectively handle different class distributions, ensuring that performance remains robust regardless of class size. On the other hand, SVM achieves the best results in F1-macro Avg and F1-samples Avg, demonstrating its effectiveness in maintaining precision-recall balance and handling unbalanced data distributions. These measures indicate how well SVM maintains precision-recall balance, especially in situations where the data distributions are asymmetric. Thus, each algorithm has unique characteristics that make it suitable for different aspects of multi-label classification. RF's strength lies in its overall balanced performance across different class sizes, while SVM excels at handling unbalanced data distributions and maintaining precision-recall balance. Both are strong choices for multi-label classification applications, depending on the specific needs and characteristics of the dataset.

Comparing both multi-label classification models, it becomes clear that the L-LDA representations consistently outperforms the Word2Vec embeddings across a spectrum of evaluation metrics, except when used with LR. This tendency is also confirmed at the per-sample level (samples average evaluation). This means that the L-LDA approach, is more adept at handling individual samples and providing accurate multi-label predictions.

TABLE 5.1. Word2Vec Multi Label Classifier Results with Label PowerSet, Classifier Chain and Binary Relevance

Label Powerset

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.632	0.557	**0.592**	0.680	0.525	**0.571**	0.638	0.557	**0.572**	0.631	0.593	**0.604**
SVM	0.630	0.553	0.589	0.673	0.518	0.553	0.635	0.553	0.558	0.631	0.589	0.602
LR	0.630	0.554	0.589	0.629	0.530	0.561	0.614	0.554	0.568	0.631	0.592	0.603
KNN	0.594	0.530	0.560	0.606	0.544	0.565	0.594	0.530	0.553	0.596	0.563	0.572
DT	0.407	0.415	0.411	0.388	0.396	0.390	0.409	0.415	0.410	0.429	0.435	0.421

Classifier Chain

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.793	0.327	0.464	0.819	0.310	0.420	0.790	0.327	0.435	0.318	0.353	0.357
SVM	0.635	0.417	0.503	0.573	0.382	0.424	0.596	0.417	0.441	0.480	0.446	0.457
LR	0.646	0.414	0.505	0.634	0.373	0.437	0.635	0.414	0.462	0.464	0.444	0.448
KNN	0.602	0.531	0.564	0.630	0.542	0.566	0.605	0.531	0.551	0.597	0.565	0.573
DT	0.376	0.433	0.402	0.366	0.426	0.391	0.383	0.433	0.405	0.372	0.454	0.389

Binary Relevance

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.798	0.323	0.461	0.804	0.309	0.417	0.781	0.324	0.431	0.366	0.348	0.352
SVM	0.826	0.307	0.448	0.623	0.294	0.373	0.666	0.307	0.390	0.353	0.333	0.339
LR	0.720	0.332	0.455	0.644	0.318	0.401	0.667	0.332	0.421	0.357	0.357	0.350
KNN	0.684	0.477	0.562	0.688	0.492	0.561	0.667	0.477	0.547	0.527	0.504	0.506
DT	0.376	0.433	0.402	0.366	0.426	0.391	0.383	0.433	0.405	0.372	0.454	0.389

5.3. Dynamic Embeddings

In the following section, we will conduct an in-depth analysis of the RoBERTa model to determine its effectiveness in the task of categorizing mobile applications. Through systematic experimentation and evaluation, we aim to elucidate the strengths and limitations of this method.

5.3.1. Setup

The dataset utilized in this study is the same as the one described in Section 4.2.1 and the preprocessing is the same as in Section 5.2.1. For all experiments, we consistently the same dataset for testing.

In our experiments, we used three different strategies to address the multi-label classification problem: Label PowerSet (LP), a modified version of Binary Relevance (BR) and Multi-Label Binarizer (MLB).

Our experimental setup included several fine-tuning configurations. In some cases, we fine-tuned the entire model, while in others we fine-tuned only the last one or two layers. The purpose of this variation was to assess the impact of different levels of model fine-tuning on performance. To optimize model performance and prevent overfitting, we used the ModelCheckpoint callback during training. This callback ensured that the best

TABLE 5.2. LLDA Multi Label Classifier Results with Label PowerSet, Classifier Chain and Binary Relevance

Label Powerset

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.663	0.583	**0.620**	0.703	0.576	0.623	0.668	0.583	**0.612**	0.663	0.619	0.633
SVM	0.662	0.581	0.619	0.709	0.580	**0.626**	0.673	0.581	0.609	0.664	0.621	**0.634**
LR	0.632	0.549	0.588	0.737	0.510	0.575	0.678	0.549	0.573	0.632	0.590	0.603
KNN	0.653	0.578	0.613	0.684	0.585	0.619	0.649	0.578	0.603	0.655	0.616	0.627
DT	0.557	0.560	0.546	0.539	0.532	0.532	0.529	0.528	0.527	0.557	0.560	0.546

Classifier Chain

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.749	0.471	0.579	0.760	0.473	0.569	0.741	0.471	0.565	0.532	0.510	0.514
SVM	0.607	0.530	0.566	0.719	0.524	0.588	0.676	0.530	0.563	0.609	0.570	0.582
LR	0.707	0.341	0.460	0.755	0.277	0.374	0.734	0.341	0.424	0.392	0.370	0.376
KNN	0.648	0.573	0.608	0.691	0.578	0.615	0.652	0.573	0.596	0.651	0.611	0.623
DT	0.519	0.568	0.524	0.540	0.556	0.546	0.523	0.541	0.530	0.519	0.568	0.524

Binary Relevance

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.755	0.475	0.583	0.751	0.481	0.574	0.737	0.475	0.568	0.531	0.510	0.513
SVM	0.746	0.494	0.594	0.740	0.517	0.598	0.732	0.494	0.580	0.560	0.529	0.537
LR	0.800	0.278	0.412	0.749	0.243	0.348	0.750	0.278	0.390	0.319	0.300	0.305
KNN	0.714	0.548	0.620	0.725	0.566	0.624	0.705	0.548	0.608	0.613	0.584	0.590
DT	0.519	0.568	0.524	0.540	0.556	0.546	0.523	0.541	0.530	0.519	0.568	0.524

model, based on the lowest validation loss, was saved. In addition, we sometimes used EarlyStopping, which stops training if the validation loss does not improve for three consecutive epochs. Validation Loss is a metric used to evaluate a model's performance on unseen data during training, helping to monitor and prevent overfitting. This technique helps to avoid unnecessary training cycles and reduces the risk of overfitting. All experiments were performed with a batch size of 16 and a maximum sequence length of 512 tokens. These parameters were chosen to balance computational efficiency with the ability to capture sufficient contextual information from the text.

5.3.2. Results

Table in 5.3 shows the achieved results of RoBERTa using the three different approaches: Label Powerset, Binary Relevance and Multi-Label Binarizer. As in previous experiments, the metrics provided include precision (P), recall (R) and F1 score (F1) over micro, macro, weighted and sample averages. The table compares different training configurations, including models trained without tuning, models tuned for different epochs and models trained with frozen layers and early stopping mechanisms (patience).

In the Label PowerSet approach, when the performance of the two best performing models, the 13-epoch model with patience of 3 and freezing of the first 11 layers and the 20-epoch model with freezing of the first 11 layers, are examined through the lens of

F1-score metrics, distinct advantages emerge for each, depending on the specific metric considered. The 20-epoch model has a slightly higher micro-average F1 score (0.659) compared to the 13-epoch model (0.656). This indicates that, on a per-instance basis, the 20-epoch model achieves a slightly better balance between precision and recall across all instances. Conversely, the 13-epoch model outperforms the 20-epoch model in terms of the Macro Average F1 score, with values of 0.654 and 0.646, respectively. The Macro Average F1-score emphasizes performance across all classes equally, suggesting that the 13-epoch model provides more uniform performance across different classes, which may be particularly beneficial in scenarios where class imbalance is a concern. The 13-epoch model also shows a slight advantage in the Weighted Average F1 score (0.644) compared to the 20-epoch model (0.642). The Weighted Average F1-score takes into account the prevalence of each class, indicating that the 13-epoch model may be better at handling classes with a larger number of instances. In the context of the Samples F1-score, the 20-epoch model shows a slight superiority (0.674) over the 13-epoch model (0.669). This metric takes into account the contribution of each individual sample to the F1-score, highlighting the improved performance of the 20-epoch model in scenarios where the emphasis is on individual sample predictions. If the primary concern is to achieve the best possible performance on a per-instance basis and to ensure individual sample accuracy, the 20-epoch model proves to be more advantageous. However, if the goal is to maintain balanced performance across classes, especially in the presence of class imbalance and to ensure robust handling of prevalent classes, the 13-epoch model offers a more suitable solution.

In the Binary Relevance approach, the table shows that all models were evaluated using a ModelCheckpoint mechanism to ensure that the best model was always selected based on validation performance. This approach ensures that the model does not overfit during training and consistently provides the optimal performance metrics. As we can see, the model with 10 frozen layers had better results.

In the Multi-Label Binarizer (MLB) approach, the model performed poorly without fine-tuning, achieving a Micro F1 score of 0.059 and a Macro F1 score of 0.031. This indicates that the model initially struggled to effectively predict multiple labels. Fine-tuning the model for 9 epochs with a patience mechanism set to 3 significantly improved performance, resulting in a Micro F1 score of 0.652 and a Macro F1 score of 0.643. This setup proved to be the most effective, highlighting the importance of balancing training time and stopping early to avoid overfitting. Introducing layer freezing (first 11 layers) to the 9-epoch configuration with patience slightly decreased performance, achieving a Micro F1 score of 0.633 and a Macro F1 score of 0.630. This suggests a trade-off between the stability provided by freezing layers and the flexibility needed to learn complex label correlations. Training for 20 epochs without freezing layers resulted in potential overfitting, as indicated by lower scores (Micro F1: 0.618, Macro F1: 0.574). In contrast, freezing the first 11 layers during the 20-epoch training improved generalization,

TABLE 5.3. Roberta Classifier Results with Label PowerSet, Binary Relevance and Multi-Label Binarizer

Label Powerset

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Without Fine Tuning	0.038	0.199	0.064	0.007	0.182	0.013	0.009	0.199	0.017	0.038	0.193	0.062
10 epochs	0.685	0.599	0.639	0.697	0.585	0.623	0.677	0.599	0.624	0.687	0.639	0.654
20 epochs	0.685	0.595	0.637	0.691	0.585	0.624	0.670	0.595	0.623	0.685	0.636	0.651
6 epochs patience 3	0.682	0.598	0.637	0.723	0.586	0.630	0.696	0.598	0.627	0.683	0.637	0.652
8 epochs patience 3 freeze 10 layers	0.705	0.614	0.656	0.732	0.599	0.638	0.705	0.614	0.641	0.705	0.656	0.671
20 epochs freeze first 10 layers	0.699	0.611	0.652	0.736	0.599	0.640	0.707	0.611	0.638	0.699	0.653	0.667
13 epochs patience 3 freeze first 11 layers	0.699	0.618	0.656	0.725	0.622	**0.654**	0.694	0.618	**0.644**	0.701	0.656	0.669
20 epochs freeze first 11 layers	0.706	0.618	**0.659**	0.730	0.605	0.646	0.696	0.618	0.642	0.708	0.659	**0.674**

Binary Relevance

	Micro Avg			Macro Avg			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Patience 3	0.688	0.524	0.595	0.604	0.541	0.546	0.617	0.524	0.543	0.549	0.555	0.540
Patience 3 freeze 10 layers	0.731	0.549	**0.627**	0.686	0.557	**0.598**	0.687	0.549	**0.596**	0.584	0.586	**0.572**

Multi-Label Binarizer

	Micro			Macro			Weighted			Samples		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
without fine tuning	0.032	0.422	0.059	0.017	0.465	0.031	0.025	0.422	0.045	0.032	0.423	0.059
9 Epochs patience 3	0.721	0.595	**0.652**	0.723	0.595	**0.643**	0.712	0.595	**0.641**	0.659	0.631	**0.635**
9 Epochs patience 3 freeze first 11 layers	0.687	0.586	0.633	0.724	0.588	0.630	0.701	0.586	0.622	0.649	0.621	0.625
20 Epochs	0.743	0.528	0.618	0.759	0.508	0.574	0.739	0.528	0.592	0.592	0.558	0.567
freeze first 11 layers 20 Epochs	0.740	0.537	0.622	0.719	0.526	0.583	0.712	0.537	0.592	0.607	0.573	0.583

resulting in better performance (Micro F1: 0.622, Macro F1: 0.583). Overall, the best performance in the MLB approach was achieved with 9 epochs of training and a patience setting of 3, which balanced adequate learning with effective overfitting prevention.

5.4. Large Language Models

In this section, we describe the methodology used to classify mobile application descriptions into predefined categories using several state-of-the-art pre-trained language models. The models evaluated include:

- `mistralai/Mistral-7B-Instruct-v0.3` [18]
- `CallComply/Starling-LM-11B-alpha` [74]
- `meta-llama/Meta-Llama-3-8B-Instruct` [21]
- `microsoft/Phi-3-mini-128k-instruct` [22]
- `microsoft/Phi-3-small-128k-instruct` [24]
- `microsoft/Phi-3-medium-128k-instruct` [23]
- `gpt-3.5 turbo` [25]
- `gpt-4o` [26]

Our primary objective was to develop an optimal prompt that could be used effectively across these models to achieve the highest classification accuracy. After multiple iterations, we reached the following prompt, which is described at the Listing 5.1.

The prompt is designed to guide the models in classifying a given text into one of the specified categories. It explicitly instructs the model to use only the categories listed and not to create new categories or use any that are not on the list. This constraint ensures consistency and relevance in the classification task. Additionally, we provide

LISTING 5.1. Prompt function for classifying mobile app descriptions.

```
def new_prompt(description, examples, invalid_category=None):
    prompt = """
You are provided with a list of mobile app categories below. Your task is to classify
    ↪ the following text into one of these categories.
Use only the categories listed. Do not create new categories or use categories that
    ↪ are not on the list.
Answer with the single word that matches one of the categories exactly.

Categories:
- Photography
- Communication
- Tools
- Social
- Business
- Entertainment
- Adventure
- Action
- Travel & Local
- Strategy
- News & Magazines
- Education
- Sports Games
- Shopping
- Productivity
- Books & Reference
- Video Players & Editors
- Music & Audio
- Lifestyle
- Casual
- Personalization
- Media & Video
- Finance
- Arcade
- Simulation
- Racing
- Maps & Navigation
- Board
- Health & Fitness
- Educational
- Puzzle
- Sports
- Role Playing

Here are some examples:

""" + examples + "\nNow, classify the following text:\n" + description

    if invalid_category:
        prompt += f"\nNote: The response '{invalid_category}' given previous does not
            ↪ belong to the provided categories."

    return prompt
```

10 examples of descriptions along with their respective categories to aid in the model's understanding. During our execution, if the model fails to return one of the possible categories, an if-statement will appear in the next attempt, indicating that the result is not part of the possible categories. We allow the model a maximum of three tries to produce an acceptable response. If after three attempts the model does not return a valid category, we will default to the category that appears most frequently among the examples, which is "Tools". This fallback mechanism ensures that every description is categorized, maintaining the integrity and completeness of the classification process.

5.4.1. Results

After reaching this prompt, we utilized the same test set used in all previous experiments to evaluate the performance of each model. The results are presented in Table 5.4. The attribute "#Errors" represents the number of misclassifications made by the model. Specifically, it indicates how many times, after three attempts, the model failed to predict a correct category and instead returned one or more categories outside the set of thirty-three possible options.

When analyzing Microsoft's pre-trained language models in the Table 5.4, it is clear that models with more parameters generally produce better results. However, it is interesting to note that the best performing model, which has 14 billion parameters, has a higher number of misclassifications. As shown in Table 5.4, the best performing model across multiple metrics was `gpt-4o`. It achieved the highest F1 scores in the Micro, Macro and Weighted averages, with values of 0.626, 0.609 and 0.609 respectively. Additionally, it demonstrated the best performance in the Samples category with an F1 score of 0.640 and had zero wrong classifications, indicating its robustness and accuracy in classifying mobile app descriptions using the categories given. It is also noteworthy that Meta's language model was the only other model able to predict the categories without any misclassifications.

When comparing with one of the best RoBERTa models from the section 5.3 and GPT-4o model, several key points emerge:

- **Micro F1 Score:** RoBERTa (0.659) vs. GPT-4o (0.626)
  - RoBERTa outperforms GPT-4o by 0.033 in the Micro F1 score, indicating better precision and recall when considering individual label instances.
- **Macro F1 Score:** RoBERTa (0.646) vs. GPT-4o (0.609)
  - RoBERTa surpasses GPT-4o by 0.037 in the Macro F1 score, showing superior performance in treating all classes equally, regardless of their frequency.
- **Samples F1 Score:** RoBERTa (0.674) vs. GPT-4o (0.640)
  - RoBERTa exceeds GPT-4o by 0.034 in the Samples F1 score, highlighting its effectiveness in evaluating the classification performance on a per-sample basis.
- **Weighted F1 Score:** RoBERTa (0.642) vs. GPT-4o (0.609)

TABLE 5.4. Large Language Model Results

	Micro Avg			Macro Avg			Weighted			Samples			#Errors
Model	P	R	F1	P	R	F1	P	R	F1	P	R	F1	
mistralai/Mistral-7B-Instruct-v0.3	0.462	0.608	0.525	0.549	0.582	0.538	0.523	0.608	0.533	0.530	0.645	0.555	22
CallComply/Starling-LM-11B-alpha	0.548	0.476	0.509	0.630	0.460	0.472	0.623	0.476	0.472	0.548	0.508	0.521	54
meta-llama/Meta-Llama-3-8B-Instruct	0.422	0.636	0.508	0.517	0.583	0.508	0.485	0.636	0.511	0.499	0.669	0.542	0
microsoft/Phi-3-mini-128k-instruct	0.493	0.436	0.463	0.524	0.462	0.462	0.541	0.436	0.453	0.494	0.460	0.468	21
microsoft/Phi-3-small-128k-instruct	0.497	0.435	0.464	0.537	0.446	0.439	0.544	0.435	0.438	0.498	0.461	0.471	19
microsoft/Phi-3-medium-128k-instruct	0.563	0.499	0.529	0.561	0.490	0.496	0.585	0.499	0.519	0.565	0.530	0.538	44
gpt-3.5 turbo	0.640	0.565	0.600	0.680	0.575	0.598	0.671	0.565	0.595	0.643	0.601	0.611	5
gpt-4o	0.667	0.589	**0.626**	0.676	0.588	**0.609**	0.674	0.589	**0.609**	0.672	0.630	**0.640**	0

– RoBERTa achieves a higher Weighted F1 score by 0.033, indicating better performance across labels considering their frequency.

The better results achieved by the RoBERTa model, which is an encoder-based architecture, over the decoder-based GPT-4o model can be attributed to several inherent advantages of encoder models in certain NLP tasks. Encoders, such as those used in the RoBERTa architecture, excel at understanding the context and structure of the input text. This is particularly beneficial for tasks such as text classification, where the goal is to assign a category or label based on the input description. A key reason for the better performance of encoders is their ability to capture bidirectional context. Unlike decoders, which typically generate text sequentially, encoders process the entire input text simultaneously, allowing them to understand the relationships between words in both directions. This bidirectional nature helps build a more complete representation of the text, leading to more accurate classifications. Furthermore, encoder models often benefit from extensive pre-training on large datasets, followed by fine-tuning on specific tasks, which enhances their ability to generalize and perform well on diverse datasets. In contrast, decoder models, while powerful at generating coherent and contextually relevant text, may not be as adept at tasks that require nuanced understanding and classification of input data. Thus, the structured and context-aware processing capabilities of encoders contribute significantly to their superior performance in classification tasks, as evidenced by the higher F1 scores achieved by the RoBERTa model in this comparison.

CHAPTER 6

Conclusions

In this study, we aimed to perform a comprehensive comparative analysis of different text representation and feature extraction methods to evaluate their effectiveness in categorizing mobile applications into predefined categories. This categorization is crucial for improving application discoverability, enhancing user experience and organizing the app ecosystem.

The rapid growth of mobile applications has resulted in an overwhelming number of applications available to users. Effective categorization plays a key role in managing this vast array of applications and ensuring that users can quickly and efficiently find the apps they need. In addition to improving searchability, proper categorization improves the overall user experience by organizing applications in a logical and intuitive manner. This in turn helps users navigate app stores with ease and find relevant apps without being overwhelmed by unrelated content. In addition, accurate categorization contributes significantly to the health and functionality of the app ecosystem. For developers, being in the right category increases the visibility of their apps to the right audience, potentially leading to higher download rates and user engagement. It also enables better analytics and insight into app performance within specific categories, helping developers understand user preferences and improve their offerings accordingly. From a business perspective, well-categorized apps benefit from improved discoverability, leading to better monetization opportunities. App stores can implement targeted marketing strategies, promote relevant apps to specific user segments, and offer personalized recommendations. This not only increases app store revenue, but also improves user satisfaction by providing a tailored browsing experience.

In the first phase of our research, we focused on acquiring a up-to-date dataset of mobile applications from the year 2024. This approach allowed us to compare the characteristics of the 2024 dataset with those of a 2021 dataset, thereby allowing us to analyze the evolution and differences over time. From the comprehensive 2024 dataset, we then derived a smaller subset to serve as the focus of our experiments. This subset was selected based on specific criteria relevant to our study, ensuring that it provides meaningful insights and supports the goals of our research. In our research, we focused on evaluating several advanced text representation and feature extraction methods, including Word2Vec, L-LDA, RoBERTa and other pre-trained language models. Each of these methods offers unique advantages in processing and interpreting the textual metadata associated with mobile applications. By comparing these techniques, we aimed to identify the most effective approach for accurately categorizing applications based on their

descriptions. Word2Vec, for example, creates word embeddings that capture semantic relationships between words, which can be crucial for understanding app descriptions. L-LDA, on the other hand, is a topic modeling technique that helps identify underlying themes within a corpus of text, providing insight into the key functionalities and features of an app. RoBERTa, a transformer-based model, uses deep learning to understand context and semantics at a much deeper level, making it highly effective for nuanced text analysis. Our comparative analysis involved rigorous experimentation with these methods to determine their performance in multi-label classification tasks. In addition, other pre-trained language models, such as GPT-4o, further improve classification performance by using extensive training on diverse datasets to capture intricate language patterns and contextual relationships. We evaluated their ability to accurately classify applications into multiple relevant categories, taking into account various performance metrics such as F1 scores. In doing so, we aimed to provide a clear understanding of which text representation techniques are most effective for categorizing mobile apps. Finally, our results showed that the RoBERTa model outperformed the other methods, making it the most effective technique for this task.

In conclusion, through our comprehensive analysis, we provided insights concerning the best methods to improve this categorization process, ultimately contributing to a more efficient and user-friendly app marketplace.

6.1. Future Work

Despite the promising results, several innovative avenues for future research could significantly enhance the findings of this study:

- **Integration of Multimodal Data**: Future research could explore the integration of multimodal data, such as app screenshots, user reviews and videos, to enrich text representation models. Combining visual and textual information could provide a more holistic understanding of mobile applications and improve categorization accuracy.
- **Using Federated Learning**: Investigating federated learning approaches could enable model training across decentralized datasets, ensuring privacy while benefiting from a broader range of user data. This could improve the robustness and generalizability of classification models across diverse user bases and environments.
- **Adapting to Evolving Taxonomies**: It would be valuable to develop adaptive models that can dynamically adapt to changes in application categorization taxonomies over time. This could include creating models that can learn from new categories and subcategories as they emerge to maintain relevance in a rapidly evolving app ecosystem. For example, in our study, we identified 334 applications that shifted categories between 2021 and 2024. By implementing adaptive models, we could detect additional apps that also meet the criteria for category changes but may not have been manually reclassified.

- **Interactive and Explainable AI Systems**: Creating interactive AI systems that allow users to interrogate and understand model decisions in real time could improve trust and usability. Developing methods for explainable AI that provide clear, interpretable reasons for categorization decisions would be particularly beneficial to developers and end users.
- **Cross-Language and Cross-Cultural Studies**: Expanding the scope to include cross-lingual and cross-cultural datasets could assess the performance and adaptability of models in different linguistic and cultural contexts. This could include training multilingual models capable of seamlessly handling application descriptions in different languages.
- **Personalized Categorization Models**: Research could explore personalized categorization models that adapt to individual user preferences and usage patterns. This could include using user-specific data to fine-tune models and provide personalized app recommendations and categorizations.
- **Assessing Ethical Implications**: Investigating the ethical implications of automated app categorization, including potential biases and fairness issues, could ensure that the models developed are not only effective, but also fair and equitable. This could include creating frameworks for auditing and mitigating bias in categorization models.
- **Integration with application development lifecycles**: Future studies could explore the integration of categorization models into the app development lifecycle, providing real-time feedback to developers during the app creation process. This could help developers optimize their apps for better discoverability and user engagement from the start.
- **Gamification and User Engagement**: Implementing gamification strategies to engage users in the categorization process could provide valuable data for model training while improving user interaction. This could include creating user-friendly interfaces that allow users to participate in categorization tasks and earn rewards.

By pursuing these innovative research directions, the field can make progress toward creating highly effective, reliable and user-centric systems for categorizing mobile applications, ultimately improving the user experience and the organization of the mobile application ecosystem.

References

[1] 4. AG, *Google Play Statistics and Trends 2023 — 42matters.com*, https://42matters.com/google-play-statistics-and-trends, [Accessed 02-12-2023].

[2] 4. AG, *IOS Apple App Store Statistics and Trends 2023 — 42matters.com*, https://42matters.com/ios-apple-app-store-statistics-and-trends, [Accessed 02-12-2023].

[3] *The App Store turns 10 — apple.com*, https://www.apple.com/newsroom/2018/07/app-store-turns-10/, [Accessed 02-12-2023].

[4] R. Churchill and L. Singh, "The evolution of topic modeling," *ACM Comput. Surv.*, vol. 54, no. 10s, 215:1–215:35, 2022. DOI: 10.1145/3507900. [Online]. Available: https://doi.org/10.1145/3507900.

[5] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Processes*, vol. 25, no. 2–3, pp. 259–284, Jan. 1998, ISSN: 1532-6950. DOI: 10.1080/01638539809545028. [Online]. Available: http://dx.doi.org/10.1080/01638539809545028.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003. [Online]. Available: http://jmlr.org/papers/v3/blei03a.html.

[7] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, 6-7 August 2009, Singapore, A meeting of SIGDAT, a Special Interest Group of the ACL*, ACL, 2009, pp. 248–256. [Online]. Available: https://aclanthology.org/D09-1026/.

[8] J. Barnard, *What are word embeddings?* Sep. 2024. [Online]. Available: https://www.ibm.com/topics/word-embeddings.

[9] Y. Wang, Y. Hou, W. Che, and T. Liu, "From static to dynamic word representations: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 7, pp. 1611–1630, 2020. DOI: 10.1007/S13042-020-01069-8. [Online]. Available: https://doi.org/10.1007/s13042-020-01069-8.

[10] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, L. Vanderwende,

H. D. III, and K. Kirchhoff, Eds., The Association for Computational Linguistics, 2013, pp. 746–751. [Online]. Available: https://aclanthology.org/N13-1090/.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds., ACL, 2014, pp. 1532–1543. DOI: 10.3115/V1/D14-1162. [Online]. Available: https://doi.org/10.3115/v1/d14-1162.

[12] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *CoRR*, vol. abs/1607.01759, 2016. arXiv: 1607.01759. [Online]. Available: http://arxiv.org/abs/1607.01759.

[13] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 6294–6305. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/20c86a628232a67e7bd46f76fba7ce12-Abstract.html.

[14] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds., Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/V1/N18-1202. [Online]. Available: https://doi.org/10.18653/v1/n18-1202.

[15] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, I. Gurevych and Y. Miyao, Eds., Association for Computational Linguistics, 2018, pp. 328–339. DOI: 10.18653/V1/P18-1031. [Online]. Available: https://aclanthology.org/P18-1031/.

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[17] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[18] Mistralai, *Mistral-7b-instruct-v0.3*, https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3, Accessed: 2024-07-04, 2024.

[19] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de Las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," *CoRR*, vol. abs/2310.06825, 2023. DOI: 10.48550/ARXIV.2310.06825. [Online]. Available: https://doi.org/10.48550/arXiv.2310.06825.

[20] CallComply, *Starling-lm-11b-alpha*, https://huggingface.co/CallComply/Starling-LM-11B-alpha, Accessed: 2024-08-05, 2023.

[21] AI@Meta, "Llama 3 model card," 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[22] Microsoft, *Phi-3-mini-128k-instruct*, https://huggingface.co/microsoft/Phi-3-mini-128k-instruct, Accessed: 2024-07-04, 2024.

[23] Microsoft, *Phi-3-medium-128k-instruct*, https://huggingface.co/microsoft/Phi-3-medium-128k-instruct, Accessed: 2024-07-04, 2024.

[24] Microsoft, *Phi-3-small-128k-instruct*, https://huggingface.co/microsoft/Phi-3-small-128k-instruct, Accessed: 2024-07-04, 2024.

[25] OpenAI, *Gpt-3.5 turbo*, https://platform.openai.com/docs/models/gpt-3-5-turbo, Accessed: 2024-07-04, 2023.

[26] OpenAI, *Gpt-4o*, https://openai.com/index/hello-gpt-4o/, Accessed: 2024-07-04, 2023.

[27] M. Zhang, Y. Li, X. Liu, and X. Geng, "Binary relevance for multi-label learning: An overview," *Frontiers Comput. Sci.*, vol. 12, no. 2, pp. 191–202, 2018. DOI: 10.1007/S11704-017-7031-7. [Online]. Available: https://doi.org/10.1007/s11704-017-7031-7.

[28] E. O. Kiyak, D. Birant, and K. U. Birant, "Comparison of multi-label classification algorithms for code smell detection," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2019, pp. 1–6. DOI: 10.1109/ISMSIT.2019.8932855.

[29] J. David Costa Júnior, E. Faria, J. Silva, and R. Cerri, "Label powerset for multi-label data streams classification with concept drift," Oct. 2017.

[30] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains: A review and perspectives," *J. Artif. Intell. Res.*, vol. 70, pp. 683–718, 2021. DOI: 10.1613/JAIR.1.12376. [Online]. Available: https://doi.org/10.1613/jair.1.12376.

[31] scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html.

[32] *How to do SLR — drshahizan.gitbook.io*, https://drshahizan.gitbook.io/slr/introduction/how-to-do-slr, [Accessed 02-12-2023].

[33] *Understanding PRISMA 2020 — drshahizan.gitbook.io*, https://drshahizan.gitbook.io/slr/prisma-2020/understanding-prisma-2020, [Accessed 02-12-2023].

[34] S. Mokarizadeh, M. T. Rahman, and M. Matskin, "Mining and analysis of apps in google play," in *WEBIST 2013 - Proceedings of the 9th International Conference on Web Information Systems and Technologies, Aachen, Germany, 8-10 May, 2013*, K. Krempels and A. Stocker, Eds., SciTePress, 2013, pp. 527–535.

[35] S. Vakulenko, O. Müller, and J. vom Brocke, "Enriching itunes app store categories via topic modeling," in *Proceedings of the International Conference on Information Systems - Building a Better World through Information Systems, ICIS 2014, Auckland, New Zealand, December 14-17, 2014*, M. D. Myers and D. W. Straub, Eds., Association for Information Systems, 2014. [Online]. Available: http://aisel.aisnet.org/icis2014/proceedings/EBusiness/18.

[36] M. Nayebi, H. Farrahi, A. Lee, H. Cho, and G. Ruhe, "More insight from being more focused: Analysis of clustered market apps," in *Proceedings of the International Workshop on App Market Analytics, WAMA@SIGSOFT FSE, Seattle, WA, USA, November 14, 2016*, M. Nagappan, F. Sarro, and E. Shihab, Eds., ACM, 2016, pp. 30–36. DOI: 10.1145/2993259.2993266. [Online]. Available: https://doi.org/10.1145/2993259.2993266.

[37] A. A. Al-Subaihin, F. Sarro, S. Black, and L. Capra, "Empirical comparison of text-based mobile apps similarity measurement techniques," *Empir. Softw. Eng.*, vol. 24, no. 6, pp. 3290–3315, 2019. DOI: 10.1007/S10664-019-09726-5. [Online]. Available: https://doi.org/10.1007/s10664-019-09726-5.

[38] A. Finkelstein, M. Harman, Y. Jia, W. J. Martin, F. Sarro, and Y. Zhang, "Investigating the relationship between price, rating, and popularity in the blackberry world app store," *Inf. Softw. Technol.*, vol. 87, pp. 119–139, 2017. DOI: 10.1016/J.INFSOF.2017.03.002. [Online]. Available: https://doi.org/10.1016/j.infsof.2017.03.002.

[39] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, P. Jalote, L. C. Briand, and A. van der Hoek, Eds., ACM, 2014, pp. 1025–1035. DOI: 10.1145/2568225.2568276. [Online]. Available: https://doi.org/10.1145/2568225.2568276.

[40] X. Che and Q. Sun, "A feature and deep learning model recommendation system for mobile application," in *2021 IEEE 7th International Conference on Big Data Intelligence and Computing (DataCom)*, IEEE, 2021, pp. 35–40.

[41] A. Fuad and M. Al-Yahya, "Analysis and classification of mobile apps using topic modeling: A case study on google play arabic apps," *Complex.*, vol. 2021, 6677413:1–6677413:12, 2021. DOI: 10.1155/2021/6677413. [Online]. Available: https://doi.org/10.1155/2021/6677413.

[42] B. Cao, J. Chen, J. Liu, and Y. Wen, "A topic attention mechanism and factorization machines based mobile application recommendation method," *Mob. Networks Appl.*,

vol. 25, no. 4, pp. 1208–1219, 2020. DOI: 10.1007/S11036-020-01537-Z. [Online]. Available: https://doi.org/10.1007/s11036-020-01537-z.

[43] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016*, ACM, 2016, 38:1–38:10. DOI: 10.1145/2961111.2962600. [Online]. Available: https://doi.org/10.1145/2961111.2962600.

[44] D. L. B. Lulu and T. Kuflik, "Functionality-based clustering using short textual description: Helping users to find apps installed on their mobile device," in *18th International Conference on Intelligent User Interfaces, IUI 2013, Santa Monica, CA, USA, March 19-22, 2013*, J. Kim, J. Nichols, and P. A. Szekely, Eds., ACM, 2013, pp. 297–306. DOI: 10.1145/2449396.2449434. [Online]. Available: https://doi.org/10.1145/2449396.2449434.

[45] H. Qiu, Z. Wu, and X. Zhang, "Exploring multiple genres text classification: Classifying 61 genres of mobile app description based on naïve bayes and count vectorizer," in *2022 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI)*, IEEE, Jan. 2022. DOI: 10.1109/iwecai55315.2022.00039. [Online]. Available: http://dx.doi.org/10.1109/IWECAI55315.2022.00039.

[46] G. Berardi, A. Esuli, T. Fagni, and F. Sebastiani, "Multi-store metadata-based supervised mobile app classification," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, R. L. Wainwright, J. M. Corchado, A. Bechini, and J. Hong, Eds., ACM, 2015, pp. 585–588. DOI: 10.1145/2695664.2695997. [Online]. Available: https://doi.org/10.1145/2695664.2695997.

[47] S. E. Robertson and S. Walker, "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, W. B. Croft and C. J. van Rijsbergen, Eds., ACM/Springer, 1994, pp. 232–241. DOI: 10.1007/978-1-4471-2099-5_24. [Online]. Available: https://doi.org/10.1007/978-1-4471-2099-5%5C_24.

[48] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, and P. G. Bringas, "On the automatic categorisation of android applications," in *2012 IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, January 14-17, 2012*, IEEE, 2012, pp. 149–153. DOI: 10.1109/CCNC.2012.6181075. [Online]. Available: https://doi.org/10.1109/CCNC.2012.6181075.

[49] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian, "Exploiting enriched contextual information for mobile app classification," in *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October*

29 - November 02, 2012, X. Chen, G. Lebanon, H. Wang, and M. J. Zaki, Eds., ACM, 2012, pp. 1617–1621. DOI: 10.1145/2396761.2398484. [Online]. Available: https://doi.org/10.1145/2396761.2398484.

[50] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian, "Mobile app classification with enriched contextual information," *IEEE Trans. Mob. Comput.*, vol. 13, no. 7, pp. 1550–1563, 2014. DOI: 10.1109/TMC.2013.113. [Online]. Available: https://doi.org/10.1109/TMC.2013.113.

[51] F. Ebrahimi, M. Tushev, and A. Mahmoud, "Classifying mobile applications using word embeddings," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, 20:1–20:30, 2022. DOI: 10.1145/3474827. [Online]. Available: https://doi.org/10.1145/3474827.

[52] B. C. Buqing Cao, W. Z. Buqing Cao, X. X. Weishi Zhong, L. Z. Xiang Xie, and Y. Q. Lulu Zhang, "A multi-modal feature fusion-based approach for mobile application classification and recommendation," *Journal of Internet Technology*, vol. 23, no. 6, pp. 1417–1427, Nov. 2022, ISSN: 1607-9264. DOI: 10.53106/160792642022112306023. [Online]. Available: http://dx.doi.org/10.53106/160792642022112306023.

[53] J. Jiang, L. Zheng, F. Luo, and Z. Zhang, *Rednet: Residual encoder-decoder network for indoor rgb-d semantic segmentation*, 2018. arXiv: 1806.01054 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1806.01054.

[54] M. Qorich and R. E. Ouazzani, "Text sentiment classification of amazon reviews using word embeddings and convolutional neural networks," *J. Supercomput.*, vol. 79, no. 10, pp. 11 029–11 054, 2023. DOI: 10.1007/S11227-023-05094-6. [Online]. Available: https://doi.org/10.1007/s11227-023-05094-6.

[55] S. J. Putra, M. N. Gunawan, and A. A. Hidayat, "Feature engineering with word2vec on text classification using the k-nearest neighbor algorithm," in *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, IEEE, Sep. 2022. DOI: 10.1109/citsm56380.2022.9935873. [Online]. Available: http://dx.doi.org/10.1109/CITSM56380.2022.9935873.

[56] A. A. Amer and D. Elreedy, "Aggressive driver behavior detection using multi-label classification," in *18th International Conference on Ubiquitous Information Management and Communication, IMCOM 2024, Kuala Lumpur, Malaysia, January 3-5, 2024*, S. Lee, H. Choo, and R. Ismail, Eds., IEEE, 2024, pp. 1–7. DOI: 10.1109/IMCOM60618.2024.10418298. [Online]. Available: https://doi.org/10.1109/IMCOM60618.2024.10418298.

[57] P. S. Yadav, R. S. Rao, and A. Mishra, "An evaluation of multi-label classification approaches for method-level code smells detection," *IEEE Access*, vol. 12, pp. 53 664–53 676, 2024.

[58] D. Won, S. Chi, and J. O. Choi, "UAV imagery-based automatic classification of ground surface types for earthworks," en, *KSCE J. Civ. Eng.*, Mar. 2024.

[59] S. Kumar, N. Kumar, A. Dev, and S. Naorem, "Movie genre classification using binary relevance, label powerset, and machine learning classifiers," *Multim. Tools Appl.*, vol. 82, no. 1, pp. 945–968, 2023. DOI: 10.1007/S11042-022-13211-5. [Online]. Available: https://doi.org/10.1007/s11042-022-13211-5.

[60] R. K. Shah, S. Kumar, and Shashank, "Multilabel news category classification using machine learning," in *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, IEEE, Jun. 2023. DOI: 10.1109/icces57224.2023.10192826. [Online]. Available: http://dx.doi.org/10.1109/ICCES57224.2023.10192826.

[61] S. Nahak and G. Saha, "Ensembled feature based multi-label ecg arrhythmia classification," in *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, IEEE, Jul. 2023. DOI: 10.1109/embc40787.2023.10340005. [Online]. Available: http://dx.doi.org/10.1109/EMBC40787.2023.10340005.

[62] J. Divya Venkatesh, A. Jaiswal, and G. Nanda, "Comparing human text classification performance and explainability with large language and machine learning models using eye-tracking," *Scientific Reports*, vol. 14, no. 1, Jun. 2024, ISSN: 2045-2322. DOI: 10.1038/s41598-024-65080-7. [Online]. Available: http://dx.doi.org/10.1038/s41598-024-65080-7.

[63] E. H. Nfaoui and H. Elfaik, "Evaluating arabic emotion recognition task using chatgpt models: A comparative analysis between emotional stimuli prompt, fine-tuning, and in-context learning," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 19, no. 2, pp. 1118–1141, May 2024, ISSN: 0718-1876. DOI: 10.3390/jtaer19020058. [Online]. Available: http://dx.doi.org/10.3390/jtaer19020058.

[64] H. Rouzegar and M. Makrehchi, "Enhancing text classification through llm-driven active learning and human annotation," 2024. DOI: 10.48550/ARXIV.2406.12114. [Online]. Available: https://arxiv.org/abs/2406.12114.

[65] K. Tao, Z. A. Osman, P. L. Tzou, S.-Y. Rhee, V. Ahluwalia, and R. W. Shafer, "Gpt-4 performance on querying scientific publications: Reproducibility, accuracy, and impact of an instruction sheet," *BMC Medical Research Methodology*, vol. 24, no. 1, Jun. 2024, ISSN: 1471-2288. DOI: 10.1186/s12874-024-02253-y. [Online]. Available: http://dx.doi.org/10.1186/s12874-024-02253-y.

[66] Y. Qiu and Y. Jin, "Chatgpt and finetuned bert: A comparative study for developing intelligent design support systems," *Intelligent Systems with Applications*, vol. 21, p. 200 308, Mar. 2024, ISSN: 2667-3053. DOI: 10.1016/j.iswa.2023.200308. [Online]. Available: http://dx.doi.org/10.1016/j.iswa.2023.200308.

[67] N. Arici, L. Putelli, A. E. Gerevini, L. Sigalini, and I. Serina, "Llm-based approaches for automatic ticket assignment: A real-world italian application," in *Proceedings of the Seventh Workshop on Natural Language for Artificial Intelligence (NL4AI 2023)*

co-located with 22th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2023), Rome, Italy, November 6th-7th, 2023, E. Bassignana, D. Brunato, M. Polignano, and A. Ramponi, Eds., ser. CEUR Workshop Proceedings, vol. 3551, CEUR-WS.org, 2023. [Online]. Available: https://ceur-ws.org/Vol-3551/paper4.pdf.

[68] J. Coelho, A. Neto, M. Tavares, C. Coutinho, R. Ribeiro, and F. Batista, "Semantic search of mobile applications using word embeddings," in *10th Symposium on Languages, Applications and Technologies, SLATE 2021, July 1-2, 2021, Vila do Conde/Póvoa de Varzim, Portugal*, R. Queirós, M. Pinto, A. Simões, F. Portela, and M. J. Pereira, Eds., ser. OASIcs, vol. 94, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 12:1–12:12. DOI: 10.4230/OASICS.SLATE.2021.12. [Online]. Available: https://doi.org/10.4230/OASIcs.SLATE.2021.12.

[69] J. Coelho, D. Mano, B. Paula, C. Coutinho, J. Oliveira, R. Ribeiro, and F. Batista, "Semantic similarity for mobile application recommendation under scarce user data," *Eng. Appl. Artif. Intell.*, vol. 121, p. 105 974, 2023. DOI: 10.1016/J.ENGAPPAI.2023.105974. [Online]. Available: https://doi.org/10.1016/j.engappai.2023.105974.

[70] J. Coelho, A. Neto, M. Tavares, C. Coutinho, J. Oliveira, R. Ribeiro, and F. Batista, "Transformer-based language models for semantic search and mobile applications retrieval," in *Proceedings of the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2021, Volume 1: KDIR, Online Streaming, October 25-27, 2021*, R. Cucchiara, A. L. N. Fred, and J. Filipe, Eds., SCITEPRESS, 2021, pp. 225–232. DOI: 10.5220/0010657300003064. [Online]. Available: https://doi.org/10.5220/0010657300003064.

[71] *Papluca/xlm-roberta-base-language-detection · Hugging Face — huggingface.co*, https://huggingface.co/papluca/xlm-roberta-base-language-detection, [Accessed 13-03-2024].

[72] *FacebookAI/xlm-roberta-base · Hugging Face — huggingface.co*, https://huggingface.co/FacebookAI/xlm-roberta-base, [Accessed 13-03-2024].

[73] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, *Unsupervised cross-lingual representation learning at scale*, 2020. arXiv: 1911.02116 [cs.CL].

[74] B. Zhu, E. Frick, T. Wu, H. Zhu, and J. Jiao, *Starling-7b: Improving llm helpfulness & harmlessness with rlaif*, Nov. 2023.