Automatic Maturity Rating for Android Apps

Chenyu Zhou
Southern University of Science and Technology
Shenzhen, China
11930634@mail.sustech.edu.cn

Xian Zhan*
Southern University of Science and Technology, The Hong Kong Polytechnic University
Shenzhen, China
chichoxian@gmail.com

Linlin Li
Southern University of Science and Technology
Shenzhen, China
lill3@mail.sustech.edu.cn

Yepang Liu*
Southern University of Science and Technology
Shenzhen, China
liuyp1@sustech.edu.cn

ABSTRACT

Nowadays, various apps greatly facilitate children's lives and studies, while some apps also make illegal and inappropriate content (e.g., gambling, pornography) more accessible to children and adolescents. As the primary source of apps, several app markets adopt maturity ratings for apps, enabling users to distinguish whether apps are age-appropriate. However, if an incorrectly-rated app is acquired by users who are not of the appropriate age, it will bring severe consequences, especially for children. Giving an accurate maturity rating to an app can be time-consuming, both for developers and app market reviewers, while automatic rating tools can help solve this problem. Existing work on automatic app maturity ratings only analyzes app metadata obtained from app markets, but does not systematically consider the features of the apps themselves. In this work, we extract app features from both the app market and the apps themselves. We train machine learning models on Google Play, the official Android app market which has maturity ratings, and propose a cost-effective feature combination that achieves 96.98% accuracy, 96.21% precision, and 97.80% recall on within-market testing, and achieves 88.74% accuracy, 98.75% precision, and 83.72% recall on cross-market testing. Also, our method outperforms existing tools on every common metric.

CCS CONCEPTS

• **Software and its engineering**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

Maturity Rating, Mobile Apps, Android, Machine Learning

*Xian Zhan and Yepang Liu are corresponding authors. Yepang Liu is also affiliated with the Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology.

Internetware 2022, June 11-12, 2022, Hohhot, China

ACM ISBN 978-1-4503-9780-3/22/06…$15.00
https://doi.org/10.1145/3545258.3545282

ACM Reference Format:
Chenyu Zhou, Xian Zhan, Linlin Li, and Yepang Liu. 2022. Automatic Maturity Rating for Android Apps. In . ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3545258.3545282

1 INTRODUCTION

With the rapid development of communication techniques, the number of various smart devices (e.g., tablets and mobile phones) has been growing rapidly in recent years. According to Statista, the number of smartphone users worldwide is 6.648 billion in 2022, which means 83.89% of the world's population owns a smartphone [19]. Nowadays, smartphone users cover almost all age groups, even including adolescents and children. Besides, the ownership of smartphones among children is significantly increasing [8]. In 2015, 41% of 12-year-olds owned a smartphone, while in 2019, that number jumped to 69%. Different information can be easily spread to kids through diverse apps. However, not all contents (e.g., pornography, gambling, and drug use) of apps are suitable for children. Inappropriate content may have an adverse impact on kids' growth [39, 44, 57].

To prevent inappropriate content from harming kids and teens, various countries have made efforts. Since 1996, the United States has passed four related laws, including: Communications Decency Act (CDA) [7], Child Online Protection Act (COPA) [12], Children's Online Privacy Protection Act (COPPA) [2], Children's Internet Protection Act (CIPA) [3]. The essence of these laws is to differentiate between children and adults, protecting children from online content that only adults should be exposed to.

As the main channel for app distribution, app markets should help prevent children from obtaining inappropriate apps, or help parents to choose age-appropriate apps for their children. Some app markets set the maturity rating policy to distinguish the app contents for different users based on their age level. Without such a rating policy, children can easily download age-inappropriate apps, which will cause many problems or risks. For example, if children can access an in-app purchase function, wrong clicks may lead to unnecessary financial loss. Some in-app ads and analytic libraries may collect sensitive data from users and transmit them to intermediaries. Based on these data, users' profiles can be constructed, and privacy leakage can be caused through some third-party libraries in apps [25]. Moreover, COPPA has restricted rules about children's apps; some in-app purchases and ads are prohibited. As the dominant mobile OS system with the largest number of users, Android needs to own a sound maturity strategy. The official Android app

market, Google Play [14], requires developers to assign the maturity ratings for their apps and the rating authorities to review the submitted apps. However, developers need to know enough about their apps and the maturity rating strategy of the app market to give an accurate maturity rating. Rating authorities also need to examine apps to determine incorrect ratings thoroughly. These processes are time-consuming and potentially error-prone.

To solve this problem, a few studies [28, 38, 42] have proposed methods on Automatic Maturity Rating for apps. Most of the previous studies extract text information as features to train the classification model to distinguish children's and adults' apps. Hu et al. [38] extract the app descriptions as the feature to predict the maturity level by using SVM as a multi-label classifier. Chen et al. [28] proposed a text-mining-based method to predict the maturity rating based on the app's description and reviews. Chen et al. only use the keyword matching method to examine the maturity and ignore the semantic meaning of words. Furthermore, most reviews may not be related to the maturing rating and cannot offer any valuable keywords, thus will add noises for prediction. In fact, apart from the text information, apps also include many other useful features (e.g., images, icons) that can help us predict maturity. For example, the reviewers of iOS also use the features like UI screenshots, descriptions, icons to judge the maturity. Liu et al. [42] also designed an automatic maturity rating method by leveraging the machine learning classifier to identify children's apps. They choose multiple types of features as inputs, including the category of apps, the old maturity rating, title, descriptions, readability score, color distribution of the icon and screenshots, and frequency of keywords from screenshots. The authors thought the children's apps usually have s brighter color and shorter word length. However, these features cannot accurately indicate the rating policy, generating some false positives.

In this work, we aim to identify apps designed for children, which is an important sub-task of maturity rating. We attempt to conduct a thorough and systematic analysis of various types of app features, find the most effective combination of these features, and select the suitable machine learning model to identify apps for children. The main contributions of this paper are as follows:

- We thoroughly analyze different features from Android apps and try to find the most effective combination of these features. We replenish the previous research and find more valuable features in automatic maturity rating.
- Unlike most of the previous work that only focuses on apps' descriptions, we explore multi-type app features extracted from different sources. Based on the experimental result, we found some novel app features that can help better characterize apps.
- We propose a novel method to implement the automatic maturity rating to identify children's apps for the official Android app market. The evaluation results show that our method can perform better than existing tools.

2 BACKGROUND

This section gives a brief introduction to the app maturity rating and different maturity rating of different markets.

2.1 App Maturity Rating

Maturity rating has two common sub-tasks, i.e., finding suitable user and finding target user (the traditional maturity rating refers to the former one). Maturity rating classifies apps into several age ranges of users based on a uniform standard. Suitable user depends on app content, while target user is related to the functional design. Currently, there are several rating authorities to help app developers evaluate their apps' maturity ratings [18], for instance, American Entertainment Software Rating Board (ESRB) [13], Pan European Game Information (PEGI) [17] in Europe and the Middle East, Unterhaltungssoftware Selbstkontrolle (USK) in Germany [20]. For countries and territories without rating authorities, some app markets suggest developers to choose International Age Rating Coalition (IARC) [15] to rank their apps' maturity. The maturity rating policy mainly has the following usages towards app customers: 1) Helps app users, especially parents, to know potentially harmful content in an app. 2) Blocks certain content in some territories or countries to specific users based on local laws. 3) Blocks the acquisition and purchase of inappropriate content for non-supervised accounts that are not owned by adults.

2.2 Maturity Ratings in Different App Markets

Different app markets may adopt different maturity rating standards. In this paper, we mainly introduce two rating policies used by App Store (iOS) and Android Google Play.

• **App Store.** App Store has a centralized maturity rating system to manage the submitted apps. It divides the apps into four age ranges of suitable user: "4+", "9+", "12+" and "17+" [5]. App Store first requires app developers to fill a form with a list of harmful content and corresponding intensity. Then the system will generate a maturity rating for an app. Each newly submitted app will be reviewed by an expert, and the expert will adjust the inappropriate rating during the review process until the app can be released to users [4]. Also, App Store labels some children's apps, and states the age range of target user in the maturity rating of apps.

• **Google Play.** Before March 17, 2015, Google Play classified apps using the following maturity ratings, i.e., Everyone, Low maturity, Medium maturity, and High maturity. However, it has been reported that many apps were rated incorrectly due to their lax management of app content [28]. At that time, Google Play just let developers assign maturity ratings to their apps and lacked a review process. The ratings mainly depend on developers' comprehension of the maturity policy and the functionality of their apps. Now, Google Play adopts a new maturity policy. Google first requires app developers to finish a questionnaire and then assigns the maturity for their apps. Their apps will be sent to the rating authorities based on the countries and territories. In countries and territories not represented by a participating rating authority, the apps will be sent to IARC. The experts will review the maturity ratings for the submitted apps. If they find an app with an inappropriate maturity rating, they will contact the developers until adjusting the rating to a suitable one [6]. Google Play labels children's apps, and lists the apps in the Kids category page according to the different age ranges of target user.

Apart from the app markets mentioned above, to the best of our knowledge, we just find Huawei AppGallery, Samsung Galaxy

Store, Amazon Appstore, and Blackberry World have the maturity rating policy to classify age-appropriate apps for different users. Numerous third-party app markets lack a maturity rating policy to manage their apps, increasing the risk of exposing inappropriate content to children.

3 RELATED WORK

• **Automatic app maturity ratings.** Maturity ratings are designed to determine whether mobile apps include inappropriate content for kids and adolescents. Nowadays, numerous third-party app markets still lack maturity strategies. However, there is only a little research on the automatic maturity ratings for mobile apps. Chen et al. [28] systematically explored the extent and severity of unreliable maturity of apps on Google Play compared to iOS apps on Apple Store. The authors deem the maturity rating policy of Apple Store [5] is more reliable than that of Google Play [6]. Apple Store has a stricter review process for newly published apps, which requires censors to manually recheck the app ratings until the new apps become available to users. They collected the iOS apps and the counterparts of Android versions. They used the iOS apps and rating policy as the baseline. Also, they proposed a text-mining-based Automatic Label of Maturity rating (ALM) algorithm to verify whether the maturity ratings of the corresponding apps on Google Play are correct. ALM was based on keyword matching, whose keywords were manually selected from apps' descriptions and user reviews. However, their method just uses keyword matching and ignores the semantic analysis. Therefore, its result may not be accurate and cannot explain why an app has an incorrect rating. Hu et al. [38] proposed a framework named Automatic App Maturity Rating (AAMR) that can automatically label app maturity level based on the rating policy from a specific app market. The framework takes the rating policy and the app descriptions as classification features. To keep the semantic information of app descriptions, the authors leverage the deep learning techniques to extract the *word to vector model* [46, 47] from app description. They map the rating issue to be a multi-label classification problem. They leverage the SVM [1, 31] as a multi-label classifier to catch label correlations using pearson correlation [9]. Above mentioned two methods just used the text feature (i.e., the reviews and description) to rate the maturity of apps. In fact, other information of an app also can help predict maturity, such as the ad content, UI screenshots, dynamic running behaviors. In this paper, we try to identify a combination of most valuable features to help improve the robustness of the existing framework. Liu et al. [42] designed a machine learning model to predict whether an app is specifically designed for children. This model leveraged both text-based and image-based features extracted from apps as features to identify the kids' apps. The authors used their model to find 68,000 apps for kids from 1 million free Android apps. Besides, the authors also conducted a preliminary privacy analysis on these apps for kids. They found that about 10% of these apps need more attention to potential privacy leakage. The authors choose some features, like the color distribution and usage, the average length of strings in UI screenshots. Based on our investigation, these features sometimes may generate false positives and affect accuracy. The effectiveness of these features still needs further analysis.

• **Inappropriate Content Detection.** To provide kids and adolescents with a clean and safe Internet environment, more and more researchers are committed to detecting inappropriate in-app content, such as unsuitable videos, audios, text, and advertisements. To detect whether an app is appropriate for kids, Luo et al. [44] proposed an automatic content inspection framework. It dynamically tests the app to make it communicate with the server, then captures packets between them, and finally uses a third-party tool to detect inappropriate contents in pictures, audios, or videos extracted from these packets. Some research did not detect inappropriate content for mobile apps but in-app advertisements. Chen et al. [29] and Meyer et al. [45] found that in-app advertisements are common in free children's apps, of which a large percentage contain inappropriate content. The reason is that in-app advertisements are not controlled by the maturity rating policy. Bhoraskar et al. [24] developed an automation tool called Brahmastra. They leveraged Brahmastra to test 220 children's apps and found 175 of them have potential violations of the Children's Online Privacy Protection Act (COPPA) and discovered the advertisements in 36% of kids apps show inappropriate content. They also found that 80% of them attempt to collect personal information. Previous research mainly focused on inappropriate video detection. Many studies [22, 27, 32, 48, 56] focus on inappropriate video detection targeting kids on YouTube. Based on these studies, we can find that these detection technologies are also constantly improving, from traditional binary classifiers to more accurate deep learning methods. Besides, another work studied the technologies of inappropriate content inspection for texts [23, 26], audios [52], etc.

• **App security and privacy analysis.** App security and privacy analysis [25, 30, 34, 35, 37, 40–43, 49, 54] has been a hot research topic in recent years. There are also some studies focusing on children apps. To understand the children's apps' privacy protection from developers' perspectives, Ekambaranathan et al. [33] conducted semi-structured surveys with popular Android children's app developers. They found that developers largely considered that children's privacy should be respected and that data collection should be minimized. However, they also identified that the primary barrier to data protection is from in-app third-party libraries and the lack of design guidelines for age-appropriate libraries. Binns et al. [25] found that many in-app third-party libraries, especially those used by Family apps and Games & Entertainment apps, often collect sensitive data and send these data to brokers for data profiling. Reyes et al. [50, 51] presented a dynamic analysis framework to evaluate the privacy behaviors of popular free children's apps. Results showed that most have potential privacy issues, and 19% attempt to collect personal information.

4 METHODOLOGY

Figure 1 demonstrates the overview of our work. Overall, it includes two parts: a) the framework of Automatic Maturity Rating and b) the corresponding application. Besides, this framework can be further divided into five steps, i.e., dataset collection, data pre-processing and feature extraction, feature selection, model selection, and model ensemble. The following subsections will give detailed description of each part.

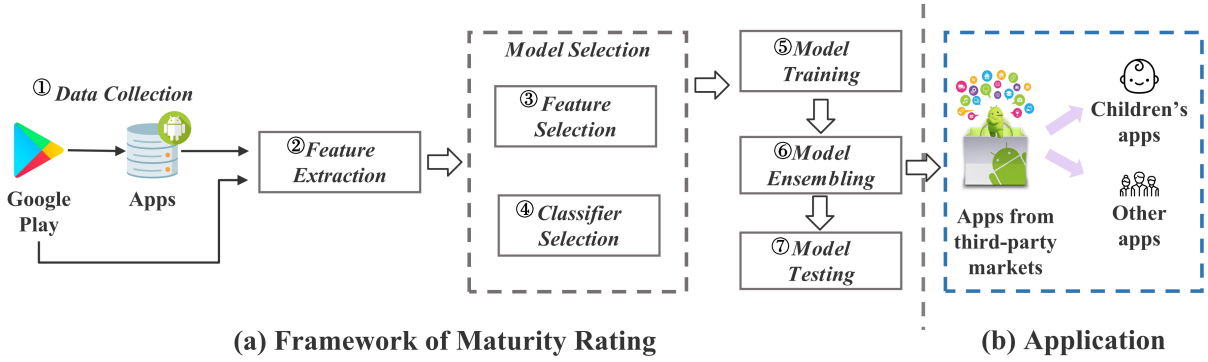

Figure 1: Overview of our work

4.1 Dataset Collection

We collect the apps from Google Play as the inputs to train our machine learning models. As the official Android app market, Google Play accepts apps from all over the world. Different countries and territories have different maturity rating authorities to review the submitted apps. The maturity standards of different authorities have some differences, which cause divergences on the age range of different maturity levels [16]. For the regions and countries without a maturity rating authority, Google Play will require IARC to rate apps. Considering that maturity rating may contribute a lot to the performance of the classification [42], we want a stable and uniformed maturity rating for Google Play's apps. As IARC is a widely-adopted maturity rating policy that rates apps for countries without rating authorities, during data collection, we select apps with IARC ratings.

We implemented a customized crawler for Google Play to download apps and their corresponding metadata, including app name, package name, app category, advertising label, maturity rating, version code, version name, release date, download volume, app description, parts of user reviews, icon, and screenshots. A naive crawling strategy is to access all apps through each app category. However, Google Play only returns a small number of apps when being accessed to each category. So we utilize the package names provided by a third-party service AndroZoo [21]. AndroZoo has kept collecting apps from several sources, including Google Play, and it records app information like the package name, sha256, version code, etc. We first extract the package names of apps derived from Google Play, then we can directly access the details page of the corresponding apps in Google Play and crawl the metadata. We got the metadata of 665,107 apps from Google Play.

Google labels apps for children and further divides them into "Ages up to 5", "Ages 6-8" and "Ages 9-12". After crawling metadata, we found 1,493 children's apps out of 665,107 apps. In this work, we conduct a binary classification task to identify apps designed for children. If an app is designed for children, we call it a positive instance, and a negative instance otherwise. As mentioned above, the proportion of positive and negative instances is unbalanced. To avoid getting skewed models that are of little practical value, we should keep the number of apps of the two classes as balanced as possible. We downloaded all children's apps and a subset of general apps. We took the top 30 apps with the largest download volume in each category to get 1,470 general apps. Thus, for a total of 2,963 apps, we downloaded their icons and all screenshots from Google's database and the corresponding version of the APK files from AndroZoo.

There are 2,963 apps in our Google Play dataset, including 1,493 positive instances and 1,470 negative instances. We randomly selected 75% of apps as the training set and the rest as the test set. The training set is used for feature selection and model selection, and we also applied 10-fold cross-validation when selecting classifiers. Then we trained the classifiers on 75% of the training data, and evaluated the combinations of models on the rest 25% of the training data. Eventually, we evaluated the effectiveness of our method by measuring the performance of several top-performing combined models on the test set.

4.2 Data Pre-Processing and Feature Extraction

After data collection, we obtained metadata, APK file, icon, and screenshots of each app in our dataset. Some of the raw data needs pre-processing or feature extraction to be transformed into a format compatible with machine learning algorithms. Also, APKs and images are complicated and have rich information. Finding the most effective features or feature combinations from these information to identify children's apps is one of the primary tasks of our work. Therefore, we first summarize the features used in existing work. We also consider some commonly-used features that were not used by previous researchers but could be used by reviewers in maturity rating decisions. Then, we explain which features we extracted and how we extracted and processed them. In later sections, we will verify the effectiveness of all these features, and try to find a proper combination of them.

Table 1 shows previous studies on maturity rating. We give their published venue, used features, and how they handle these extracted features. We can see from Table 1 that all of these studies choose the description in maturity rating classification, but different transform methods and algorithms will lead to different performances. Chen et al. [28] extract the keywords from descriptions and reviews and rate the app maturity by matching keywords. Hu et al. [38] leverage the deep learning method to keep the semantic information of description. Liu et al. [42] extract features from apps and transform these features into a vector as input for classification-based method. We can find that the method of Liu et al. [42] just

Table 1: The features used by existing maturity rating work

Previous Studies	Venue	Features	Details
Chen et al. [28]	IW3C2, 2013	description	Keywords matching.
		reviews	
Hu et al. [38]	CIKM, 2015	description	Keywords matching, word embedding, and TF-IDF.
Liu et al. [42]	Hotmobile, 2016	app category	A binary value, whether a category includes more children's apps or not.
		maturity rating	Old content rating in Google Play, i.e., Everyone, Low Maturity, Medium Maturity, High Maturity, and Unrated. The authors consider that kids usually belong to 'Everyone' and 'Low maturity'.
		title	A 5-dimensional vector, representing TF-IDF values of five keywords, i.e., 'children', 'fun', 'game', 'kid', and 'toddler'.
		description	A 10-dimensional vector, representing TF-IDF values of 10 keywords that are typical in children's apps.
		readability of description	A value indicating the readability of the description using Flesch-Kincaid.
		picture resources	A 3*49-dimensional vector representing the color distribution and usage of the icon and 2 screenshots.
		strings on screenshots	A 6-dimensional vector, of which the first 5 represent TF-IDF values of 5 keywords of children's apps, and the last one represents the average length of strings on the screenshots.

extracted some keywords of description and used limited features to represent them, while ignoring some helpful information. We can also find that some selected features may be inaccurate to represent children's apps. For example, the color distribution, hue, saturation and brightness value of screenshots cannot effectively distinguish the children's and adults' apps, leading to false positives or false negatives. Besides, using the text information alone, such as the descriptions, sometimes cannot identify whether an app is suitable for children or not. For example, COPPA requires that children's apps cannot contain ads, but app descriptions are usually explaining the main functions of an app and cannot show whether there are inappropriate in-app ads. Therefore, in this paper, we try to thoroughly analyze features from apps and use a more effective transform method for feature representation.

As shown in Table 2, we summarize the features used in previous work and some commonly-used features in other program analyses. We extract app features from several sources, including app market and APK files. We hope to find the most effective combination of features for identifying children's apps. The following part will explain why we select these features and how we transform these features into appropriate representations.

• **App Category.** We choose the app category as one of the features in identifying children's apps because some categories are designed for children or usually include more children apps, such as Education, Comics, Entertainment. Intuitively, these features can help us to distinguish children's apps to some extent. There are 32 app categories and 17 game categories in Google Play, so we convert the app category into a 49-dimensional vector. If an app belongs to a certain category, we set the corresponding dimension as 1 and the remaining dimensions as 0.

•**Maturity Rating** Some app markets provide a maturity rating for apps to indicate the age range of suitable users. Maturity ratings often delineate the lower bound of the age range, such as over 3 years old and over 12 years old, while the ground truth we use is essentially the upper bound of the age range, such as under 5 years old and under 9 years old. Although these two types of age ranges are different, they can have overlaps. We believe that apps designed for children (i.e., 3-12 years old) tend to have a maturity rating of lower than 12 years old.

• **Advertising Label.** The app markets mark whether an app includes any advertisements or not. According to COPPA regulations, the apps for children with age 3 to age 11 cannot include any advertisements or in-app purchase components. Therefore, we choose the advertising label as one of the features in the maturity rating. We use a binary value (i.e., 0 or 1) to show whether an app contains in-app advertisements and get the label from the app homepage in the app market.

• **Text Resources.** The text information usually contains rich functional information about an app. In this paper, we mainly analyze three text features, i.e., title (app name), and text in images. We do not extract features from the user reviews because most reviews are irrelevant to age-appropriate content. Using such a feature will introduce many noises in extracted features.

For these text resources, we first collect all app names, descriptions, strings on the screenshots of apps in our dataset to construct the corpus for each type of text feature. Raw data of text needs several steps of pre-processing. We split the text into the bag of words and delete stop words, punctuation, and non-ASCII words. For English letters, we convert them to lowercase and stem the words; for Chinese characters, we convert them to simplified; and for consecutive letters or numbers, we ignore them if the length is too short. We then use TF-IDF or word embedding to vectorize them. We use the pre-trained BERT model provided by Google [53] for word embedding. We will find which transform method is better in the later section.

Table 2: Our selected features for maturity rating

Data Source	Selected Features	Details
Metadata information in app market	App category	An n-dimensional vector, where n is the number of app categories in the app market. If the app belongs to a certain category, set the corresponding dimension as 1 and the remaining dimensions as 0
	Maturity rating	An n-dimensional vector, where n is the number of maturity ratings used in the app market. For our dataset, n = 5
	Advertising label	A binary value, representing whether the app includes advertisements or not
	App name	An n-dimensional vector, representing TF-IDF values of n keywords selected from all app names
	Description	Vectorize by TF-IDF or word-embedding
	Icon	**Image labels**: An n-dimensional vector, where n is the number of selected image labels that have children tendency or adult tendency. Image labels are obtained from selected images of all apps via third-party image annotation tool. **Text in images**: Vectorize by TF-IDF or word-embedding. Text are obtained from selected images from all apps via third-party OCR tool. **Color usage**: A 3-dimensional vector for each selected image, representing the average HSV values
	Screenshots	
Unzipped APK	Image files	
Manifest file of unpacked APK	Permissions	An n-dimensional vector, where n is the number of permissions we include
Source code of decompiled APK	APIs	An n-dimensional vector, where n is the number of APIs we include

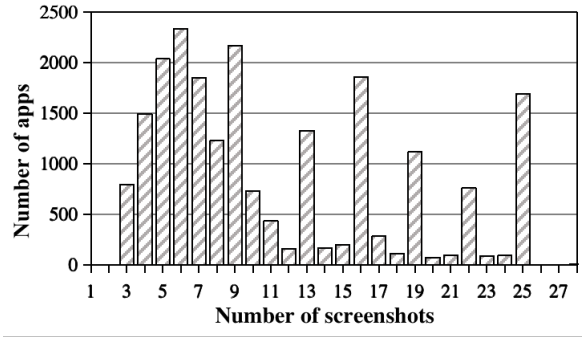

Figure 2: Distribution of screenshots count

Table 3: The number of dangerous permissions and related APIs in each API level

API level	#dangerous permissions	#dangerous APIs
23 / 6.0	24	71
24 / 7.0	24	108
25 / 7.1	24	110
26 / 8.0	26	146
27 / 8.1	26	152
28 / 9	27	162
29 / 10	29	195
30 / 11	29	218
union	30	246

• **Image Resources.** For image files, we mainly consider two types, i.e., icons and screenshots. Image resources also contain rich information to show an app's functionality, and market reviewers also leverage icons and screenshots to identify age-inappropriate content. We crawl the icons and screenshots from the app markets. There may be multiple app screenshots in an app, so we did simple filtering. One intuition is that users may inspect several screenshots before downloading an app but tend to only skim over the first few when there are many screenshots. Considering that each app may have a different number of screenshots, we counted the screenshots of all apps crawled from Google Play. The distribution is shown in Figure 2. It can be seen that the number of app screenshots fluctuates in a wide range (3-28), but apps with 6 screenshots account for the most significant proportion (11.1%). So we set the number limit of screenshots to 6.

As a result, after image filtering, we kept one icon and the first 6 screenshots (in the order as they are listed in the app market) for each app. Then, we use Google Vision API [36] for image annotation, use Tesseract-OCR [10] for text recognition, and calculate the color usage upon HSV (Hue, Saturation, and Brightness) values of pixels. After vectorizing all images, the average value of image vectors is used as app's feature.

Code features refer to the information extracted from the app source code. Permissions and the used APIs are essential code features for Android apps. We also try to verify whether these features are helpful in identifying children's apps.

• **Permissions.** Permissions are commonly used in app security and privacy analysis. In this paper, we also want to investigate how such a feature can distinguish the children' and adults' apps. Android apps need to declare all necessary permissions in the AndroidManifest.xml file. The Android system must make sure that an app has obtained the corresponding permission before running a specific API. We use AndroGuard to decompile each app and

extract its required permissions as a feature. Android classifies permissions into several protection levels: *normal* permissions do not bring risks to the privacy of users or devices, while *dangerous* permissions usually do, and *signature* permissions are automatically granted as long as signatures of the app and the permission match. The Android platform is evolving very quickly, and the permission set also varies on different Android versions. In order to ensure that our tool has high adaptability and can handle apps in different app markets, we choose not only a single version of the Android permission set, but the union of all dangerous permissions from Android 6.0 to 11 (namely, API level 23 to 30) [11, 55]. Table 3 shows the number of dangerous permissions in each version and the size of the union of dangerous permissions for all versions. We use one-hot representation for the permission feature. If the app involves a specific permission, we set the corresponding dimension to 1, and to 0, otherwise. As for which permissions to use, we will discuss in the later section.

• **APIs.** We use AndroGuard to decompile each app and analyze the app source code to obtain all the APIs used by the app. As the Android version updates quickly, the API set also varies on different Android versions. To ensure the adaptability of our method, we use the union of all dangerous APIs from Android 6.0 to 11 (namely, API level 23 to 30) as shown in Table 3. We will discuss how to select an efficient API set for our classification task in the later section.

4.3 Feature Selection

Some of the app features we involve in this work have a great number of possible values, such as text features, code features, and image features. We need to identify critical sub-features among them, or set a proper feature size to ensure the efficiency of our machine learning classifiers. We used the Chi-square test to measure the correlation between features and our ground truth, on the training set.

• **Keywords in Text features.** For text features, we can use the TF-IDF vectorizer to extract features from the corpus, but the dimension of the vectorizers, or to say, the number of keywords, needs to be designated. By calculating the Chi-square values, we determine the feature size to 100 for the app name, 120 for description, and 50 for text in images.

• **Image Labels.** From the training set apps we collected from Google Play, the Google Vision API identified a total of 1,720 distinct image labels. But not all of these labels are related to children's apps, such as "orange", "rectangle", etc. So we apply Chi-square test on all detected image labels and selected the top-130-correlated image labels.

• **Permissions.** In the training set, we found 1,905 distinct permission names. We use Chi-square test to select the top-240 permissions. We inspect several top permissions and analyze why they are related to children's apps. Here are the common reasons: it involves complex functionalities and rarely needs to be used by children's apps; or it is easy to bring risks to children and should be avoided using by children's apps. Top normal, dangerous, and third-party permissions are all highly correlated to our classification task, while signature permissions have relatively low correlation scores, probably because they are not available to general apps.

• **APIs.** There are a huge number of APIs involved in an app and need to be filtered because not all APIs are useful for our purpose. We consider combining the Term-Frequency and Chi-square, or dangerous APIs and Chi-square to select an API subset. We will use the API sets selected by these two methods to train and validate the classifiers, respectively. Based on the classifiers' performance, we will determine the method of feature selection.

4.4 Model Selection

As we consider various types of app features, there is no universally applicable machine learning algorithm for all of them. We should determine the most effective one for each type of app feature. In this work, our task is binary classification of supervised learning. We considered six classifiers commonly used in this task as candidates: Naive Bayes, Decision Tree, Logistic Regression, Linear SVM, Random Forest and Multilayer Perceptron. For each feature type, we train and validate all classifier candidates on the training set via cross validation. Besides, we also compare the performance of different encoding methods or feature sets for some feature types.

We use 75% of our collected apps as the training set to do 10-fold cross validation. And we choose the metric Area Under Curve (AUC) to evaluate the performance of models. AUC is a commonly used metric to measure the overall performance of a classifier, and can be used to rank multiple different classifiers. The results are shown in Table 4. According to the Table, we can select the optimal classifier for each type of feature. We consider not only the performance of each classifier on each type of feature, but also the complexity of those classifiers. For each feature type, we choose the classifier with the best performance and the shortest training time. The concrete process of classifier selection is as follows: First, find one or more classifiers that have the highest AUC values. To reduce the accidental factors, we set a threshold of 0.5. That is, when the difference between the AUC value of a classifier and the highest AUC value is less than 0.5, we consider that the classifier also reaches the highest AUC. Then, if only one classifier has the highest AUC value, it can be directly selected; if there are several classifiers with the highest AUC values, we select the one with the shortest training time. The results of selection are labeled by the check marks in the table.

4.5 Model Ensemble

After determining the single model for each type of app feature, we can integrate them. At present, we have known the performance of each single-type feature model for identifying children's apps. Some of them perform well and some are relatively poor. In this section, we will explore whether an ensembled model that combines multiple single-type models can further improve the performance. Intuitively, there are some complementary relationships between different types of features. As we increase the number of single models, the performance of ensemble model should gradually rise and converge. And we should finally choose one or more cost-effective model combinations. To achieve this goal, we tried all possible model combinations and evaluate them. We get trained models for each feature type using the 75% of the training set. Then we evaluate their combinations using the rest 25% of the training set. We use weighted soft voting to combine different models, in

Table 4: The best AUC value of single models (%)

Model	Feature	NB	DT	LR	SVM	RF	MLP
G	Category	93.12 ✓	93.09	93.19	86.58	93.10	93.12
A	Advertising label	58.02	58.02 ✓	58.02	58.02	58.02	58.07
R	Traditional maturity rating	64.47	64.47 ✓	64.47	64.45	64.47	64.53
N	App name - TF-IDF	84.13 ✓	80.19	84.55	79.44	82.80	83.91
T	Description - TF-IDF	87.91	78.40	87.93	86.24	87.53	87.55
T	Description - BERT	89.85	73.24	93.20	94.38 ✓	92.58	93.38
O	Text in images - TF-IDF	72.42	66.74	72.29	68.27	71.01	71.07
O	Text in images - BERT	73.79	76.07	89.05	90.9	90.78 ✓	89.06
I	Image labels	88.84	76.95	91.83	92.93 ✓	91.94	88.90
H	Average HSV	70.90	59.00	69.35	72.55 ✓	69.68	69.75
M	Permissions	93.30	85.99	94.16 ✓	93.66	94.02	94.38
D	APIs	80.95	85.49	95.81	95.6	96.53 ✓	96.17
D	Dangerous APIs	83.90	88.44	91.18	91.42	92.34	92.56

which the weight of each single model equals to the corresponding AUC value as shown in Table 4.

Since there are thousands of possible combinations for ten single-type feature models, we should filter them according to a uniform rule. We defined effective combination and efficient combinations, and their explanations are given here. In most cases, the performance of the combined model will be between the performances of its sub-models. But sometimes the performance of the combined model can be higher than the performances of all its sub-models, we called this combination an *effective* combination. We tend to have higher expectations for combined models, as their training consumes more computational resources than single-type feature models. Assuming that a combined model of two features has a performance equal to that of another single-feature model, it is evident that the single-feature model is a more efficient one. Thus, we say that an n+1-type feature model is *efficient* if its performance is higher than that of all n-type feature models.

When determining effective or efficient combined models, we only focus on a single metric, AUC, which can measure the overall performance of a model. We found 150 effective combinations and 12 efficient combinations in the Google Play training set. We plot the AUC distribution of all single, effective and efficient models in Figure 3(a), and the composition of good combinations can be seen in Figure 3(b).

5 EVALUATION

In this section, we evaluate our approach by answering the following four research questions.

- **RQ1: (Performance of the ensembled models):** How is the performance of the ensembled models? Which single-type models can be combined to form good ensembled models?
- **RQ2: (Comparison with the state of the art):** How effective is our tool compared to the existing tools?
- **RQ3: (Usefulness of our tool):** How does our tool perform in the cross-market application scenario?

5.1 Experimental Setup

In RQ1, the experiment is conducted on the training dataset of Google Play. In RQ2, we select several good ensembled models and compare them with the existing approaches. The machine learning models are trained on Google Play's training set apps, and tested on Google Play's test set apps. In RQ3, we regard Huawei AppGallery dataset as the test dataset measuring the models trained on the Google Play dataset, to evaluate our tool's performance in the cross-market application scenario. The three experiments are conducted on a server with 4 Intel(R) Xeon(R) Gold 6238 CPUs @ 2.10GHz and 256 GB RAM.

5.2 Results and Analyses

5.2.1 RQ1: Performance of the Ensembled Models. From Figure 3, we can see that the performance of the effective combination gradually converges as the number of combined single type features increases. The efficient models only occur in 2-type combined models, because the best 2-type model has reached an extremely high performance and none of subsequent effective models can outperform it. However, it may have been overfitted, so we will also consider several other effective models.

According to Figure 3(b), DT (APIs + description) is the most effective emsembled model on the training dataset of Google Play. To avoid overfitting, we also consider the best model among the combined models at each combination level. As a result, we select the model DT, DIR, DGAI, MDGNI, MDGAIH, and MDGANIR as candidates. In the RQ2, we will show their evaluation on the Google Play test set.

On the other hand, considering that the app category does not actually belongs to the features of the app itself, but the features given by the app market that vary across different app markets. Some app markets require app developers to assign the maturity rating and app category before submitting the app to the market, and then there may be a review mechanism. In the app market with manual review, the maturity rating and app category designated by the developer should be the object of review, rather than the established features of the app itself. That is, imagine that if our tool being used to help app reviewers in app markets to verify that

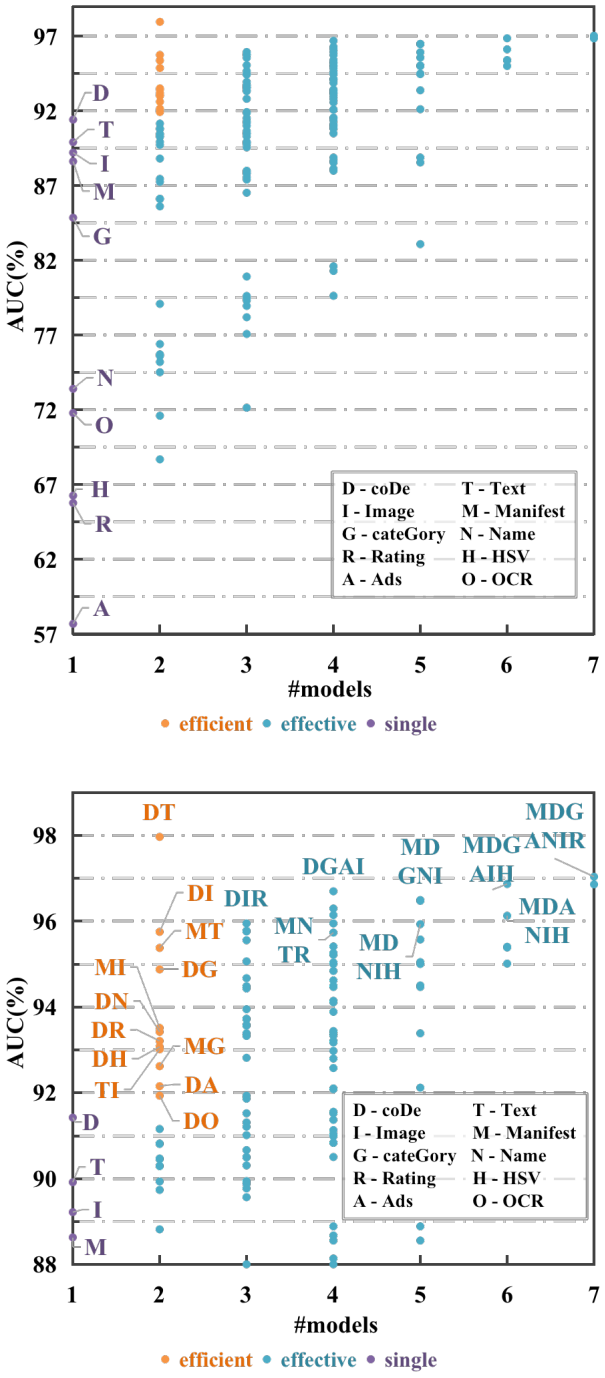

Figure 3: The AUC distribution of the effective and efficient ensemble models

Table 5: The comparison with existing work on within-market testing (%)

Model	Accuracy	Precision	Recall (TPR)	TNR	F1 score	AUC
AAMR	82.85	85.00	79.61	86.07	82.22	82.84
Liu et al.	88.61	87.63	89.81	87.43	88.71	88.62
DT	93.55	91.80	95.59	91.53	93.66	93.56
DIR	96.98	96.21	97.80	96.17	96.99	96.99
DGAI	96.30	96.41	96.14	96.45	96.28	96.30
MDGNI	96.57	95.92	97.25	95.90	96.58	96.57
MDGAIH	96.71	96.19	97.25	96.17	96.71	96.71
MDGANIR	97.12	96.47	97.80	96.45	97.13	97.12

an app is correctly age-rated, the app's category should not be used as a feature. Therefore, we hope to find a model whose performance is not lower than that of the simple category model, but does not contain category features, to evaluate whether our method can work well on another app market instead of Google Play. We choose the model DT, DIR, MNTR, MDNIH, and MDANIH as candidates and will evaluate them in RQ3 for cross-market application scenario.

5.2.2 RQ2: Comparison with the State of the Art. As mentioned before, we select the best-performing ensembled models of all combination levels, including DT, DIR, DGAI, MDGNI, MDGAIH, and MDGANIR. In this section, we compare them with two existing methods: Hu et al.[38] and Liu et al.[42].

In this research question, we conducted experiments based on the Google Play dataset. Machine learning models are trained on Google Play's training set apps, and tested on Google Play's test set apps. We compared the candidate models using six metrics: accuracy, precision, recall (also the true positive rate, TPR), TNR (true negative rate), F1 score, and AUC. As shown in Table 5, our models are superior to the existing methods on every metric. According to the performance on test set, the model MDGANIR (permissions + APIs + category + ads + name + image labels + maturity rating) is the best-performing one. Also, we found that the model DIR, with only three models combined, has a similar performance to the best-performing model. So we regard DIR (APIs + image labels + maturity rating) as the most cost-effective model. Except for the model DIR, as the combined level of the combined model increases, the performance of the model on the test set also increases. The result shows that our chosen features and transform methods are effective, compared to existing work. We can find that AAMR gets the lowest scores on all six metrics. Also, it has higher TNR than TPR, which means it has relatively poor accuracy in identifying children's apps than identifying non-children's apps. This is reasonable because this method was initially designed for identifying apps not suitable for children. It makes decisions mainly based on the sensitive words in the app description. So after being applied to identifying apps designed for children, it has the worst performance among all candidate models, with higher TNR than TPR. Meanwhile, Liu et al.'s method and most of our models have higher TPR than TNR. This is understandable because these models mainly focused on identifying apps designed for kids. The features they use mainly focus on those that are typical in children's apps, but care less about non-children's apps.

Table 6: The comparison with existing work on cross-market testing (%)

Model	Accuracy	Precision	Recall (TPR)	TNR	F1 score	AUC
AAMR	83.86	88.46	77.35	90.19	82.53	83.77
Liu et al.	83.77	90.41	75.04	92.25	82.01	82.65
DT	83.77	81.43	86.90	80.72	84.08	83.81
DIR	88.74	92.75	83.72	93.63	88.00	88.67
MNTR	70.24	95.16	41.77	97.93	58.06	69.85
MDNIH	77.23	93.93	57.52	96.37	71.35	76.95
MDANIH	77.49	93.73	58.23	96.21	71.83	77.22

5.2.3 RQ3: Usefulness of Our Tool. We hope that our method can be used across different app markets, so we use Huawei AppGallery, a third-party app market with "Children" category, to evaluate the models' performance when being adopted across app markets. In this section, the candidate models are different from those used in the former section. We need to apply the models to different app markets, but there's no one-to-one correspondence between categories of different app markets. Also, as Huawei AppGallery classified children's apps into a separate category, app categories cannot be used as a feature for Huawei AppGallery's apps. Under this premise, we select the best models for every level of the combined model without category feature, as labeled in Figure 3(b). Besides, we use Google Translate to translate the content of text features of Huawei's apps to English. The evaluation results are shown in Table 6.

Through comparing Table 5 and Table 6, we find that our models DT and DIR have about 10% performance degradation in the cross-market application scenario. By contrast, Liu et al.'s model drops by 6%, and AAMR even improves by 1%. In the cross-market application scenario, the 2-type model DT and the two baselines result in a similar performance. Our 3-type model, DIR (APIs + image labels + maturity rating), performs the best among all candidate models, outperforming both baselines on all the six metrics. However, the performance of the 4-, 5-, and 6-type models degrades greatly due to the low accuracy in identifying positive instances. We further analyze why the performances decrease, and the reasons are as follows: 1) Different ground truth between app markets; 2) Features are not feasible for cross-market application, or have overfitted on the training set; 3) Cultural differences between countries; 4) Translation introduces noise.

6 THREATS TO VALIDITY

Internal Validity. We choose apps from Google Play as the basis to train our automatic rating models. However, previous work has reported that Google Play may incorrectly rate apps [28]. Google Play already improved its rating policy in 2015. To mitigate the threat, we only collected apps that were released after the rating policy was updated. Also, we conducted cross-validation of these selected apps in our experiment to guarantee that the maturity ratings were as accurate as possible.

Construct Validity. The selected data has a significant impact on our experiment results. To eliminate the effects of data selection on model selection and training, the choice of parameters for machine learning models, three people of us manually inspected apps in Google Play to constructed the dataset. And for each training process, we conducted 10-fold cross-validation.

7 LIMITATIONS AND FUTURE WORK

In this paper, we designed a machine learning model for predicting apps for kids. We just used some static features from apps. In the future, we can add some dynamic features into our model to improve robustness, such as the app behaviors and the dynamically loaded ad content. We just implemented a binary classification. In the future, we will try to add multi-label classification that can predict multiple maturity ratings, which is more practical for real-world applications. We will also try to incorporate different maturity policies into our framework to enhance the generality and robustness of our framework.

Nowadays, many functionalities can only be implemented through third-party libraries [33]. However, in-app advertisements and other third-party libraries are not controlled by the maturity rating policy. The app developers may ignore whether the in-app ads violate the content policy. A promising research direction can focus on the in-app ad contents detection or identification of the behaviors of other third-party libraries that violate the maturity policy. Such a study could be useful, which can help enhance privacy protection.

Another research direction is the automatic maturity rating of the Android third-party app market. While there are many third-party app markets for Android, most of them do not include a maturity rating mechanism. We hope to develop an automatic maturity rating tool that can be widely used in different app markets to help app markets and app users deal with the chaos in third-party app markets.

8 CONCLUSION

Maturity rating in app markets can prevent inappropriate content in apps from negatively impacting minors. Manually performing the classification (either by developers or by reviewers) is time-consuming and error-prone, and should be aided by automatic rating tools. However, existing tools only leverage simple features obtained from the app markets, while ignoring many useful app features.

Therefore, we comprehensively explored the performance of various types of app features on this classification problem, including metadata, manifest-based features, code-based features, and resource-based features. We constructed a dataset based on the maturity ratings of the official Android app market, Google Play, and trained machine learning models to identify children's apps. According to the experiment results, we finally found a most cost-effective multi-type feature model, which takes app category, description, and image annotation of screenshots as features. We further compared the performance of this model and several existing tools. For the within-market testing, our model achieved 96.98% accuracy, 96.21% precision, and 97.80% recall, outperforming existing tools on multiple metrics.

ACKNOWLEDGMENTS

We would like to thank Internetware 2022 reviewers for their comments and suggestions, which helped improve this paper. This work is partially supported by the Guangdong Basic and Applied Basic Research Fund (Grant No. 2021A1515011562), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), and the National Natural Science Foundation of China (Grant No. 61802164).

REFERENCES

[1] 1995. SVM. https://en.wikipedia.org/wiki/Support-vector_machine.
[2] 1998. Children's Online Privacy Protection Rule ("COPPA"). https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule.
[3] 2000. Children's Internet Protection Act. http://ifea.net/cipa.pdf.
[4] 2012. App Store Review Guidelines. https://developer.apple.com/app-store/review/guidelines/.
[5] 2012. Apple Application Ratings. http://itunes.apple.com/WebObjects/MZStore.woa/wa/appRatings.
[6] 2012. Google Application Ratings. https://support.google.com/googleplay/answer/6209544.
[7] 2017. Communications Decency Act of 1996. http://www.cybertelecom.org/cda/cda.htm.
[8] 2019. The Common Sense Census: Media Use by Tweens and Teens, 2019. https://www.commonsensemedia.org/research/the-common-sense-census-media-use-by-tweens-and-teens-2019.
[9] 2021. Pearson correlation coefficient. https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.
[10] 2021. Tesseract OCR. https://github.com/tesseract-ocr/tesseract.
[11] 2022. APER-mapping. https://github.com/sqlab-sustech/APER-mapping.
[12] 2022. Children's Online Protection Act COPA. https://www.law.cornell.edu/uscode/text/47/231.
[13] 2022. ESRB. https://www.esrb.org/.
[14] 2022. Google Play. https://play.google.com.
[15] 2022. International Age Rating Coaltion. https://www.globalratings.com/ratings-guide.aspx.
[16] 2022. Mobile software content rating system. https://en.wikipedia.org/wiki/Mobile_software_content_rating_system.
[17] 2022. PEGI. https://pegi.info/.
[18] 2022. Rating authorities. https://support.google.com/googleplay/android-developer/answer/9859655#ratings.
[19] 2022. Statista, smartphone usage. https://www.bankmycell.com/blog/how-many-phones-are-in-the-world.
[20] 2022. USK. https://usk.de/alle-lexikonbegriffe/iarc/.
[21] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories* (Austin, Texas) *(MSR '16)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2901739.2903508
[22] Camila Souza Araújo, Gabriel Magno, Wagner Meira, Virgilio Almeida, Pedro Hartung, and Danilo Doneda. 2017. Characterizing videos, audience and advertising in Youtube channels for kids. In *International Conference on Social Informatics*. Springer, 341–359.
[23] Gonzalo Molpeceres Barrientos, Rocío Alaiz-Rodríguez, Víctor González-Castro, and Andrew C Parnell. 2020. Machine Learning Techniques for the Detection of Inappropriate Erotic Content in Text. *International Journal of Computational Intelligence Systems* 13, 1 (2020), 591–603.
[24] Ravi Bhoraskar, Seungyeop Han, Jinseong Jeon, Tanzirul Azim, Shuo Chen, Jaeyeon Jung, Suman Nath, Rui Wang, and David Wetherall. 2014. Brahmastra: Driving apps to test the security of third-party components. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. 1021–1036.
[25] Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. 2018. Third Party Tracking in the Mobile Ecosystem. In *Proc. WebSci '18*. 23–31. https://doi.org/10.1145/3201064.3201089
[26] Eric Brewer and Yiu-Kai Ng. 2019. Age-Suitability Prediction for Literature Using a Recurrent Neural Network Model. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 1592–1596.
[27] Marina Buzzi. 2011. Children and YouTube: access to safe content. In *Proceedings of the 9th ACM SIGCHI Italian Chapter International Conference on Computer-Human Interaction: Facing Complexity*. 125–131.
[28] Ying Chen, Heng Xu, Yilu Zhou, and Sencun Zhu. 2013. Is this app safe for children? A comparison study of maturity ratings on Android and iOS applications. In *Proceedings of the 22nd international conference on World Wide Web*. 201–212.
[29] Ying Chen, Sencun Zhu, Heng Xu, and Yilu Zhou. 2013. Children's exposure to mobile in-app advertising: an analysis of content appropriateness. In *2013 International Conference on Social Computing*. IEEE, 196–203.
[30] Saksham Chitkara, Nishad Gothoskar, Suhas Harish, Jason I. Hong, and Yuvraj Agarwal. 2017. Does This App Really Need My Location? Context-Aware Privacy Management for Smartphones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 42 (sep 2017), 22 pages. https://doi.org/10.1145/3132029
[31] Corinna Cortes and Vladimir Vapnik. 1995. Support-Vector Networks. 20, 3 (sep 1995), 273–297. https://doi.org/10.1023/A:1022627411411
[32] Carsten Eickhoff and Arjen P. de Vries. 2010. Identifying Suitable YouTube Videos for Children. In *Networked and electronic media summit*.
[33] Anirudh Ekambaranathan, Jun Zhao, and Max Van Kleek. 2021. "Money Makes the World Go Around": Identifying Barriers to Better Privacy in Children's Apps From Developers' Perspectives. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Article 46, 15 pages. https://doi.org/10.1145/3411764.3445599
[34] Adrienne Porter Felt, Helen J Wang, Alexander Moshchuk, Steve Hanna, and Erika Chin. 2011. Permission Re-Delegation: Attacks and Defenses.. In *USENIX security symposium*, Vol. 30. 88.
[35] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. 2012. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In *International Conference on Trust and Trustworthy Computing*. Springer, 291–307.
[36] Google. 2021. Vision AI. https://cloud.google.com/vision/.
[37] Michael C. Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. 2012. Unsafe Exposure Analysis of Mobile In-App Advertisements. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks* (Tucson, Arizona, USA) *(WISEC '12)*. Association for Computing Machinery, New York, NY, USA, 101–112. https://doi.org/10.1145/2185448.2185464
[38] Bing Hu, Bin Liu, Neil Zhenqiang Gong, Deguang Kong, and Hongxia Jin. 2015. Protecting your children from inappropriate content in mobile apps: An automatic maturity rating framework. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1111–1120.
[39] Chang-Hyun Jin. 2013. The effects of individual innovativeness on users' adoption of Internet content filtering software and attitudes toward children's Internet use. *Computers in Human Behavior* 29, 5 (2013), 1904–1916. https://www.sciencedirect.com/science/article/pii/S0747563213000988
[40] Ilias Leontiadis, Christos Efstratiou, Marco Picone, and Cecilia Mascolo. 2012. Don't Kill My Ads! Balancing Privacy in an Ad-Supported Mobile Application Market. In *Proc. HotMobile '12*. Article 2, 6 pages.
[41] Jialiu Lin. 2013. *Understanding and Capturing People's Mobile App Privacy Preferences*. Ph. D. Dissertation. Advisor(s) Sadeh, Norman and Hong, Jason I.
[42] Minxing Liu, Haoyu Wang, Yao Guo, and Jason Hong. 2016. Identifying and analyzing the privacy of apps for kids. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*. 105–110.
[43] Xing Liu, Jiqiang Liu, Sencun Zhu, Wei Wang, and Xiangliang Zhang. 2020. Privacy Risk Analysis and Mitigation of Analytics Libraries in the Android Ecosystem. *IEEE TMC* 19, 5 (2020), 1184–1199.
[44] Qian Luo, Jiajia Liu, Jiadai Wang, Yawen Tan, Yurui Cao, and Nei Kato. 2020. Automatic content inspection and forensics for children android apps. *IEEE Internet of Things Journal* 7, 8 (2020), 7123–7134.
[45] Marisa Meyer, Victoria Adkins, Nalingna Yuan, Heidi M Weeks, Yung-Ju Chang, and Jenny Radesky. 2019. Advertising in young children's apps: A content analysis. *Journal of developmental & behavioral pediatrics* 40, 1 (2019), 32–39.
[46] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[47] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[48] Kostantinos Papadamou, Antonis Papasavva, Savvas Zannettou, Jeremy Blackburn, Nicolas Kourtellis, Ilias Leontiadis, Gianluca Stringhini, and Michael Sirivianos. 2020. Disturbed YouTube for kids: Characterizing and detecting inappropriate videos targeting young children. In *Proceedings of the international AAAI conference on web and social media*, Vol. 14. 522–533.
[49] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. 2018. Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem. In *NDSS*.
[50] Irwin Reyes, Primal Wijesekera, Abbas Razaghpanah, Joel Reardon, Narseo Vallina-Rodriguez, Serge Egelman, Christian Kreibich, et al. 2017. " Is Our Children's Apps Learning?" Automatically Detecting COPPA Violations. In *Workshop on Technology and Consumer Protection (ConPro 2017), in conjunction with the 38th IEEE Symposium on Security and Privacy (IEEE S&P 2017)*.
[51] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. 2018. "Won't somebody think of the children?" examining COPPA compliance at scale. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 63–83.
[52] Dan Stowell, Dimitrios Giannoulis, Emmanouil Benetos, Mathieu Lagrange, and Mark D Plumbley. 2015. Detection and classification of acoustic scenes and

events. *IEEE Transactions on Multimedia* 17, 10 (2015), 1733–1746.
[53] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv preprint arXiv:1908.08962v2* (2019).
[54] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Abbas Razaghpanah, Rishab Nithyanand, Mark Allman, Christian Kreibich, and Phillipa Gill. 2016. Tracking the Trackers: Towards Understanding the Mobile Advertising and Tracking Ecosystem. In *arXiv preprint arXiv:1609.07190*.
[55] Sinan Wang, Yibo Wang, Xian Zhan, Ying Wang, Yepang Liu, Xiapu Luo, and Shing-Chi Cheung. 2022. Aper: Evolution-Aware Runtime Permission Misuse Detection for Android Apps. arXiv:2201.12542 [cs.SE]
[56] Jônatas Wehrmann, Gabriel S Simões, Rodrigo C Barros, and Victor F Cavalcante. 2018. Adult content detection in videos with convolutional and recurrent neural networks. *Neurocomputing* 272 (2018), 432–438.
[57] Kan Yang, Qi Han, Hui Li, Kan Zheng, Zhou Su, and Xuemin Shen. 2017. An Efficient and Fine-Grained Big Data Access Control Scheme With Privacy-Preserving Policy. *IEEE Internet of Things Journal* 4, 2 (2017), 563–571.