The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

PROTECT CHILDREN ONLINE SAFETY ON SOCIAL MEDIA AND MOBILE PLATFORMS

A Dissertation in

Computer Science and Engineering

by

Ying Chen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2014

The dissertation of Ying Chen was reviewed and approved* by the following:

Sencun Zhu
Associate Professor of Computer Science and Engineering & Information Sciences and Technology
Dissertation Co-Adviser
Co-Chair of Committee

Heng Xu
Associate Professor of Information Sciences and Technology
Dissertation Co-Adviser
Co-chair of Committee

Wang-Chien Lee
Associate Professor of Computer Science and Engineering

Alan MacEachren
Professor of Geography

Lee Coraor
Associate Professor of Computer Science and Engineering
Director of Academic Affairs

*Signatures are on file in the Graduate School

ABSTRACT

Recent advances in social media and smart mobile technologies make ubiquitous information and communication environments more of a technical reality than a distant vision. The sweeping popularity of these information and communication environments also affects children and adolescents (later referred as "children"). There is a dearth of systematic scholarly work examining those threats for children's protection. In addressing this void, the goal of our research is to explore and understand the risks and threats of such information and communication environments on children. In particular, our primary research interests lie in the three main threats to children's online safety (Palfrey et al., 2008): online harassment and cyberbullying, exposure to problematic contents, sexual solicitation and Internet-initiated offline encounters.

Online harassment and cyberbullying mostly occur in social media, and always involve offensive language. Since the textual contents on online social media are highly unstructured, informal, and often misspelled, existing research on message-level offensive language detection cannot accurately detect offensive content. Meanwhile, user-level offensiveness detection is an under researched area. We propose the Lexical Syntactic Feature (LSF) architecture to detect offensive contents and identify cyber bullies. Results from experiments showed that the LSF framework achieves accuracy of 96.6%, 77.8% in sentence and user offensiveness detection respectively. Besides, the processing speed of LSF is 10 ms per sentence, suggesting its potential for effective deployment in social media.

Problematic contents and solicitation mostly appear on mobile platforms. Since people are heavily relying on the search results and rankings on app stores to explore new apps, on the market side, we must help parents easily choose appropriate apps for their kids with minimal online threats. Since we concern about problematic contents and cyber solicitation, and posting personal information and interacting with online strangers are the two major types of behaviors expose children to cyber solicitation (Ybarra et al., 2007), we monitor two types of risks on mobile devices: content risks and privacy risks.

For content risks, we develop automatic mechanisms to detect problematic contents and verify the content ratings of mobile apps and in-app ads. The results show that 27.8% of Android apps have unreliable content ratings. In addition, a large percent of the in-app advertisements carry inappropriate contents for children, and 35.9% of Android in-app ads and 38.9% of iOS in-app ads are found exceeding the host apps' content ratings. For privacy risks, we adopt the contextual integrity theory (Nissenbaum, 2009) to develop quantitative measures of privacy risks on mobile apps. We also propose an automatic system with user interface to assist parents being aware of apps' privacy risks, therefore, to make informed decisions when choosing apps for their children. We believe that the findings have important implications for the legal and educational departments, social medias, platform providers (e.g., Google or Apple), as well as for regulatory bodies and application developers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

I greatly thank my committee members for their valuable comments on this dissertation. Especially, I am grateful that I have two wonderful advisers: Dr. Sencun Zhu and Dr. Heng Xu. They not only provide me with great advices for years, but also support me financially. They are great mentors. When I first came to the program, I was a naïve new grad B.E., had no idea how to do research and how to publish papers. They helped me, educated me, and guided me along the way to become an independent researcher.

In addition, I need to thank Dr. Yilu Zhou, who is an associate professor in the Fordham Schools of Business. Her research interests include business intelligence, web/text/data mining, multilingual knowledge discovery, and human-computer interaction. She was introduced to me by Dr. Xu in my third year in the PhD program. She is a great and patient mentor, a friend, a professional researcher, and a salesman. She teaches me how to sale and motivate my works. With her and my advisers' helps, I published my first full paper in 2012. To me, she is my third adviser. I believe we will continue collaborating since the collaboration is such a rewarding, educational, and enjoyable experience.

Furthermore, I want to thank two students: Feng Deng and Chen Chen. Feng Deng is a third year undergraduate student in the Electrical Engineering department of the Pennsylvania State University, while Chen Chen is a master student in the Computer Science and Engineering department of the Pennsylvania State University. They implement the app privacy rating system and the user interface described in the cyber-solicitation chapter. Feng Deng, especially, contributes the most. With their devoted efforts, currently the system can automatically download mobile apps and analyze the privacy risks. We plan to further polish the system, and to do some user studies. We hope eventually the system can largely help parents understand the privacy risks of mobile apps, and assist them finding the ones with minimum privacy risks.

Last but not least, I want to thank all the reviewers for their valuable comments on my papers. They spent much time reading my works, correcting wording issues, giving me valuable suggestions and revising advices. Based on their precious inputs, I could finally get my work published and shared with scholars all over the world.

Chapter 1 Introduction

1.1 Motivation

Bullying, problematic content, and solicitation are parents' nightmares since the old days. When experienced bullying, problematic content, or solicitation, despite of the physical harms, children are suffered psychologically and found 1) becoming low self-esteem, afraid of outside world; 2) becoming "immune" or numb to the horror of abusive language and behavior; 3) gradually accepting abusive language and behave as a way to solve arguments; and 4) imitating the offensive language and even the behavior.

When the internet and mobile becomes popular, threats come from all over the world. The problems are no longer local. Parents can no longer protect their children by talking to other parents and the neighborhood kids, or by coming to the school and talking to the teachers. As reported by CNN (CNN, 2013):

- 95% teenagers are Internet users, comparing to 78% adults;
- 80% teenagers are social network users, comparing to 69% adults;
- 69% teenagers reported that their peers are “mostly kind” on social networks, comparing to 85% adults;
- 20% teenagers reported that their peers are “mostly unkind” on social networks, comparing to 5% adults;
- as to teenagers' peers, 12% of them are reported as “frequently being mean or cruel”, 29% of them are reported as “sometimes being mean or cruel”, 47% of them are reported as “once in a while being mean or cruel”, 11% of them are reported as “never being mean or cruel”, while in adults, the ratios are 7%, 18%, 44%, and 29% respectively.

There are more statistics reflecting that children and adolescents are suffered from cyberbullying, problematic content, and cyber-solicitation. It has been found that 19% of teens report that someone has written or posted mean or embarrassing things about them on social networking sites (Johnson et al., 2011). In a limited, small-scale analysis of chat transcripts from two of the most popular teen sites, chat participants had 19% chance exposing to negative racial or ethnic remarks in monitored chat and 59% chance in unmonitored chat (Tynes et al., 2004). Wolak et al (Wolak et al., 2006) found that one in seven kids received at least one online sexual solicitation in 2006. Therefore, it is critical to protect children and adolescents from cyberbullying, problematic content, and cyber-solicitation.

1.2 Challenges and Research Goals

Different entities have been aware of the situation and made efforts from education and legal perspective to protect children and adolescents. There are numbers of online websites such as NetSmarz, Teachtoday, iKeepSafe, WiredSafety, Teenangels, girlscounts and PBS kids that educate parents and adolescents how to recognize and react on cyber threats. Additionally, the Children's Internet Protection Act (CIPA) was enacted in early 2001 to address concerns on children's access to visual offensive content over Internet. The Children's Online Privacy Protection Act (COPPA), effective April 21, 2000, requires website operators must seek verifiable consent from a parent or guardian before collecting personal information from children under 13 years old.

Unfortunately, the implementations of the legal protections are disappointing. Social media, where cyberbullying mostly occurs, is full of user-generated unstructured content. To comply with CIPA requirements, administrators of social media have to manually review online contents to detect and delete offensive materials. However, the manual review tasks of identifying offensive contents are labor intensive, time consuming, and thus not sustainable and scalable in reality. Some automatic content filtering software packages, such as Appen[1] and Internet Security

---

[1] http://www.appen.com.au/index.cfm?pageid=103

Suite[2], have been developed to detect and filter online offensive contents. Most of them simply blocked webpages and paragraphs that contained dirty words. These word-based approaches not only affect the readability and usability of web sites, but also fail to identify subtle bullying messages. Therefore, automatic method to effectively detect cyberbullying is critical to protect children and adolescents on social media.

In addition, none of these legal regulations have considered mobile devices, which have become the major platform for children to connect to the Internet. Children have increasingly become fans of mobile devices. Zact, a mobile phone services company, surveyed US parents in April 2013 (eMarketer) on the smartphone habits of their children and found 25% children between 2 and 5, 39% between 6 and 9, and over 50% between 10 and 17 use smartphones. 58% of parents allowed their child to scroll and swipe smartphones for 1 to 4 hours per day. A more recent survey commissioned by PBS Kids (Forbes, 2013) found that more than 36% of parents say they plan to purchase smartphones or tablets for their kids as 2013 winter holiday gifts. In addition, nearly 70% of parents plan to give their kids mobile apps and 90% of parents believe that educational apps will play an important role in children's learning in the future.

Since mobile devices have been proven fun and educable for children to use, denying the young generation to access mobile devices is no longer a viable way to protect children from being hurt by online parties. Parents need greater awareness of privacy and safety as well as appropriate advice when choosing apps for their kids (Forbes, 2013). According to a multi-choice survey (readwrite, 2009), 62% of people discover new apps by searching in app stores, and 60% of people discover new apps by browsing through top app store rankings. In other words, people are heavily relying on the search results and rankings on app stores to explore new apps. Therefore, for children's safety on using mobile devices, we believe that there is one critical and

---

[2] http://shop.ca.com/malware/internet_security_suite.aspx?ggus=36640429&gclid=CJ3LhJmsnZ8CFdA65QodnV5BRQ

urgent need. That is, on the market side, we must help parents easily choose appropriate apps for their kids with minimal online threats. These are the exact research goals for this dissertation.

1.3 Research Plans

This dissertation is composed of three parts. The first part of this dissertation is on cyber bullying issues pertaining to the adoption of social media. We propose a text mining mechanism called Lexical Syntactic Feature (LSF) architecture to detect offensive contents and identify potential cyber-bullies. Specifically, we introduce syntactic rules in identifying name-calling harassment, and incorporate users' writing styles, structures, and specific language features to predict the potential of individuals to send out offensive contents. Results from the experiments showed that our LSF framework performed significantly better than existing methods in offensive content detection. It achieves precision of 98.24% and recall of 94.34% in sentence offensiveness detection, as well as precision of 77.9% and recall of 77.8% in user offensiveness detection. Meanwhile, the processing speed of LSF is approximately 10 ms per sentence, suggesting the potential for effective deployment in social media.

The second part of this dissertation is about mature content detection on smart devices. While movie and video game industries have their official content rating organization such as MPAA and ESRB, mobile apps do not. Instead of having standard rating rules across platforms, each mobile platform established its own rating policy and rating strategy. There is much skepticism about the effectiveness of platform self-regulation in protecting children's mental health, which has resulted in parents clamoring for an accurate and straightforward standard to monitor mature contents in apps generated by third party developers. From a theoretical perspective, our research examines the insufficiency of rating policies and prompts the development of a universal standard. From a practical perspective, it develops automatic mechanisms to derive app maturity from the descriptions, highlights several important implications for various social factors in affecting the correctness of app maturity ratings, including price, platform, and privacy policy of developer's country. The maturity of in-app ads are also quantifiably classified and examined.

The last part of this dissertation is to prevent online sexual solicitation and internet-initiated offline encounters. The Federal Trade Commission (FTC) published a report[3] in 2011 examining the privacy disclosures and practices of mobile apps offered for 0~13 years old children in the Google Play and Apple iTune app stores. It shows currently mobile platforms violate the Children's Online Privacy Protection Act (COPPA). Parents have little clue to determine whether personal information has been collected from their children. Hence, we adopt Nissenbaum's contextual integrity theory to examine the information boundary on mobile platforms and guide the development of automatic algorithms quantifying mobile applications' privacy risks.

The automatic mechanisms developed in this thesis provide parents more insights about children's online risks. We believe that the findings have important implications for the legal and educational departments, social medias, platform providers (e.g., Google or Apple), as well as for regulatory bodies and application developers.

---

[3] Mobile Apps for Kids: Disclosures Still Not Making the Grade: Available at http://www.ftc.gov/os/2012/12/121210mobilekidsappreport.pdf

Chapter 2 Background and Related Work

As discussed above, the research goal of this dissertation is to protect children and adolescents from cyberbullying, problematic content, and cyber-solicitation. In particular, we aim to develop automatic method to effectively detect cyberbullying on social media, as well as to help parents easily choose appropriate apps for their kids with minimal online threats. In this chapter, we review the backgrounds and related works.

2.1 Cyberbullying

Cyberbullying mostly involves name calling, which is offensive, abusive, or insulting language, referring to a person or group. Therefore, to detect cyberbullying, the most critical and urgent need is to detect name calling (aka “offensive language”). While there is no universal definition of "offensive," in this study we employ Jay and Janschewitz’s (2008) definition of offensive language as vulgar, pornographic, and hateful language. Vulgar language refers to coarse and rude expressions, which include explicit and offensive reference to sex or bodily functions. Pornographic language refers the portrayal of explicit sexual subject matter for the purposes of sexual arousal and erotic satisfaction. Hateful language includes any communication outside the law that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, and religion. All of these are generally immoral and harmful for adolescents’ mental health.

2.1.1 Offensiveness Content Filtering Methods in Social Media

Popular online social networking sites apply several mechanisms to screen offensive contents. For example, Youtube’s safety mode, once activated, can hide all comments containing offensive languages from users. But pre-screened content will still appear—the pejoratives replaced by asterisks, if users simply click "Text Comments." On Facebook, users can add comma-separated keywords to the "Moderation Blacklist." When people include blacklisted keywords in a post and/or a comment on a page, the content will be automatically identified as spam and thus be screened. Twitter client, “Tweetie 1.3,” was rejected by Apple Inc. for allowing foul languages to appear in users’ tweets. Currently, Twitter does not pre-screen users’ posted contents, claiming that if users encounter offensive contents, they can simply block and unfollow those people who post offensive contents.

In general, the majority of popular social media use a simple lexicon-based approach to filter offensive contents. Their lexicons are either predefined (such as Youtube) or composed by the users themselves (such as Facebook). Furthermore, most sites rely on users to report offensive contents to take actions. Because of their use of a simple lexicon-based automatic filtering approach to block the offensive words and sentences, these systems have low accuracy and may generate many false positive alerts. In addition, when these systems depend on users and administrators to detect and report offensive contents, they often fail to take actions in a timely fashion. For adolescents who often lack cognitive awareness of risks, these approaches are hardly effective to prevent young users from being exposed to offensive contents. Therefore, parents need more sophisticated software and techniques to efficiently detect offensive contents to protect their adolescents from potential exposure to vulgar, pornographic and hateful languages.

2.1.2 Using Text Mining Techniques to Detect Online Offensive Contents

Offensive language identification in social media is a difficult task because the textual contents in such environment is often unstructured, informal, and even misspelled. While defensive methods adopted by current social media are not sufficient, researchers have studied intelligent ways to identify offensive contents using a text mining approach. Implementing text mining techniques to analyze online data requires the following phases: 1) data acquisition and preprocess, 2) feature extraction, and 3) classification. The major challenges of using text mining to detect offensive contents lie on the feature extraction phase, which will be elaborated in the following sections.

2.1.2.1 Message-level Feature Extraction

Most offensive content detection research extracts two kinds of features: lexical and syntactic features.

Lexical features treat each word and phrase as an entity. Word patterns such as the appearance of certain keywords and their frequencies are often used to represent the language model. Early research used Bag-of-Words (BoW) in offensiveness detection(McEnery et al., 2000). The BoW approach treats a text as an unordered collection of words and disregards the syntactic and semantic information. However, using a BoW approach alone not only yields low accuracy in subtle offensive language detection, but also brings in a high false positive rate especially during heated arguments, defensive reactions to others' offensive posts, and even conversations between close friends. An N-gram approach is considered as an improved approach in that it brings words' nearby context information into consideration to detect offensive contents (Pendar, 2007). N-grams represent subsequences of N continuous words in texts. Bi-gram and

Tri-gram are the most popular N-grams used in text mining. However, N-gram suffers from difficulty in exploring related words separated by long-distances in texts. Simply increasing N can alleviate the problem but will slow down system processing speed and bring in more false positives.

Syntactic features: Although lexical features perform well in detecting offensive entities, without considering the syntactical structure of the whole sentence, they fail to distinguish offensiveness of sentences that contain the same words but in different orders. Therefore, to consider syntactical features in sentences, natural language parsers (Marneffe et al., 2006) are introduced to parse sentences on grammatical structures before feature selection. Equipping with a parser can help avoid selecting un-related word sets as features in offensiveness detection.

2.1.2.2 User-level Feature Extraction

Most contemporary research on detecting online offensive language only focuses on sentence-level and message-level constructs. Since no detection technique is 100% accurate, if users keep connecting with the sources of offensive contents (e.g., online users or websites), they are at high risk of continuous exposure to offensive contents. However, user-level detection is a more challenging task and studies associated with the user level of analysis are largely missing. There are some limited efforts at the user level. For example, Kontostathis et al (2009) propose a rule-based communication model to track and categorize online predators. Pendar (2007) uses lexical features with machine learning classifiers to differentiate victims from predators in online chatting environments. Pazienza and Tudorache (2011) propose utilizing user profiling features to detect aggressive discussions. They use users' online behavior histories (e.g., presence and conversations) to predict whether or not users' future posts will be offensive. Although their work points out an interesting direction to incorporate user information in detecting offensive contents, more advanced user information such as users' writing styles or posting trends or reputations has not been included to improve the detection rate.

Therefore, we aim to develop an automatic algorithm which efficiently detect both message-level and user-level offensiveness, and also can be used on social media to protect children from cyberbullying.

2.2 Problematic Content

2.2.1 Problematic Content in Mobile Apps

Many researchers have studied the risk level of smartphone platforms from the security and privacy perspective (Lin, Amini, et al., 2012; Möller et al., 2012). The threats and attacks introduced by Android permission systems have also been analyzed (Chia et al., 2012; Felt, Chin, et al., 2011). These studies have found that Android apps normally require more permissions than they actually need in order to support advertisements with the potentially malicious purpose of harvesting users' personal information. Unfortunately, permission warnings make little sense to the general public and cannot help users make correct decision before downloading apps. For example, Kelly et al. (Kelley et al., 2012) found that users have poor understandings of what permission disclosures imply and live under the illusion that app marketplaces censor applications in order to reject malicious and low-quality apps. Similarly, Felt et al. (Felt et al., 2012) found that only 17% of users pay attention to permissions during installation and only 3% of users understand the permission implications comprehensively. Therefore, current permission warnings are both inaccurate and an ineffective means to protect the security and privacy of app users.

Similar to the permissions which are used to measure the security and privacy risks brought by mobile apps, maturity ratings are designed to determine whether there is problematic content in mobile apps for children's protection. In addition, parents highly rely on apps' maturity ratings when choosing apps for their children and adolescents; therefore, an accurate system is essential for assisting them to make correct decisions. Unfortunately, as we already discussed, there is no such system. Apps are assigned incorrect and even conflicting ratings on both Android and iOS platforms (Agten, 2012; Siegler, 2009). However, to date, no research has systematically uncovered the extent and severity of these unreliable app maturity ratings, nor has any research shown the types of apps that are most often mis-categorized. To bridge this gap, we aim to develop mechanisms to verify the maturity ratings of mobile apps and to investigate the possible reasons behind the incorrect ratings.

2.2.2 Problematic Content in Mobile In-app Advertisements

Researchers have examined the security and privacy risks caused by the in-app advertisements on mobile platforms (Chia et al., 2012; Felt, Chin, et al., 2011; Grace et al., 2012b; Haddadi et al., 2011; Lin, Amini, et al., 2012; Möller et al., 2012). As discussed above, this stream of research concludes that, to support the in-app advertisements, apps usually require more permissions than they actually need (Felt, Chin, et al., 2011; Felt, Greenwood, et al., 2011). In addition, to better target customers, the in-app advertisements also aggressively collect users' information from their mobile devices (Egele et al., 2011; Enck et al., 2010b; Grace et al., 2012a; Hornyack et al., 2011). To address privacy and security concerns, researchers have proposed different mechanisms to control and separate advertisements from the host apps (Leontiadis et al., 2012; Pearce et al., 2012; Shekhar et al., 2012; Ter Louw et al., 2010). However, currently little research has considered children as a special group of mobile users or studied the in-app advertisements from the content appropriateness perspective for protecting children's safety. Therefore, we aim to examine the content appropriateness of the in-app advertisements.

2.3 Cyber Solicitation

As discussed above, two major types of behaviors on mobile devices expose children to online sexual solicitation (Ybarra et al., 2007): posting personal information and interacting with online strangers. Therefore, to protect children from cyber solicitation, the most critical and urgent need is to protect children's privacy.

Internet Users' Information Privacy Concerns (IUIPC) (Malhotra et al., 2004) argues that three factors affect users' privacy concerns: collection, control, and awareness. To ease mobile users' privacy concerns, numerous works have been done on Android platform seeking to restrict the collection of personal information (Nauman et al., 2010a), enable privacy controls (Nauman et al., 2010b), and generate useful risk indicators to improve user's privacy awareness (Frank et al., 2012; Lin, Sadeh, et al., 2012; Pandita et al., 2013; Peng et al., 2012b). However, restricting the collection of personal information or enabling controls are not optimal solutions to protect privacy on mobile platforms. Nauman et al (Nauman et al., 2010a) propose to alter Android all-or-nothing permission system to a fine-grained system where users can allow certain permission requests while deny the others. However, restricting information collection may stop service providing or break mobile eco-system, because some apps do need users' personal information for service (e.g., Google Maps need users' location information to provide navigation service) and mobile advertising networks do need users' personal information for in-app advertisement targeting. Apex (Nauman et al., 2010b) allows Android users to generate fine-grained rules to control and constraint the accessibility of PI on their mobile devices. Unfortunately, the access control policies in users' minds are not static, and many times, they are even unclear. It is found that although users' ideal access control policies are complicated, they make poor decisions when actually configuring the policies (Sadeh et al., 2009); people are not always defined exclusively in standard role-based terms, but in-situ decisions (Mazurek et al., 2010); when familiar with a product, people may relax the rules (Lin, Sadeh, et al., 2012).

Our study lies in the research stream which aims to improve user's privacy awareness on the Android platform. Our goal is to acknowledge parents about how apps use their children's personal information, so that they can make informed decision when choosing apps. Previous researches in this stream derive apps' security and privacy risk levels from the app permissions, app codes, user studies, and app descriptions.

From app permissions, researchers discover that apps tend to over-request permissions to support advertisements with the potentially malicious purpose of harvesting users' personal information (Felt, Chin, et al., 2011); Peng et al (Peng et al., 2012b) scores the risks of Android apps by analyzing the rareness of requested permissions; Frank et al (Frank et al., 2012) mines the permission request patterns from Android and Facebook apps. From app codes, dynamic analysis tools (Enck et al., 2010a) are developed to detect sensitive information flows in apps. Unfortunately, those tools can only perform manually. Existing static analysis tools can detect the resource (e.g., personal information) carried in information flows, but fail to differentiate whether the senders and receivers involved in the information flows are servers or human users. In addition, although code-level analysis can easily tell whether certain types of information are collected by apps, it cannot infer the collection purposes.

Through user surveys, researchers study whether the collections of personal information in apps contradict to users' privacy expectations (Lin, Sadeh, et al., 2012). The results show that users feel more comfortable when informed their location information is collected for major functionality than informed location is collected for sharing/tagging. However, FTC (FTC, 2012) found that the privacy notification on mobile devices are disappointing. Therefore, in this study, we aim to provide clear privacy notifications to parents to assist decision making process.

Some researchers study risks on Android platform using data-mining techniques to process app descriptions. It associates mobile apps with human beings (e.g., end-users, app developers), for developers normally reveal app functionalities and development intents in their app descriptions. Data-mining methods can be used to efficiently detect user roles, app contents, functions, and purposes. Our study majorly lies in this research stream. We propose a systematic method to quantify privacy risks on mobile platforms. We not only want to tell parents *what* personal information is collected, we also want to give them clues about *why* and *how* the personal information is collected. Therefore, we use both app descriptions and app codes to derive privacy risks on mobile platforms.

Chapter 3 Online Harassment and Cyber bullying

3.1 Introduction

With the rapid growth of social media, users especially adolescents are spending a significant amount of time on various social networking sites to connect with others, to share information, and to pursue common interests. In 2011, 70% of teens used social media sites on a daily basis (Johnson et al., 2011) and nearly one in four teens hit their favorite social-media sites 10 or more times a day (Gwenn et al., 2011). While adolescents benefit from their use of social media by interacting with and learning from others, they are also at risk of being exposed to large amounts of offensive online contents. ScanSafe's monthly "Global Threat Report" (Cheng, 2007) found that up to 80% of blogs contained offensive contents and 74% included porn in the format of image, video, or offensive languages. In addition, cyber-bullying occurs via offensive messages posted on social media. It has been found that 19% of teens report that someone has written or posted mean or embarrassing things about them on social networking sites (Johnson et al., 2011). To prevent adolescents from being negatively affected by biased and harmful contents, detecting online offensive contents becomes an urgent task.

To address concerns on children’s access to offensive content over the Internet, administrators of social media often manually review online contents to detect and delete offensive materials. However, the manual review tasks of identifying offensive contents are labor intensive, time consuming, and thus not sustainable and scalable in reality. Some automatic content filtering software packages, such as Appen and Internet Security Suite, have been developed to detect and filter online offensive contents. Most of them simply blocked webpages and paragraphs that contained ‘dirty’ words. These word-based approaches not only affect the readability and usability of web sites, but also fail to identify subtle offensive messages. For example, under these conventional approaches, the sentence “you are such a crying baby” will not be identified as offensive content, because none of its words is included in general offensive lexicons. In addition, the false positive rate of these word-based detection approaches is often high, due to the word ambiguity problem, i.e., the same word can have very different meanings in different contexts. Moreover, existing methods treat each message as an independent instance without tracing the source of offensive contents.

To address these limitations, we propose a more powerful solution to improve the deficiency of existing offensive content detection approaches. Specifically, we propose the Lexical Syntactic Feature-based (LSF) language model to effectively detect offensive language in social media to protect adolescents. LSF provides high accuracy in subtle offensive message detection, and it can reduce the false positive rate. Besides, LSF not only examines messages, but also the person who posts the messages and his/her patterns of posting. LSF can be implemented as a client-side application for individuals and groups who are concerned about adolescent online safety. It is able to detect whether online users and websites push recognizable offensive contents to adolescents, trigger applications to alert the senders to regulate their behavior, and eventually block the sender if this pattern continues. Users are also allowed to adjust the threshold of acceptable level of offensive contents. Our language model may not be able to make adolescents completely immune to offensive contents, because it is hard to fully detect what is “offensive.” However, we aim to provide an improved automatic tool to detect offensive contents in social media to help school teachers and parents have better control over the contents adolescents are viewing.

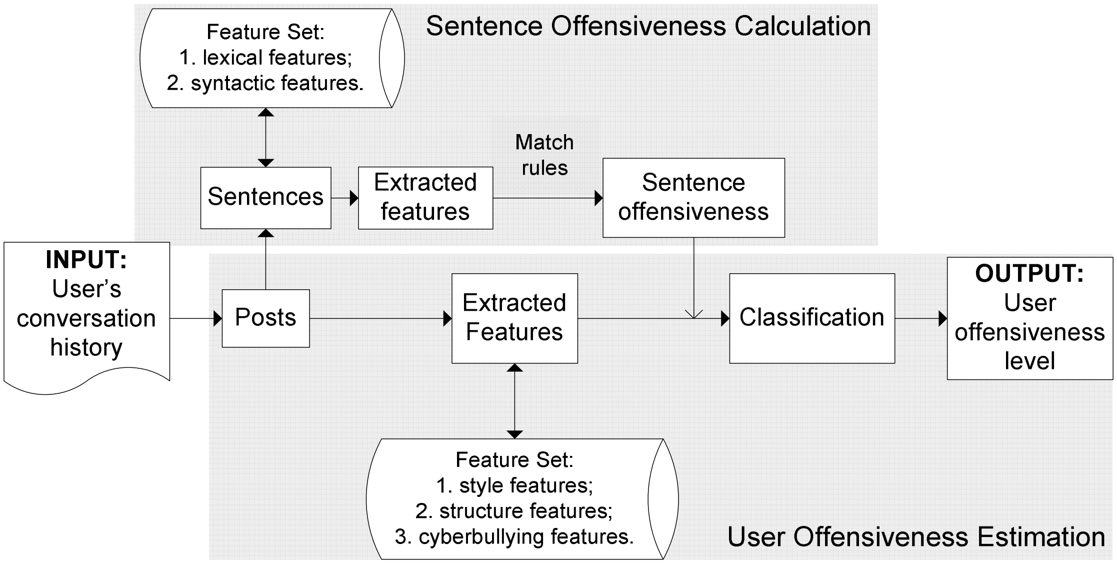


Figure 1. Framework of LSF-based offensive language detection

3.2 Research Questions

Based on our review, we identify the following research questions to prevent adolescents from being exposed to offensive textual content:

- What strategy is effective in detecting and evaluating the level of offensiveness in a message? Will advanced linguistic analysis improve the accuracy and reduce false positives in detecting message-level offensiveness?
- What strategy is effective in detecting and predicting user-level offensiveness? Besides using information from message-level offensiveness, could user profile information further improve the performance?
- Is the proposed framework efficient enough to be deployed on real time social media?

3.3 Design Framework

In order to tackle these challenges, we propose a Lexical Syntactic Feature (LSF) based framework to detect offensive content and identify offensive users in social media. We propose to include two phases of offensiveness detection. Phase 1 aims to detect the offensiveness on the sentence level and Phase 2 derives offensiveness on the user level. In Phase 1, we apply advanced text mining and natural language processing techniques to derive lexical and syntactic features of each sentence. Using these features, we derive an offensiveness value for each sentence. In Phase 2, we further incorporate user-level features where we leverage research on authorship analysis. The framework is illustrated in Fig.1.

The system consists of pre-processing and two major components: sentence offensiveness prediction and user offensiveness estimation. During the pre-processing stage, users' conversation history is chunked into posts, and then into sentences. During sentence offensiveness prediction, each sentence's offensiveness can be derived from two features: its words' offensiveness and the context. We use lexical features to represent words' offensiveness in a sentence, and syntactic features to represent context in a sentence. Words' offensiveness nature is measured from two lexicons. For the context, we grammatically parse sentences into dependency sets to capture all dependency types between a word and other words in the same sentence, and mark some of its related words as intensifiers. The intensifiers are effective in detecting whether offensive words are used to describe users or other offensive words. During user offensiveness estimation stage, language patterns are used to predict the likelihood of individuals being offensive.

3.3.1 Sentence Offensiveness Calculation

To address the limitations of the previous methods for sentence offensiveness detection (Mahmud, 2008; Razavi et al., 2010; Spertus, 1997; Yin et al., 2009), we propose a new method of sentence-level analysis based on offensive word lexicons and sentence syntactic structures. Firstly, we construct two offensive word dictionaries based on different strengths of offensiveness. Secondly, the concept of syntactic intensifier is introduced to adjust words' offensiveness levels based on their context. Lastly, for each sentence, an offensiveness value is generated by aggregating its words' offensiveness. Since we already use intensifiers to further adjust words' offensiveness, no extra weights are assigned to words during the aggregation.

a) Lexical Features: Offensiveness Dictionary Construction

Offensive sentences always contain pejoratives, profanities, or obscenities. Strong profanities, such as "f***" and "s***", are always undoubtedly offensive when directed at users or objects; but there are many other weakly pejoratives and obscenities, such as "stupid" and "liar," that may also be offensive. This research differentiates between these two levels of offensiveness based on their strength. The offensive word lexicon used in this research includes the lexicon used in Xu and Zhu's study (Xu & Zhu, 2010) and a lexicon, based on Urban Dictionary, established during the coding process. All profanities are labeled as strongly offensive. Pejoratives and obscenities receive the label of strongly offensive if more than 80% of their use in our dataset is offensive. The dataset is collected from Youtube command board (details will be described in the experiment section). Otherwise, known pejoratives and obscenities receive the label of weakly offensive word. Word offensiveness is defined as: for each offensive word, w, in sentence, s, its offensiveness

$$O_w = \begin{cases} a_1 & \text{if w is a strongly offensive word} \\ a_2 & \text{if w is a weakly offensive word} \\ 0 & \text{othewise} \end{cases} \tag{1}$$

where $1 > a_1 > a_2$, for the offensiveness of strongly offensive words is higher than weakly offensive words.

b) Syntactic Features: Syntactic Intensifier Detection

Once pejoratives or obscenities are directed at online users, or semantically associated with another pejorative or obscenity, they become more offensive from users' perspectives. For example, "you stupid" and "f***ing stupid," are much more insulting than "This game is stupid." In addition, the dataset from Content Analysis for the Web2.0 Workshop[4] shows that most offensive sentences include not only offensive words but also user identifiers, i.e. second person pronouns, victim's screen names, and other terms referring to people. Table 1 lists some examples of this type of sentences.

When offensive words grammatically relate to user identifiers or other offensive words in sentences, the offensiveness level requires adjusting. This study uses a nature language process parser, proposed by Stanford Natural Language Processing Group, to capture the grammatical dependencies within a sentence. The parsing results of sentences become combinations of a dependency-type and word-pair with the form "(governor, dependent)." For example, the typed dependency "appos (you, idiot)" in the sentence "You, by any means, an idiot." means that "idiot", the dependent, is an appositional modifier of the pronoun "you," the governor. The governor and dependent can be any syntactic elements of sentences. Some selected dependency types capture the possible grammatical relations between an offensive word and a user-identifier (or another offensive word) in a sentence. The study also proposes syntactical intensifier detection rules listed in Table 2 (A represents a user identifier, and B represents an offensive word).

Table 1. Language Features of Offensive Sentences

Language Features	**Example**

---

[4] http://caw2.barcelonamedia.org/

Second person pronoun (victim's screen name) + pejorative (i.e. JK, gay, wtf, emo, fag, loner, loser)	<You, gay>
Offensive adjective (i.e. stupid, foolish, sissy) + people referring terms (i.e. emo, bitch, whore, boy, girl)	<stupid, bitch> <sissy, boy>

The offensiveness levels of offensive words and other inappropriate words receive adjustment by multiplying their prior offensiveness levels by an intensifier (Zhang et al., 2009). In sentence, s, words syntactically related to the offensive word, w, are categorized in an intensifier set, $i_{w,s} = \{c_1, ..., c_k\}$, for each word $c_j (1 \le j \le k)$, its intensify value, d_j, is defined as:

$$d_j = \begin{cases} b_1 & \text{if } \mathrm{c_j} \text{ is a user identifier} \\ b_2 & \text{if } \mathrm{c_j} \text{ is an offensive word} \\ 1 & \text{otherwise} \end{cases} \qquad (2)$$

where $b_1 > b_2 > 1$, for offensive words used to describe users are more offensive than the words used to describe other offensive words. Thus, the value of intensifier, I_w, for offensive word, w, can be calculated as $\sum_{j=1}^{k} d_j$.

c) Sentence Level Offensiveness Value Generation

Consequently, the offensiveness value of sentence, s, becomes a determined linear combination of words' offensiveness, $O_s = \sum o_w I_w$.

3.3.2 User Offensiveness Estimation

In user offensiveness estimation stage, our design has two major steps: aggregating users' sentence offensiveness and extracting extra features from users' language styles. We incorporate sentence offensiveness values and user language features to classify users' offensiveness.

a) Sentence Offensiveness Aggregation

While there are few studies on user-level offensiveness analysis, studies on document-level sentiment analysis share some similarity with this research (Pang et al., 2002; Tsou et al., 2005; Turney, 2002; Zhang et al., 2009). Document-level sentiment analysis predicts the overall polarity of a document by aggregating polarity scores of individual sentences. Since the importance of each sentence varies in a document, one assigns weights to all sentences to adjust their contributions to the overall polarity. Similarly, we cannot simply sum up the offensive values of all sentences to compute users' offensiveness, because the strength of sentence offensiveness depends on its context. For example, one may post "Stupid guys need more care. You are one of them." If we calculate offensiveness level of this sentence without considering the context, the offensiveness of this post will not be detected even using natural language parsers. To bypass the limitation of current parsers, we modify each post by combining sentences and replacing the periods with commas before feeding them to parsers. Then the parser generates different phrase sets for further calculation of the offensiveness level of the modified posts. However, since the modified posts may sometimes miss the original meanings, we have to balance between using the sum of sentence offensiveness and using the offensiveness of the modified posts to represent post offensiveness. In this case, the greater value of the two is chosen to represent the final posts' offensiveness levels. The detail of the schema is illustrated as follows:

Given a user, u, we retrieve his/her conversation history which contains several posts $\{p_1, ..., p_m\}$, and each post $p_i (1 \le i \le m)$ contains sentences $\{s_1, ..., s_n\}$. Sentence offensiveness values are denoted as $\{O_{s_1}, ..., O_{s_n}\}$. The original offensiveness value of post p, $O_p = \sum O_s$. The offensiveness value of modified posts can be presented as, $O_{p \to s}$. So the final post offensiveness $O_p^{'}$ of post p can be calculated as, $O_p^{'} = \max(O_p, O_{p \to s}) = \max(\sum O_s, O_{p \to s})$.

Hence, the offensiveness value, O_u, of user, u, can be presented as, $O_u = \frac{1}{m}\Sigma O'_p$. We normalize the offensiveness value because users who have more posts are not necessarily more offensive than others. O_u, should be no less than 0.

Table 2. Syntactical Intensifier Detection Rules

Rules	Meanings	Examples	Dependency Types

Descriptive Modifiers and complements:
A(noun, verb, adj) ←B(adj, adv, noun)
	B is used to define or modify A.	you f***ing;
		you who f***ing;
		you…the one…f***ing.
			• abbrev (abbreviation modifier),
			• acomp (adjectival complement),
			• amod (adjectival modifier),
			• appos (appositional modifier),
			• nn (noun compound modifier),
			• partmod (participial modifier)

Object:
B(noun, verb) ←A(noun)
	A is B's direct or indirect object.	F*** yourselves;
		shut the f** up;
		f*** you idiot;
		you are an idiot;
		you say that f***...
			• dobj (direct object),
			• iobj (indirect object),
			• nsubj (nominal subject)

Subject: A(noun)→B(noun, verb)	A is B's subject or passive subject.	you f***…; you are **ed… …f***ed by you…	• nsubj (nominal subject), • nsubjpass (passive nominal subject), • xsubj (controlling subject), • agent (passive verb's subject).
Close phrase, coordinating conjunction: A and B; …A, B…; …B, B…	A and B or two Bs are close to each other in a sentence, but be separated by comma or semicolon.	F** and stupid; you, idiot.	• conj (conjunct), • parataxis (from Greek for "place side by side")
Possession modifiers: A(noun)→B(noun)	A is a possessive determiner of B.	your f*** …; s*** falls out of your mouth.	• poss (holds between the user and its possessive determiner)

Rhetorical questions: A(noun)🡐B(noun)	B is used to describe clause with A as root (main object).	Do you have a point, f***?	- rcmod (relative clause modifier)

b) Additional Features Extracted from Users' Lanuage Profiles

Other characteristics such as the punctuation used, sentence structure, and the organization of sentences within posts could also affect others' perceptions of the poster's offensiveness level. Considering the following cases:

- *Sentence styles.* Users may use punctuation and words with all uppercase letters to indicate feelings or speaking volume. Punctuation, such as exclamation marks, can emphasize offensiveness of posts. (i.e. Both "You are stupid!" and "You are STUPID." are stronger than "You are stupid."). Some users tend to post short insulting comments, such as "Holy s***." and "You idiot." Consequently, compared to those who post the same number of offensive words but in longer sentences, the former users appear more offensive for intensive usage of pejoratives and obscenities. Users may use offensive words to defend themselves when they are arguing with others who are offensive. But it is costly to detect whether their conversation partners are offensive or not. Instead, we noticed that arguments should happen in a relatively short period of time. For example, for user u, whose conversation history is valid in 100 days within 2 years, while the time period he/she is using offensive words is only 5 days, no matter how many offensive words (s)he is using, (s)he should not be considered as an offensive user. Thus, making sure that users' offensiveness values are evenly distributed over the span of their conversation history is a reasonable way to differentiate generally offensive users from the occasional ones.

Table 3. Additional Feature Selection for User Offensiveness Analysis

Style Features	**Structural Features**	**Content-specific Features**

-Ratio of short sentences -Appearance of punctuations -Appearance of words with all uppercase letters	-Ratio of imperative sentences -Appearance of offensive words as nouns, verbs, adjs and advs.	-Race -Religion -Violence -Sexual orientation -Clothes -Accent -Appearance -Intelligence -Special needs or disabilities

- *Sentence structures.* Users who frequently use imperative sentences tend to be more insulting, because imperative sentences deliver stronger sentiments. For example, a user who always posts messages such as “F***ing u” and “Slap your face” gives the impression of being more offensive and aggressive than those ones posting “you are f***ing” and “your face get slapped.”
- *Cyberbullying related content.* O'Neill and Zinga (2008) described seven types of children who, due to differences from peers, may be easy targets for online bullies, including those children from minority races, with religious beliefs, or with non-typical sexual orientations. Detecting online conversations referring to these individual differences also provides clues for identifying offensive users.

Based on the above observations, three types of features are developed to identify the level of offensiveness, which leveraged from authorship analysis research on cybercrime investigation (Hansen et al., 2007; Ma et al., 2011; Orebaugh & Allnutt, 2010; Symonenko et al., 2004; Zheng et al., 2006; Zheng et al., 2010): style features, structural features, and content-specific features. Style features and structural features capture users’ language patterns, while content-specific features help to identify abnormal contents in users’ conversations. The style features in our study infer users’ offensiveness levels from their language patterns, including whether or not they are frequently/recently using offensive words and intensifiers such as uppercase letters and punctuation. The structural features capture the way users construct their posts, which check whether or not users are frequently using imperative sentences. They also try to infer users' writing styles by checking offensive words used as nouns, verbs, adjectives (later referred as adjs), or adverbs (later referred as advs). The content-specific features check whether or not users post suspicious contents which probably will be identified as cyberbullying messages. In this study, we identify cyberbullying contents by checking whether they contain cyberbullying related words (i.e. religious words). The details of these features are summarized in Table 3.

c) Overall User Offensiveness Estimation

Besides style features, structure features and content-specific features, sentence offensiveness values are considered as one type of user language features. By using these features, machine learning techniques can be adopted to classify users' offensiveness levels.

3.4 Experiment

This section describes several experiments we conducted to examine LSF on detecting offensiveness languages in social media.

3.4.1 Dataset Description

The experimental dataset, retrieved from Youtube comment boards, is a selection of text comments from postings in reaction to the top 18 videos. Classification of the videos includes thirteen categories: Music, Autos, Comedies, Educations, Entertainments, Films, Gaming, Style, News, Nonprofits, Animals, Sciences, and Sports. Each text comment includes a user id, a timestamp and text content. The user id identifies the author who posted the comment, the timestamp records when the comment was posted and the text content contained a user's comments. The dataset includes comments from 2,175,474 distinct users.

3.4.2 Pre-processing

Before feeding the dataset to the classifier, an automatic pre-processing procedure assembles the comments for each user and chunks them into sentences. For each sentence in the sample dataset, an automatic spelling and grammar correction process precedes introduction of the sample dataset to the classifier. With the help of the WordNet corpus and spell-correction algorithm[5], correction of spelling and grammar mistakes in the raw sentences occurs by tasks such as deleting repeated letters in words, deleting meaningless symbols, splitting long words, transposing substituted letters, and replacing the incorrect and missing letters in words. As a result, words missing letters, such as "speling," are corrected to "spelling"; misspelled words, such as "korrect," change to "correct." The error of omission is 11% and the error of commission is 20%.

3.4.3 Experiment Settings in Sentence Offensive Prediction

The experiment compares six approaches in sentence offensive prediction:

d) Bag-of-words (BoW): The BoW approach disregards grammar and word order and detects offensive sentences by checking whether or not they contain both user identifiers and offensive words. This approach also acts as a benchmark.

---

[5] Spell-Correction Algorithm, at http://norvig.com/spell-correct.html

e) 2-gram: The N-gram approach detects offensive sentences by selecting all sequences of n words in a given sentence and checking whether or not the sequences include both user identifiers and offensive words. In this approach, N equals to 2, it also acts as a benchmark.

f) 3-gram: N-gram approach, selecting all sequences of 3 words in a given sentence. It also acts as a benchmark.

g) 5-gram: N-gram approach, selecting all sequences of 5 words in a given sentence. It also acts as a benchmark.

h) Appraisal approach: The appraisal approach was proposed for sentiment analysis (Whitelaw et al., 2005), here we use it on sentence offensive detection for comparison. It can detect offensive sentences by going through all types of dependency sets and checking whether or not certain offensive words and user identifiers are grammatically related in a given sentence. The major differences between applying the appraisal approach on sentence offensive detection and ours is that the appraisal approach cannot differentiate offensive words based on their strength, and it generally considers two words as “related” if they are within any type dependency set, while some of the dependency types do not really indicate one is acting on the other. For instance, type dependency “parataxis” relation (from Greek for “place side by side”) is a relation between the main verb of a clause and other sentential elements, such as a sentential parenthetical, a clause after a “:” or a “;”. An example sentence for type dependency “parataxis(left, said)” can be “The guy, John said, left early in the morning”. Here “said” and “left” are not really used to describe one another.

i) LSF: The sentence offensive prediction method proposed in this study.

3.4.4 Evaluation Metrics

In our experiments, standard evaluation metrics for classification in sentiment analysis (Pang et al., 2002; Turney, 2002; Ye et al., 2006) (i.e., precision, recall, and f-score) are used to evaluate the performance of LSF. In particular, precision presents the percent of identified posts that are truly offensive messages. Recall measures the overall classification correctness, which represents the percent of actual offensive messages posts that are correctly identified. False positive (FP) rate represents the percent of identified posts that are not truly offensive messages. False negative (FN) rate represents the percent of actual offensive messages posts that are unidentified. F-score (Yin et al., 2009) represents the weighted harmonic mean of precision and recall, which is defined as:

$$f-score = \frac{2(precision \times recall)}{precision + recall} \qquad (3)$$

3.4.5 Experiment 1: Sentence Offensiveness Calculation

In this experiment, we randomly select a uniform distributed sample from the dataset, which includes 1700 sentences. The number of sentences is limited because we manually label each of them, and the labeling process is time consuming. In total, 359 strongly offensive words and 251 weakly offensive words are selected as offensive word lexicons, and the experimental parameters are set as: $a_1 = 1; a_2 = 0.5; b_1 = 2; b_2 = 1.5.$ We define "1" to be the threshold for offensive sentence classification, that is, sentences with offensiveness values more than (inclusive) "1" receive labels of offensive sentences, because by our definition, offensive sentence means a sentence containing strongly offensive words, or containing weakly offensive words used to describe another user. We develop coding standards on the sentences, and train two other coders to label the sentences as "offensive" or "inoffensive". Each person firstly labels the sentences individually. For sentences with disagreements, labels are figured out by discussion. The inter-coder reliability is measured as 0.73 in Cohen's Kappa. After manual labeling, 173 sentences are labeled as "offensive", and 1527 sentences are labeled as "inoffensive". With such ground truth, the classifier's detection accuracy is presented in Fig. 2.

According to Fig.2, none of the baseline approaches provides recall rate higher than 70%, because many of the offensive sentences are imperatives, which omit all user identifiers. Among the baseline approaches, the BoW approach has the highest recall rate 66%. However, BoW generates a high false positive rate because it captures multiple unrelated <user identifier, offensive word> sets.

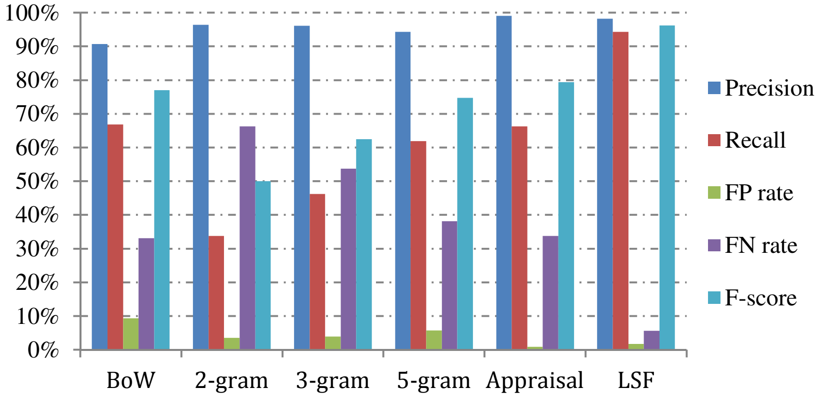


Figure 2. Accuracies of sentence level offensiveness detection

The recall of N-gram is low when n is small. However, as n increases, the false positive rate increases as well. Once N equals to the length of sentences, N-gram is equivalent to the bag-of-words approach. To further apply N-gram in the classification, application of different values of N is necessary to balance, perfectly, the trade-off between recall and false positive rate.

The appraisal approach reaches high precision, but its recall rate is poor. LSF obtains the highest f-score, because it sufficiently balances the precision-recall tradeoff. It achieves precision of 98.24% and recall of 94.34% in sentence offensive detection. Unfortunately, the parser sometimes misidentifies noun appositions, in part because of typographical errors in the input, such as: “you stupid sympathies” Here, the sender presumably meant to write “your” instead of “you.” This is the major reason for false negative rates. The false positive rate arises mainly from multiple appearances of weak offensive words, for example, “fake and stupid,” which can only represent a negative opinion for a video clip but accidently identified as “offensive” because LSF calculate a value higher than (or equal to) 1.

3.4.6 Experiment 2: User Offensiveness Estimation-with presence of strongly offensive words

In this experiment we randomly select a uniform distributed sample from the dataset, which includes 249 users. Each user has 15 posts on average, in total 3735 sentences. Each of the 249 users was coded by the same three coders. Coders were trained to label a user as being offensive if his(her) posts contained insulting or abusive language that makes the recipient feel offended, not merely if the sender expressed disagreement with the recipient. Coders label users as “offensive” or “inoffensive”. Each person firstly labels the users individually. For users with disagreements, labels are figured out by discussion. As a result, 99 users are labeled as “offensive”, and 150 users are labeled as “inoffensive”. After balancing the positive and negative results, we have 99 users in each class.

Machine learning techniques—NaiveBayes (NB) and SVM—are used to perform the classification, and 10-fold cross validation was conducted in this experiment. To fully evaluate the effectiveness of users’ sentence offensiveness value (LSF), style features, structure features and content-specific features for user offensiveness estimation, we fed them sequentially into the classifiers, and get the result in Fig.3. The “BoW (Strong+Weak)” means uses bag-of-word (including strongly offensive words and weakly offensive words) as the base feature to detect offensive user. Similarly, "LSF" means the sentence offensiveness value generated by LSF is used as the base feature. For the right two bars in Fig 3, no base feature is used. They are presented for comparison.

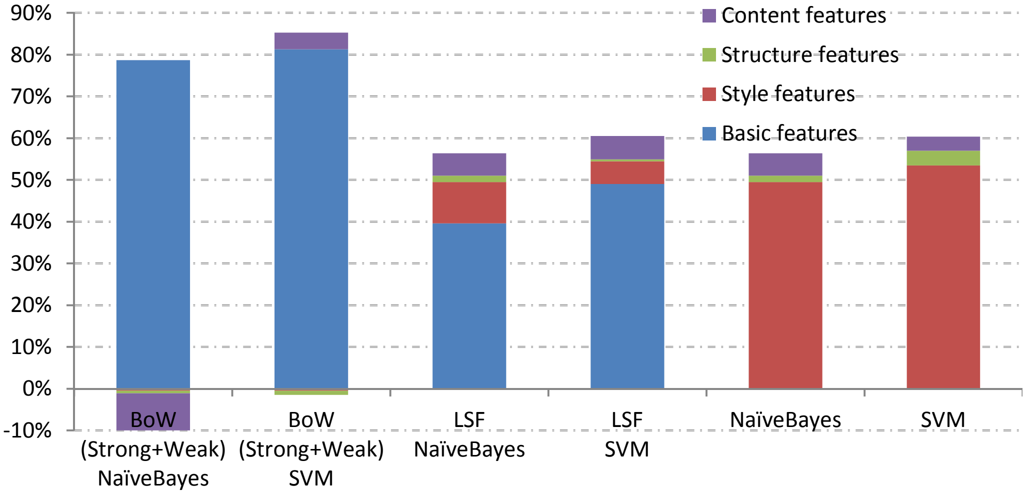


Figure 3. F-score for different feature sets using NB and SVM

According to Fig.3, offensive words and user language features are not compensating to each other to improve the detection rate, which means they are not independent. In contrast, incorporating with user language features, the classifiers have better detection rate than just adopting LSF. While all three types of features are useful to improve the classification rate, style features and content features are more valuable than structure features in user offensiveness classification. However, LSF is not as useful as using offensive words alone in detecting offensive users. One possible reason is that once the number of strongly offensive words is beyond some threshold amount, the user who posts the comments is considered being offensive anyway. In such case, LSF might be less useful than using merely offensive words. We looked further into this situation and tested the model under a situation where the messages do not contain strong offensive words and are not obviously offensive in Experiment 3.

3.4.7 Experiment 3: User Offensiveness Estimation-without strongly offensive words

In this experiment we only want to test the situation when the offensiveness of a user is subtle. We chose to use a dataset without strongly offensive words. Our testing data are randomized selections of the original data followed by filtering out messages that contain strong offensive words. We got 200 users with uniformly distributed offensiveness values. This dataset does not overlap with the one in experiment 2. The selected users have 85 posts on average, and none of the posts contains strongly offensive words. After balancing the positive and negative results, we have 81 users in each class. The experiment condition is identical to Experiment 2. The result in presented in Fig.4. “Weak” means it is simply using (weak) offensive words as the base feature to detect offensive user, because there are no strongly offensive words in this experiment.

According to Fig.4, style features and content features are still more valuable than structure features in user offensiveness classification. However, we did observe the appearance of imperative sentences frequently occurs in offensive users’ conversations. One possible cause for this is that the POS tagger does not have enough accuracy in tagging verbs, and it even marks “Yes”, ”Youre” and “Im” as verbs in some sentences. In such cases, many imperative sentences are not tagged, and the tagged ones are not necessary imperative.

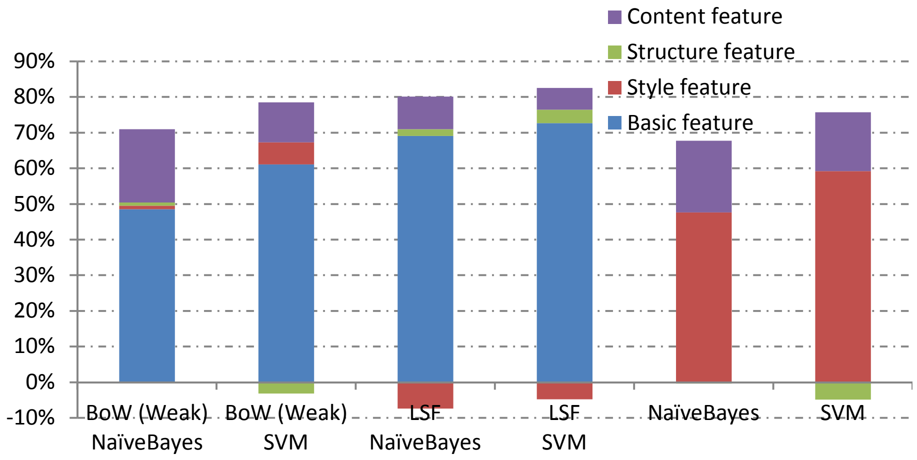


Figure 4. F-score for different feature sets using NB and SVM (without strongly offensive words)

In this experiment, LSF performs better than offensive words in detecting offensive users, it achieves precision of 77.9% and recall of 77.8% in user offensive detection using SVM, which shows LSF do work better than BoW in non-obvious user offensiveness detection, and incorporating user language features will further stir the detection rate. Therefore, we can further conclude that considering context and talking objects will help detect precisely offensive language that does not have ‘dirty’ words. However, strongly offensive words are still the primary element which annoys general readers. Our experiment results suggest a possible 2-stage offensiveness detection in general online environment. For the media where strongly offensive words appear frequently, BoW is sufficient on offensiveness detection. But for the media where strongly offensive words rarely appear, LSF works better on offensiveness detection.

3.4.8 Experiment 4: Efficiency

Experiment 4(a): Efficiency of Sentence Offensiveness Calculation

In addition to accuracy measurement, assessment of processing speed on masses of text messages is necessary, because speed is a critical attribute for offensive detection in real-time online communities. The sentence processing time in each case appear in Fig.5.

The average time for reading each word is 0.0002 ms, and it takes 0.0033 ms to compare it with the words in dictionaries to determine whether it is a user identifier or an offensive word. In our sample, each sentence contains about 10.42 words. Thus, the average processing time for BoW and N gram can be calculated as read time plus twice comparison time for each word in the sentence, which is about 0.07 ms (shown in Fig.5). However, for the appraisal approach, it takes a longer time to grammatically parse sentences before the analysis. In contrast, the LSF method firstly checks whether the sentence contain offensive words. If it does contain offensive words, LSF will proceed to parse the sentence and search for their intensifiers. We list the worst case for the LSF method in Fig.5, and its performance really depends on the offensive sentence ratio on social media. However, we still can prove it is practical for application to online social media and other real-time online communities. Take Youtube as an example, over 80% of its content doesn't contain offensive words, so the sentence processing rate for LSF can be cut down to 2.6 ms.

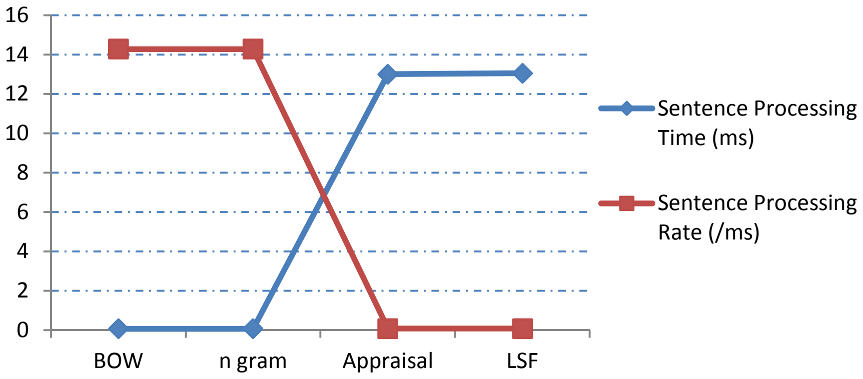


Figure 5. Sentence Processing Time for different methods

Experiment 4(b): Efficiency of User Offensiveness Estimation

In experiment 2, users have 15 posts on average, and each post contains 2 sentences, totaling 31 sentences posted by each user. In experiment 3, users have 85 posts on average, and each post contains 4 sentences, totaling 339 sentences posted by each user. The feature extraction times for different feature sets in experiment 2 and experiment 3 are presented in Fig.6.

From Fig.6, we find that aggregating users' sentences offensiveness (LSF) takes most of the time, and it is positively correlated with the number of sentences a user posts. Other than that, the calculation of structure features also takes much more time than style features and content-specific features. Assume an online user has 100 sentences in his (her) conversation history; it takes approximately 1.9s to extract both the sentence feature and language features, which will not even be noticed.

We further examined the classification rates for different feature sets using NaiveBayes and SVM classifiers. Since the rates vary from time to time, we run each instance 5 times and take the average. The result is shown in Fig.7. As shown, we find that the calculation rate of machine learning techniques is much faster than feature extraction time in Fig.6, the longest running time for machine learning classifiers is only 0.33s to predict users' offensiveness. And the classification rate is independent on the number of users and the number of sentences. Generally, NaiveBayes works much faster than SVM in classification, but SVM produces more accurate classification results.

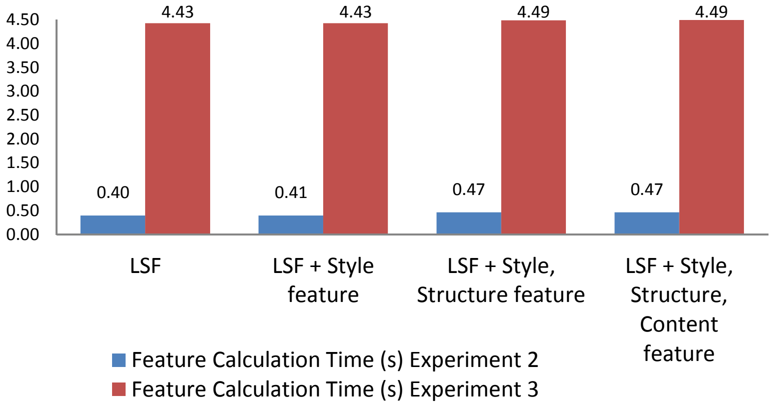


Figure 6. Feature extraction time for different feature sets in Experiment 2 and Experiment 3 (per user)

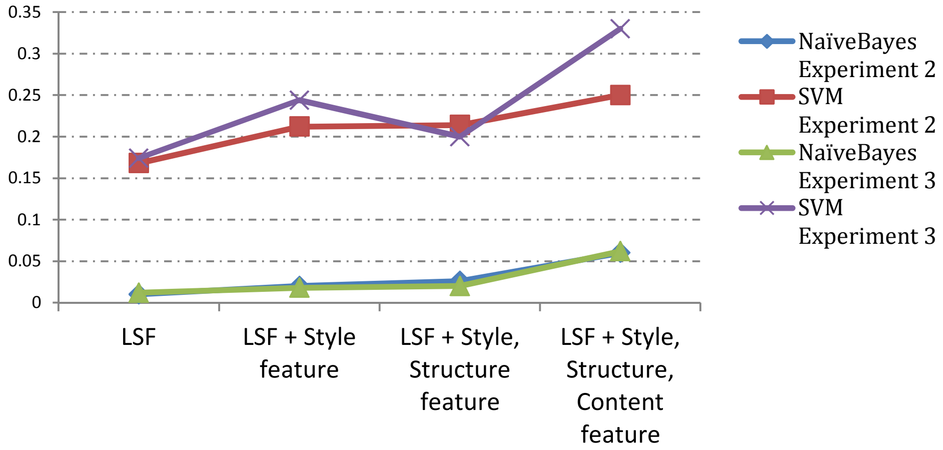


Figure 7. Classification Time for different feature sets using NaiveBayes and SVM classifiers in Experiment 2 and Experiment 3

To sum up, for a user who posts 100 sentences on social media, LSF takes approximately 2.2 second to predict users' offensive potential. While this is substantially longer than the fastest method, it is still a very short time and one that will be acceptable for users to wait.

3.5 Conclusion

In this study, we investigate existing text-mining methods in detecting offensive contents for protecting adolescent online safety. Specifically, we propose the Lexical Syntactical Feature (LSF) approach to identify offensive contents in social media, and further predict a user's potentiality to send out offensive contents. Our research has several contributions. First, we practically conceptualize the notion of online offensive contents, and further distinguish the contribution of pejoratives/ profanities and obscenities in determining offensive contents, and introduce syntactic features in identifying name-calling harassment. Second, we improved the traditional machine learning methods by not only using lexical features to detect offensive language, but also incorporating style features, structure features and context-specific features to better predict a user's potentiality to send out offensive content in social media. Experimental results show that the LSF sentence offensiveness prediction and user offensiveness estimate algorithms outperform traditional learning-based approaches in terms of precision, recall and f-score. It also achieves high processing speed for effective deployment in social media. Besides, the LSF tolerates informal and misspelling contents, and it can easily adapt to any formats of English writing styles. We believe that such a language processing model will greatly help online offensive language monitoring, and eventually build a safer online environment.

Chapter 4 Exposure to Problematic Content—Mobile Apps

4.1 Introduction

With the rapid adoption of smartphones, tablets, and mobile apps, more and more people use these personal digital devices for communication, entertainment, and professional activities. According to a 2012 survey, approximately half of U.S. mobile consumers own either a smartphone or a tablet (Mitchell et al., 2012), and this number was predicted to increase to 70 percent by 2013 (Hardawar, 2012). The sweeping popularity of smartphones and tablets also affects the user population of children and adolescents. It has been shown that 25% of toddlers used their parents' smartphones in 2011 (Carmichael, 2011), and 23% of children and teens between the ages of 12 and 17 owned their own smartphones in 2012 (EnterpriseAppsTech, 2012).

Among smartphone and tablet operating systems, Android and Apple's iOS dominate the U.S. smartphone market by 52.5 and 34.3 percent, respectively (Graziano, 2012). Meanwhile, the growing pace of mobile app offerings is exponential. Approximately 25,000 new apps are added to the Google Play Store per month, amounting to a total of 567,322 apps as of 2012 (Statista, 2012). There are 18,389 new apps added to the iOS App Store every month, totaling 723,750 apps as of 2012 (148Apps.biz, 2012).

In order to help parents determine age-appropriate mobile apps for their children, both Android and iOS apps come with maturity ratings that are similar to the movie and video game industry. Such maturity ratings examine the existence and intensity of mature themes such as mature content, violence, offensive language, sexual content, and drug usage within each app. However, movie and video game industries have official rating organizations such as the Motion

Picture Association of America (MPAA) and Entertainment Software Rating Board (ESRB), which set standards for film rating systems – mobile apps do not. Instead of having standard rating rules across platforms, each mobile platform establishes its own rating policy and rating strategy. For example, Android maturity rating policy contains four maturity-rating levels: “Everyone,” “Low Maturity,” “Medium Maturity,” and “High Maturity,” while iOS’s policy provides four different maturity-rating levels based on the suitable age of audience: “4+,” “9+,” “12+,” and “17+.” Both rating systems classify types of objectionable content into four maturity levels, and their classification rules for each level are similar except some minor differences. For instance, apps with intense usage of offensive language are rated as “Low Maturity” (maturity level 2) on Android platform, but they are “12+” (maturity level 3) on iOS.

In terms of *implementing* maturity rating policy, the main difference between iOS and Android platforms is *who* determines or reports the actual ratings. iOS rates each app submitted according to its own policies, but the Android’s rating system is not as centralized. In fact, a centralized maturity rating system for Android apps’ is absent. The maturity ratings for Android apps are purely a result of app developers’ self-report. Developers are required to choose one from the four maturity levels before publishing their apps. After submitting to the Google Play Store, an app is available for download in just a few hours. Google does not verify each app’s maturity rating unless there are a number of user complaints. The public may raise concerns about the authenticity of the maturity ratings of Android apps, but this requires diligent policing on the part of the end user community. In contrast, iOS has a more strict review process for newly released apps. Apple first requires developers to select from a list of objectionable content and indicate the intensity of the content to generate the maturity rating. According to Apple’s “App Store Review Guidelines,” Apple examines the contents of apps and adjusts any inappropriate ratings during a review process before the app becomes available to users (Apple, 2012a).

Due to the laxity of Android's maturity rating policy and the lack of objective judgment of apps' maturity levels provided by developers, many news articles have recognized the drawbacks of Android's rating system. They claim that the Android rating policy is unclear, and it is difficult for developers to understand the difference between the four maturity-rating levels (Rasmussen, 2011). In addition, according to the Washington Post (Kang, 2011) and recent reports from Federal Trade Commission (FTC, 2012, 2013), there is a rising concern among parents who have experienced that the maturity ratings of the apps are unreliable. However, according to our knowledge, little systematic research has been conducted to analyze the problems with Android's maturity rating policy and its implementation, not to mention uncovering the risk level of Android apps for children's protection. Therefore, this work is designed to fill this gap.

We contribute the following:

1) We develop a text mining algorithm to automatically predict an app's actual maturity rating from the app description.

2) By comparing Android ratings with iOS ratings, we illustrate the percentage of Android apps with incorrect maturity ratings and examine the types of apps which tend to be misclassified.

3) We conduct some preliminary analyses to explore the factors that may lead to untruthful maturity ratings in Android apps.

4.2 Research Questions

As discussed earlier, there is no standard rating policy for mobile apps. The maturity ratings of Android apps are provided purely by self-report and are rarely verified. While iOS app ratings are considered to be more accurate, they can also be inconsistent with Apple's published policies (Rasmussen, 2011). Therefore, the first research question is to ascertain:

1. Does iOS rating strictly reflect its policy?

Although iOS's implementation of ratings and its announced policy may be slightly different, Apple's review procedure is still generally accepted as strict and objective. Apple's review guidelines put emphasis on apps' descriptions being relevant to the application content (Apple, 2012a). Therefore, the brief introduction to the content in the apps' description is a good data source for maturity rating prediction. Therefore, if Apple's rating is reflected in the description of an app, the maturity ratings can be automatically learned and applied to new apps. Thus, the second research question is raised as:

2. Are app ratings reflected in app descriptions? If so, can we build an effective text mining approach to predict the true rating of an app?

Since iOS maturity ratings cause few complains, we exam whether they are consistent and true. If iOS maturity ratings are truthful maturity ratings, by comparing maturity ratings on iOS and Android, we can further reveal the reliability of maturity ratings on Android. Our third research question is to ascertain:

3. Do Android developers provide accurate maturity ratings for their own apps? For apps published in both markets, are Android ratings consistent with iOS ratings?

If Android developers are found to provide incorrect maturity ratings for their own apps, this study also attempts to identify the factors for the incorrect ratings. Therefore, the last research question is:

4. What are the factors that could lead to untruthful maturity ratings in Android apps in comparison to iOS apps?

4.3 Methodology

In order to answer the above research questions, we first analyzed the difference between iOS's rating policy and implementation to assure that iOS's rating scheme can be used as a baseline for verifying the reliability of Android's ratings. Based on the iOS's ratings, a text-mining algorithm ALM was developed by analyzing the app descriptions and users' reviews in order to predict the maturity ratings of apps.

4.3.1 iOS Maturity Rating Policy vs. Implementation

As shown in Table 4, the maturity rating policy of iOS (Apple, 2012b) contains four levels based on user's age: "4+", "9+", "12+", and "17+". Its rating policy describes four categories of mature content: violence, offensive language, sex, and other. To clearly identify the categories, we manually categorized iOS maturity rating policy and each category was given an abbreviation:

- The *violence* category includes cartoon/fantasy violence (*A*) and realistic violence (*B*).
- The *sex* category includes suggestive themes (*D*) and sexual content (*E*).
- The *offensive language* category includes profanity and crude humor (*F*).
- The *other* category includes drug/alcohol/tobacco usage (*G*) and simulated gambling (*H*).

In Table 4, we also assign "*1*" or "2" to indicate the intensity or frequency of the above mentioned harmful category: "*1*" denotes mild/infrequent appearance, and "2" denotes the intense/frequent appearance. For example, *A1* means mild/infrequent appearance of cartoon and fantasy violence, while *E2* means intense/frequent appearance of sexual content.

During the implementation of its policy, iOS provides detailed reasons for each maturity rating. For example:

Rated 9+ for the following:

- *Frequent/Intense Cartoon or Fantasy Violence*

Apps may be rated to a specific maturity level for containing a single type or multiple types of objectionable content. For apps rated for containing a single type of objectionable content, the type of objectionable content becomes a “dominant reason” for the maturity level. For example, there are apps rated as “17+” only for containing “frequent/intense sexual content or nudity (*E2*)”, *E2* is determined as a dominant reason for maturity level “17+”.

The dominant reasons for each rating level are selected by a bottom-up search from maturity rating 9+ to rating 17+ in actual app ratings.

Our data source was the total of 1,464 iOS apps described in Section 5.1. First, we identify all the dominant reasons cited for rating 9+, including *A1, A2, B1, C1, D1,* and *F1*. These objectionable contents become unessential reasons for rating 12+ and rating 17+. By removing the unessential reasons from 12+ apps, some of the apps, previously containing multiple types of objectionable content, now only contain one type of objectionable content. Similarly, the dominant reasons for rating 12+ can be determined, including *B2, C2, E1, F2, G1, H1, H2*. Lastly, the dominant reasons for 17+ can be determined as *D2, E2, G2*. Table 5 summarizes the findings from actual apps with cited reasons for each rating. We find that an app’s maturity rating does not boost to the next level even if it contains all dominant reasons of the lower levels. Thus, the dominant reasons are necessary and sufficient. By comparing Table 4 and Table 5, we can conclude that Apple’s actual rating policy is quite different from its official rating policy.

By comparing iOS official rating policy and actual rating practice, we find the following main differences:

Violence category: 1) The reason "frequent/intense cartoon, fantasy violence" (*A2*), listed in both 12+ and 17+ in the iOS official policy, leads to 9+ in actual ratings. 2) The reason "frequent/intense horror themes" (*C2*), listed in rating 17+ in the iOS official policy, leads to 12+ in actual ratings.

Table 4. Apple iOS official maturity rating policy

Maturity levels	Violence	Sex	Offensive language	Other
4+	-	-	-	-
9+	Mild/infrequent cartoon, fantasy (*A1*) or realistic violence (*B1*), or infrequent/mild horror themes (*C1*)	Infrequent/mild mature, suggestive themes (*D1*)	-	-
12+	Frequent/intense cartoon, fantasy (*A2*) or realistic violence (*B2*)	Mild/infrequent mature or suggestive themes (*D1*)	Infrequent mild language (*F1*)	Simulated gambling (*H1,H2*)
17+	Frequent/intense cartoon, fantasy (*A2*) or realistic violence (*B2*), Frequent/intense horror themes (*C2*)	Frequent/intense mature and suggestive themes (*D2*), Sexual content, nudity (*E1,E2*)	Frequent/ intense offensive language (*F2*)	Alcohol, tobacco, drugs (*G1,G2*)

Table 5. Apple iOS actual maturity rating policy derived from reasons Apple cited to rate each app

Maturity levels	Violence	Sex	Offensive language	Other
4+	-	-	-	-
9+	Cartoon, fantasy violence (*A1, A2*), Infrequent/mile realistic violence (*B1*), Infrequent/mile horror/fear themes (*C1*)	Infrequent/mile mature and suggestive themes (*D1*)	Infrequent/mild profanity or crude humor (*F1*)	-
12+	Frequent/intense realistic violence (*B2*), Frequent/intense horror/fear themes (*C2*)	Infrequent/mile sexual content, nudity (*E1*)	Frequent/intense profanity or crude humor (*F2*)	Infrequent/mile alcohol, tobacco, drugs use or references (*G1*), simulated gambling (*H1, H2*)
17+	-	Frequent/intense mature and suggestive themes (*D2*), Frequent/intense sexual content,	-	Frequent/intense alcohol, tobacco, drugs use or references (*G2*)

		nudity (*E2*)		

Table 6. Maturity rating policy for Android applications

Maturity levels	**Violence**	**Sex**	**Offensive language**	**Other**	**Social feature**	**Location**
Everyone	-	-	-	-	-	-
Low maturity	Mild cartoon, fantasy violence (*A1*)	-	Potentially offensive content (*F1*)	-	Some social features but not allow user to communicate (*I1*)	Collect for service (*J1*)
Medium maturity	Intense fantasy (*A2*) or realistic violence (*B2*)	Sex reference (*D1*, *E1*)	Profanity or crude humor (*F2*)	References to drug, alcohol and tobacco use (*G1*), and simulated gambling (*H1*)	Social features allow user to communicate (*I2*)	Collect for sharing (*J2*)
High maturity	Graphic violence (*B3*)	Frequent instances of sexual (*D2*), and Suggestive content (*E2*)		Strong alcohol, tobacco, drug (*G2*), and Strong simulated gambling (*H2*)	Social features allow user to communicate (*I2*)	Collect for sharing (*J2*)

Offensive language category: 1) The reason "infrequent/mild language" (*F1*), listed in 12+ in the iOS official policy, leads to 9+ in actual ratings. 2) The reason "frequent/intense offensive language" (*F2*), listed in rating 17+ in the iOS official policy, leads to 12+ in actual ratings.

Sex category: The reason "infrequent/mild sexual and nudity content" (*E1*), listed in rating 17+ in the iOS official policy, leads to 12+ in actual ratings.

Other category: The reason "infrequent/mild alcohol, tobacco, drug use" (*G1*), listed in rating 17+ in the iOS official policy, leads to 12+ in actual rating.

Based on this analysis, we can see that iOS actually downgrades its official maturity policy during implementation. The inconsistency between iOS official policy and its actual ratings could cause problems. When parents view an app's description page at the iOS store, they may be misled. Parents who intend to choose apps with maturity rating 12+ for their children to avoid exposure to horror content, frequent offensive language, and sexual/nudity content may actually get an app that contain all aspects of such undesirable content. The only avenue to avoid this situation is to read through all the reasons, which requires significant effort on the part of the parent. Yet, parents are frequently unaware of the discrepancies between the actual maturity rating and the official policy and instead trust the actual maturity ratings as they are listed.

4.3.2 Android Apps' Maturity Ratings

As presented in Table 6, we manually categorized the Android maturity rating policy. Android has its own rating policy with four levels (Google, 2012): "Everyone", "Low Maturity", "Medium Maturity", and "High Maturity". Compared to the iOS maturity rating policy, Android's policy contains two additional categories (i.e., social feature and location) with five additional tokens. The additional tokens are: the social features that disallow users to communicate (*I1*) and the social features that allow users to communicate (*I2*); collecting user locations for service (*J1*), and collecting user locations for sharing (*J2*); the token value "*3*" represents the graphic appearance of violence content.

Table 6 presents the basic rules for differentiating Android apps' maturity levels. The rating of "Everyone" means there is no harmful content. The rating of "Low Maturity" means that violent content, offensive language, social feature, and collection of location information may appear, but are mild and infrequent with minimal effect on children's mental health and privacy. With the rating of "Medium Maturity", all six categories of objectionable content (i.e., violence, offensive language, sex, other, social feature and location.) may appear intensely and frequently with the exception of sex content, alcohol/tobacco/drug, and gambling content which are illegal for minors under 18 to view. Apps belonging to the level of "Medium Maturity" are arguably harmful for children under 13 years old for viewing or engaging. Finally, the highest maturity level – "High Maturity" contains content for adults only such as significant sexual and violent content (see Table 6). It seems that Android's maturity rating policy is reasonable and clear. However, the reliability of Android's actual ratings by developers remains a question because the actual ratings largely rely on developers' comprehension and assessment of the policy.

Our comparison of Android's maturity rating policy (Table 6) with the iOS's actual maturity rating policy (Table 5) reveals that both platforms categorize maturity ratings into four levels: "Everyone", "Low Maturity", "Medium Maturity", and "High Maturity" in Android; and "4+", "9+", "12+", and "17+" in iOS.

Table 7. Comparison of Andriod's Policy and iOS's Policy

Maturity levels	Maturity rating levels by Android	Maturity rating levels by iOS
1	Everyone	4+
2	Low maturity	9+
3	Medium maturity	12+
4	High maturity	17+

Because the maturity levels for content in each category are mostly similar (see Table 7), we argue that iOS's maturity rating scheme is reasonably reflected in that of Android's maturity rating scheme, except the following discrepancies:

- Android does not consider horror content (*C*) as mature content, while iOS does include horror content (*C*) as mature content.
- Android considers graphic violence (*B3*) as mature content while iOS directly rejects apps with graphic violence.
- Android integrates privacy protection in its maturity rating policy by including the social feature (*I*) and location collection (*J*). However, no corresponding privacy-related consideration exists in the maturity rating scheme by iOS.
- Frequent/intense cartoon violence and fantasy violence (*A2*) is rated as "Medium Maturity" (i.e., level 3) in Android but as "9+" (i.e., level 2) in iOS.

- Frequent/intense simulated gambling (*H2*) is rated as “High Maturity” (i.e., level 4) in Android but is rates as “12+” (i.e., level 3) in iOS.

Thus, to use iOS actual maturity rating as a baseline for measuring the reliability of self-reported maturity ratings on the Android platform, we had to exclude those cases in which maturity ratings contain the above schemes reflecting the discrepancies between iOS and Android’s rating policies. After such exclusions, Android’s policy and iOS’s actual rating scheme should be the similar for all the remaining maturity ratings. Thus, we can now use iOS actual maturity rating as a baseline to examine the reliability of Android apps’ maturity ratings.

4.3.3 Comparing Apps on iOS and Android

After establishing the baseline for evaluation, we match apps from Google Play with those same apps on the iOS App Store. The index scheme for each app on iOS and Android is different (each Android app has a unique package name that serves as the application ID; and each iOS app has a unique Apple ID). However, apps’ names are often consistent across the platforms of iOS and Android for branding purpose.

We used a program to automatically search the iOS App Store based on apps’ names collected from Google Play. It reveals that the same app for both platforms could have slightly different names. Thus, for each Android app, we choose up to 150 search results from the iOS App Store. For those showing similar app names, we conducted analysis to determine the closest fit.

```
    **Algorithm:** CalculateEditDistance
    **Input**: Android app name $N_{Android}$ , results returned by iTune  $N_k$
    **Output**: The edit distance, $Ed_k$, of, $N_{Android}$, and, $N_k$.
    $Ed_{k1} = SearchDifferentWords\ (N_{Android}, N_k)$
    $Ed_{k2} = SearchDifferentWords\ (N_k, N_{Android})$
    **RETURN** $min(Ed_{k1}, Ed_{k2})$

    **FUNCTION** $SearchDifferentWords\ (N_{Android}, N_k)$
Split, $N_{Android}$, into a word set, $W_{Android} = \{a_1, a_2, \cdots, a_i\}$
Split, $N_k$, into a word set, $W_k = \{b_1, b_2, \cdots, b_j\}$
**FOR** each, $a_i \in W_{Android}$
   **IF** $\exists b_j \in W_k$, equals or only one letter different from, $a_i$
        Delete, $b_j$, from, $W_k$
   **END IF**
**END FOR**
    **RETURN** number of words left in, $W_k$
    **END**
```

Specifically, to find the iOS app whose name is mostly similar to an Android app, we use minimum edit distance between the Android app name and each returned iOS app name to estimate the similarity of two names (see the above algorithm). For an Android app, denote its name by $N_{Android}$. For each $N_{Android}$, the search result from the App Store produces a set, $S = \{N_1, N_2, \cdots, N_k, \cdots, N_{result\_size}\}$ $(0 < result\_size < 150$. For each N_k , the algorithm *CalculateEditDistance* returns the name N_{iOS} which is the minimum edit distance to $N_{Android}$ in S.

To confirm that an iOS and Android app pair with similar names is the same app, their descriptions and developers' company names were further compared. Finally, the confirmed similar apps' icons and screenshots were visually compared by two individual researchers to ensure that these two apps in Android and iOS were the same. The selected app pairs were then used as our comparison dataset.

4.3.4 ALM—Automatic Label of Maturity Ratings for Mobile Apps

If an Android app has an iOS version, the algorithm in the previous subsection is sufficient to identify the same app on iOS. However, not all Android apps have a counterpart in the iOS App Store. We label these apps as “Android-only” apps. For Android-only apps, we need to determine their actual maturity rating (not based on the ratings provided by their developers). Thus, we propose a text-mining-based Automatic Label of Maturity ratings (ALM) algorithm. ALM is a semi-supervised learning algorithm, and it processes apps’ descriptions and user reviews to determine maturity ratings. The more technical detail of ALM is described below.

4.3.4.1 Building seed-lexicons for objectionable content detection

For the iOS apps in a training dataset, we group their descriptions according to the contained objectionable contents. Apps containing only one type of the objectionable content are organized based on their rating scheme together with their corresponding token, such as *A1.txt*, *A2.txt*, *B1.txt*, and *H2.txt*. For example, the *A1.txt* file contains the descriptions and users’ reviews of all the apps whose maturity ratings are “9+” caused by “infrequent/mild cartoon and fantasy violence” (A1).

I train two other coders to read grouped app descriptions and select seed lexicons to detect objectionable content. The coders are asked to pick up terms contributing to the maturity ratings. For example, for an app rated by iOS as “infrequent/mild cartoon and fantasy violence” (A1), the app descriptions may include words such as “gun”, “cannon”, “shooting”, “hunting”, “racing”, and “attacking”. And for an app rated by iOS as “Infrequent/mile mature and suggestive themes (D1)”, the app descriptions may include words such as “dating”, “boyfriend”, “girlfriend”, and “nightclub”. The coders are trained to use their best judgments to pick up such terms to assist further detection. Different coders may select different set of words. The final lexicon is determined by discussion, the most discriminative words are selected. This manual labeling procedure produces the seed-lexicons for each mature content category. The coders are then asked to extract as many terms as possible. However, for each type of objectionable content, only 10~20 common seed terms are selected. Therefore, the manual process is not labor intensive. This manual process takes advantage of human knowledge to set the start point of the detection. Furthermore, unsupervised learning algorithms are also used to add statistical significant terms into the lexicon, and also adjust the weights of each term in the detection.

After the seed-terms are generated for each type of objectionable content, they are grouped into three bigger lexicons denoted as T_i, $i \in 9, 12, 17$ for classifying the maturity rating: 9+, 12+, and 17+ (as shown in Table 8). T_i presents the objectionable contents for each level i, and it only includes nouns, verbs, adjectives, and adverbs in this study.

Table 8. Group seed-lexicons for classification

Grouped Lexicon	**Seed-lexicons**
17+	*D2*, *E2*, *G2*
12+	*B2*, *C2*, *E1*, *F2*, *G1*, *H1*, *H2*
9+	*A1*, *A2*, *B1*, *C1*, *D1*, *F1*

4.3.4.2 Assigning initial weights to seed-terms

Next we assign initial weights for the terms in the seed-lexicons. To do so, positive instances and negative instances are separately grouped into sets for each maturity level, as shown in Table 9.

Table 9. Positive and negative instances for maturity classification

Maturity levels	**Positive instances set**	**Negative instances set**
17+	Apps rated 17+ by iOS	Apps rated 4+, 9+, 12+ by iOS
12+	Apps rated 12+ by iOS	Apps rated 4+, 9+ by iOS
9+	Apps rated 9+ by iOS	Apps rated 4+ by iOS
4+	Apps rated 4+ by iOS	-

The set of positive instances and negative instances are denoted as P_i and N_i respectively for level $\mathrm{i} \in 9, 12, 17$. For each seed-term $\mathrm{t} \in \mathrm{T_i}$, denote its frequency in P_i and N_i as t_p and t_n, respectively. Therefore, the initial weight of t can be calculated using Equation (1).

$$w_t = \begin{cases} t_p & if\ t \in P_i \backslash N_i, \\ -t_n & if\ t \in N_i \backslash P_i, \\ \frac{t_p}{t_n} & if\ t \in P_i \cap N_i, \\ 0 & if\ t \notin P_i \cup N_{i.} \end{cases} \qquad (1)$$

4.3.4.3 Classification

Once the seed-terms and their weights are generated, we can calculate the apps' maturity ratings. For each app a, all terms in its description are selected and categorized as a set $A = t_k$. We further define a random threshold α to differentiate positive instances with negative instances. The value of α does not affect the result, because in the training phase, the *Expand_Adjust* algorithm (described in the next section) will adjust term weights to fit the threshold, and later the adjusted weights and the threshold are used together to calculate the maturity rating of test instances. Thus, for app a, its maturity rating m_a can be determined by Equation (2) and Equation (3).

$$s_i^a = \sum_{t_j \in T_i} (t_j * w_{t_j}) \qquad (2)$$

$$m_a = \begin{cases} 17+ & if\ s_{17}^a > \alpha \\ 12+ & if\ s_{17}^a < \alpha,\ s_{12}^a > \alpha \\ 9+ & if\ s_{17}^a, s_{12}^a < \alpha,\ s_9^a > \alpha \\ 4+ & otherwise \end{cases} \qquad (3)$$

4.3.4.4 Expanding seed-lexicons and adjusting weights

Through the human-assisted process above, we found that the classification accuracy in determining apps' maturity ratings with only the seed-lexicons and their initial weights is around 70%. This is because seed lexicons can only partially reflect objectionable content. However, other terms that appeared frequently in the positive instances should also be added to the lexicon set to further improve the accuracy of classification. In addition, weights of terms should be further adjusted to suit the content. Therefore, we further use the unsupervised learning algorithm *Expand_Adjust* to add frequent terms in the positive instances into consideration, and our algorithm automatically adjusts the weights for both seed terms and non-seed terms, to find an optimal balance of precision and recall in the classification.

All terms in $P_i \cup N_i \backslash T_i$, $i \in 9, 12, 17$, are categorized into the non-seed-terms set T_{ns}, and initial weights of the terms in T_{ns} are 0. As the auto-labeling algorithm runs, instances may be mis-classified. Therefore, sets F_p and F_n are denoted to present the false positive and false negative set, respectively.

Once the terms and weights are optimized, apps' maturity ratings can be estimated by the classification algorithm described in the previous subsection.

```
        Algorithm: Expand_Adjust
        Input: Positive instance set $P$, negative instance set $N$ , false positive set $F_p$ , false negative set $F_n$ , weights of all terms: $W = \{w_t\}$, seed-term set $T_s$ , non-seed-term set $T_{ns}$ .
        Output: Updated weights of all terms: $W = \{w_t\}$
        WHILE (the size of $F_p$ and $F_n$ can further be decreased)
  $W$ = DecreaseFN ( $T_s$ , $W$ );
  $W$ = DecreaseFP ( $T_s$ , $W$ );
  $W$ = DecreaseFN ( $T_{ns}$ , $W$ );
  $W$ = DecreaseFP ( $T_{ns}$ , $W$ );
        END WHILE

        FUNCTION DecreaseFN ( $Term\ set\ T_0$ , $W$ )
  WHILE (size of $F_n$ can further be decreased)
     FOR ($t \in T_0$ )
        Find max ($w_t$) which can decrease size of $F_n$, but not increase size of $F_p$)
     END FOR
  END WHILE
  RETURN $W$
        END
        FUNCTION DecreaseFP ( $Term\ set\ T_0$ , $W$ )
  WHILE (size of $F_p$ can further be decreased)
     FOR (each $t \in T_0$ )
        Find min ($w_t$) which can decrease size of $F_p$, but not increase size of $F_n$)
     END FOR
  END WHILE
  RETURN $W$
        END
```

4.4 Experiment

In this section, we describe our experimental dataset, design and results.

4.4.1 Data Collection

An automatic crawler was built to collect data from the Google Play Store. The crawler ran for a week from 9/26/12 to 10/2/12, and collected two datasets. The first dataset was a pretest dataset which contained metadata from 1,000 Android apps—the top 500 paid apps and top 500 free Android apps from all categories on the Google Play Store. This dataset was used to conduct an initial assessment on the types of apps that frequently received incorrect maturity ratings. Using the iOS app counterparts, our result showed that the category of "Games" received the most incorrect maturity ratings (see Fig.8). Given that the category of "Games" was shown to be the most popular category for app download (NielsenWire, 2011), our second round of data collection focused only on Android apps in the category of "Games". Since we have found that iOS maturity ratings are reasonable, we compare apps' iOS maturity ratings with their Android maturity ratings. The ones with unmatched ratings are presented in Fig. 8.

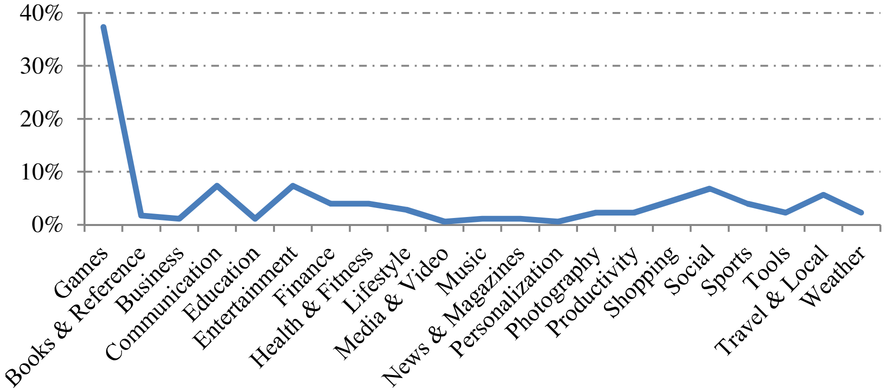


Figure 8. Distribution of apps with unmatched maturity ratings by different categories from dataset 1

The second dataset (main dataset) contained the metadata and user reviews crawled from 5,059 apps in the category of "Games". Metadata included a rich spectrum of information such as app package name (a unique id for each Android app), app name, developer's name, developer's company, developer's website, category, price, currency, number of installations, icon, screenshot, permission, and description. The collected apps were equally distributed among 8 different subcategories of games: arcade & action, brain & puzzle, cards and casino, casual, live wallpaper, racing, sports games, and widgets. A total of 729,128 user reviews were collected, resulting in 144 reviews per app on average.

For each Android app, we searched through iOS App Store using the method descripted in section 4.3. A total of 1,464 apps were found on iOS App Store and the remaining 3,595 apps were classified as Android-only apps.

4.4.2 Experiment 1: Predicting Apps' Maturity Ratings by the ALM algorithm

For Android-only apps, we used the ALM algorithm described in Section 4.4 to automatically label maturity ratings. In this experiment, the 1,464 apps which are available on both Android and iOS were used as the training set, and the 3,595 Android-only apps were used as the testing set.

We conducted a 10-fold classification on the training set. Standard evaluation metrics for classification, precision, recall, and f-score were used as our evaluation metrics. In particular, precision presents the percent of identified positive instances that are truly positive instances. Recall measures the overall classification correctness, which represents the percent of actual positive instances that are correctly identified. F-score represents the weighted harmonic mean of precision and recall, which is defined as:

$$f - score = \frac{2(precision * recall)}{precision + recall} \qquad (4)$$

The performance of the ALM algorithm on the training set is presented in Table 10, the maturity ratings estimated by ALM is compared to ratings given by iOS (i.e., the ground truth). ALM achieved high precision in maturity rating detection across all maturity levels. It performed extremely well in detecting maturity ratings of "17+" and "4+". This is intuitive because apps with high maturity rating normally contain extreme mature content, while apps with low maturity rating often do not contain any mature content. Therefore, it seems reasonable to conclude that ALM is most effective and less error prone in detecting extreme cases. For apps with maturity rating "12+" and "9+", ALM's performance was slightly lower due to the infrequent and subtle mature content.

Table 10. Detection result of ALM on the training set

Maturity levels	# of positive instances	# of negative instances	# of seed terms	# of expanded terms	Precision	Recall	F-score
17+	31	1,433	48	67	100%	100%	100%
12+	229	1,204	176	282	96.6%	99.6%	98.1%
9+	155	1,049	134	235	93.9%	99.4%	96.6%
4+	1,049	0	0	0	99.8%	98.4%	99.1%

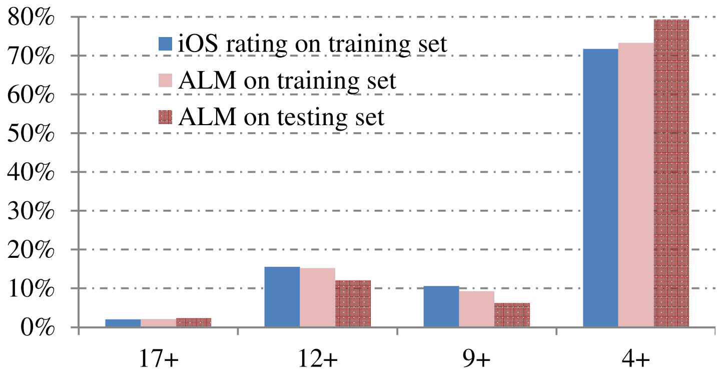


Figure 9. Distributions of apps in training and testing set

Once we verified that ALM performed effectively on the training set, we applied the same model to the testing set and compared the distributions of the detection results between the training and testing sets (as Fig. 9). We observe that the distributions of the two sets are similar. Thus, it is reasonable to use ALM to predict true maturity ratings of an app. With ALM, the maturity ratings for Android-only apps can also be verified. Currently, Apple takes a long time to manually rate every single app submitted by third party developers. If this labor-intensive procedure can be assisted by automatic rating algorithms such as ALM, the app maturity rating process for iOS apps can be significantly shortened.

4.4.3 Experiment 2: Overrated and Underrated Android Applications

For any Android app, a, its maturity rating provided by the developer, is denoted as m_a^A. Define the maturity level l_a^A as:

$$l_a^A = \begin{cases} 1 & if\ m_a^A = "Everyone", \\ 2 & if\ m_a^A = "Low\ Maturity", \\ 3 & if\ m_a^A = "Medium\ Maturity", \\ 4 & if\ m_a^A = "High\ Maturity". \end{cases} \quad (5)$$

Similarly, the actual maturity rating for the app a is denoted as m_a^i, where m_a^i is the maturity rating on iOS if the app is available on Apple Store, or it is the predicted maturity rating by ALM otherwise. Then its maturity level l_a^i can be expressed as:

$$l_a^i = \begin{cases} 1 & if\ m_a^i = "4+" \\ 2 & if\ m_a^i = "9+" \\ 3 & if\ m_a^i = "12+" \\ 4 & if\ m_a^i = "17+" \end{cases} \quad (6)$$

Therefore, if $l_a^A > l_a^i$, the app a is overrated and the overrating level is $l_a^A - l_a^i$. If $l_a^A < l_a^i$, the app a is underrated and the underrating level is $l_a^i - l_a^A$. In our dataset 2, among the 1,464 apps that were available on both Android and iOS, 265 apps (18.1%) were overrated (i.e., their maturity ratings on Android were higher than on iOS), and 142 apps (9.7%) were underrated (i.e., their maturity ratings on Android were lower than on iOS).

4.4.3.1 Overrated Android Applications

Of the 265 overrated Android apps, there were 4 apps (1.5%) with an overrating level of 3, which means those apps were rated as “High Maturity” on Android but only rated as “4+” on iOS. In addition, there were 46 Android apps (17.4%) with an overrating level of 2, and 215 Android apps (81.1%) with an overrating level of 1.

Next, we discuss some possible reasons that could be counted for the overrating phenomenon.

Intelligence (i.e., intellectual/cognitive maturity). Children's maturity level includes both intellectual/cognitive and emotional maturity[6]. Emotional maturity can also be interpreted as vulnerability. The current Android and iOS maturity rating policies only consider vulnerability (i.e., objectionable content) but not consider intellectual/cognitive maturity. Android's self-reporting system requires developers to fully comprehend its rating policy. However, many developers do not understand the rating policy, and overrate their app considering users' intellectual/cognitive maturity. For example, a chess game is rated as "Medium Maturity" but not "Everyone", because the developer may think that children younger than 12 years-old are not capable of playing the chess game. Because of this reason, many games in the subcategory "Brain & Puzzle" are overrated. Similarly, developers may also think that maturity ratings should reflect users' capability to complete some tasks such as wearing makeup, making cakes, taking care of pets, decorating houses, constructing cities, and running businesses. Android apps with these types of contents are often overrated. iOS has started to rate their apps considering users' intellectual/cognitive maturity. iOS rates each app inside the *Kids* category as for "5 years old and under", "6~8 years old", or "9~11 years old". Those ratings help parents to determine whether children can be expected to have the cognitive skills to use an app, but the current maturity system only tells parents about whether children should be allowed to use the app.

Simulated Gambling. As discussed earlier, inconsistencies exist between the maturity rating policies of Android and iOS. One of these inconsistencies lies in which maturity level simulated gambling should belong to. On iOS platform, the ratings are clear. Casino games—such as card games, bingo, bridge, backgammon, coin games, mahjong, slots, domino, poker, and

---

[6] The National Academies, Understanding Maturity and Vulnerability, http://www.nap.edu/netsafekids/protect_und.html

etc—do not necessarily involve gambling unless they require players to bid on real money. However, on Android platform, casino apps not involving bidding on real money can either be rated as simulated gambling or not, because Android didn't give developers strict definitions on what is an instance of simulated gambling, which causes lots of confusion.

Violence. With the controversy in the US about gun laws, that some people would judge anything with guns involved as violent while others would not, the difference of opinion about violence among the developers also cause undetermined maturity ratings on contents such as: gun shooting, cannon shooting, hunting, racing, and attacking territories.

Mature and Suggestive Themes. The last item causing confusion for developers is the definition of mature and suggestive themes. For example, is "dating" suggestive? Is the term "boyfriend/girlfriend" suggestive? How about "nightclub"? In these circumstances, developers have different opinions on maturity ratings.

The distribution of overrated Android apps is shown in Fig. 10. To alleviate the overrating problem, we suggest that Android maturity rating policies clearly define the meaning of maturity ratings as an indicator of harmful content for children or adolescents. Maturity rating should not reflect users' capabilities or intelligence levels as it causes undue confusion about ratings. In addition, Android maturity policy should provide clearer definitions and detailed explanations about the meanings of "simulated gambling", "violence", and "mature and suggestive themes", to guide developers to correctly rate their Android apps.

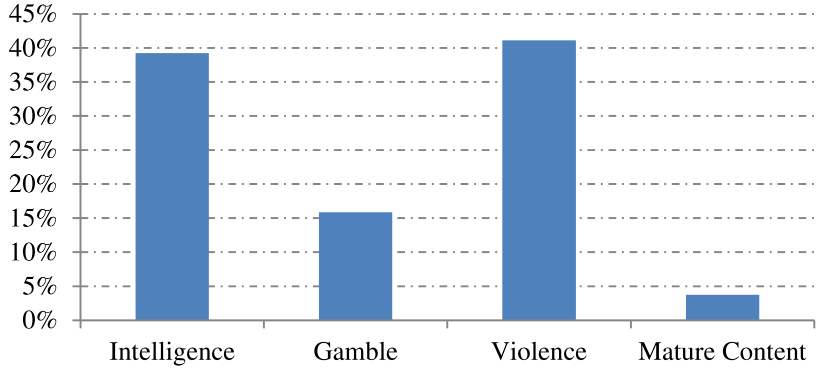


Figure 10. Distribution of overrated apps

4.4.3.2 Underrated Android Applications

Among those 142 Android apps that were underrated, 36 apps (25.4%) were underrated by 2 levels, and 106 apps (74.6%) were underrated by 1 level. Among the apps underrated by 2 levels, only 1 app was underrated from “17+” to “Low Maturity”; and 35 apps were underrated from “12+” to “Everyone”. Among the apps underrated by 1 level, 5 apps were underrated from “17+” to “Medium Maturity”; 25 apps were underrated from “12+” to “Low Maturity”; and 76 apps were underrated from “9+” to “Everyone”.

As shown in Fig. 11, most apps underrated by 2 levels contained content such as alcohol, tobacco, drug, and gamble; while apps underrated by 1 level often contained cartoon, fantasy violence, mature content, and suggestive themes.

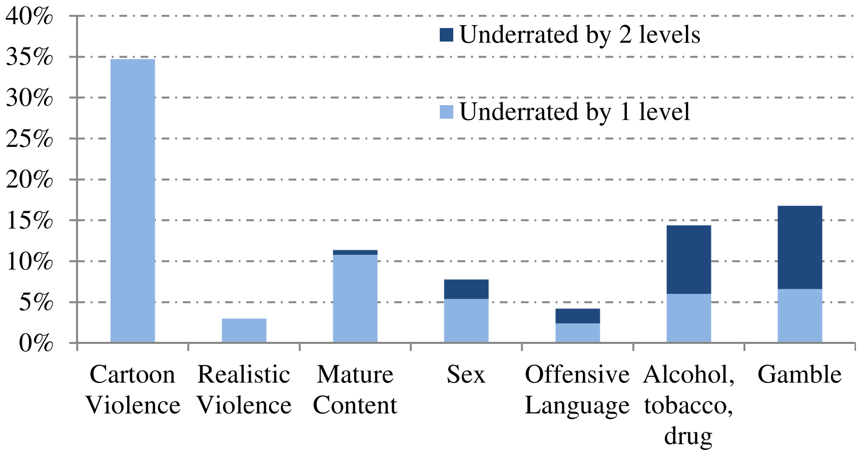


Figure 11. Distribution of underrated apps

Unlike overrated apps, the underrated apps may directly harm children's mental health, because those apps conceal their actual maturity levels to parents and minors. As shown in Figure 4, the underrated apps might contain violent content, mature and sexual content, offensive language, alcohol, drug, or gambling. Unfortunately, Google rarely verifies the maturity ratings provided by developers unless complaints are raised. Therefore, to be certain whether or not an Android app contains harmful content, parents have to painfully test the app themselves or search for the same app in iOS App Store to verify the maturity rating. Our proposed ALM algorithm can better assist parents and children to make decisions when they choose apps on the platform of Android.

4.4.4 Experiment 3: Exploring Factors Contributing to Incorrect Ratings

As discussed earlier, there is a substantial portion of Android apps with inaccurate maturity ratings. In this section, we conduct some preliminary analyses to explore the factors that may lead to the inaccurate maturity ratings.

We first captured two categories of Android apps' attributes from our dataset: apps' attributes and developers' attributes. The apps' attributes included: *popularity, price, and dangerous level of the required permissions*. Developers' attributes included: *general privacy awareness, trustworthiness, actual privacy awareness, and child safety awareness*.

For apps' attributes, the number of installations was used to infer an app's popularity. Although app rank can be retrieved to represent popularity, the ranks change every day and it is difficult to keep tracking. An app's price is assigned to a binary variable: *1* if it is a paid app and *0* if it is a free app. For apps' required permissions, Chia et al (Chia et al., 2012) divided Android permissions into three categories: *danger_info* permissions, *danger* permissions, and *ok* permissions. *danger_info* permissions consist of those permitting access to users' sensitive personal information; while *danger* permissions consist of those whose actions can be harmful to users. To clarify, *danger_info* permissions are included in the set of *danger* permissions. The permissions which belong to neither *danger_info* nor *danger* categories are *ok* permissions. This experiment adopts their definitions on the three categories of permissions. Weight values of *3*, *2*, and *1* were assigned to *danger_info*, *danger*, and *ok* permissions, respectively. Therefore, for each app, the overall score for the dangerous level of all the required permissions was generated by aggregating the weights of the permissions.

For developers' attributes, the privacy regulatory culture or norm of developers' countries is used to represent developers' general privacy awareness. Smith (Smith, 2004) divided countries into two categories based on the privacy regulatory culture or norm: *human rights* countries and *contract term* countries. According to Smith (Smith, 2004), *human rights* countries typically have comprehensive privacy regulations which address all data collection and use within the society, whereas *contract term* countries only have regulations regarding collection and use of certain types of data, which do not extend to all types of data in all sectors of the society. Since *human rights* countries have stricter privacy regulations, we argue that developers from these countries should have higher levels of privacy awareness, which should lead to higher possibility of correct maturity ratings given to their Android applications. Developers' countries are inferred from the domain of their websites, and the currency of their apps. For the countries which appeared in our dataset but were included in Smith's framework, we manually researched the privacy regulatory culture or norm of that specific country to determine its category. As a result, 18 countries were labeled as *human rights* countries, while others were all categorized as *contract term* countries (as Table 11). Score values *1* and *0* are assigned to *human rights* country and *contract term* country respectively to represent developers' general privacy awareness. The distribution of developers' countries is presented in Fig. 12. The percentage of apps from *human rights* countries was 25%, and the rest were all from *contract term* countries.

Table 11. List of human rights countries

Human rights countries
Australia, Austria, Canada, Czech Republic, Denmark, France, Finland, Germany, Hungary, Italy, Netherlands, New Zealand, Norway, Slovenia, Spain, Sweden, Switzerland, and U.K.

The evaluation of developers' websites by Web of Trust (WOT) ("Web of Trust (WOT),") was used to represent the trustworthiness, actual privacy awareness, and child safety awareness of the developers. WOT measures websites' *trustworthiness* by testing whether websites deliver reliable services; *privacy awareness* is measured by testing whether websites keep users' personal information safe; *child safety awareness* is measured by checking whether websites only contain age-inappropriate materials. WOT obtains the evaluation of websites by wisdom-of-crowd, and it provides browser add-ons so that users can evaluate visited websites. As of November 2012, WOT has collected evaluations for over 52 million websites. Therefore, WOT's evaluations on the above three categories can reflect the attributes of Android developers' websites. For each category, WOT provides reputation scores (range from 1 to 99) for each category together with their confidence levels (range from 1 to 99). Thus, for each category, the general rating can be calculated by multiplying the reputation and confidence using the following equation:

$$rating = (reputation - 50) * confidence \qquad (7)$$

In this research, we are not only interested in finding out whether attributes of apps and developers affect the correctness of maturity ratings, but also whether the observed effects vary upon additional factors. In this experiment, three additional factors are examined: The first factor is *price*, with which we aim to observe whether the influences of apps' and developers' attributes on the maturity ratings would vary upon *paid* vs. *free* apps. The second factor is *general privacy awareness*, with which we aim to observe whether the influences of apps' and developers' attributes on the maturity ratings would vary upon *human rights* vs. *contract term* countries. The last factor is *platform*, with which we aim to observe whether the influences of apps' and developers' attributes on the maturity ratings would vary upon *cross-platform apps* vs. *Android-only apps*.

Pearson's correlation and linear regression were used to conduct analysis. As shown in Table 12, *price* had negative effect on overrating, which indicates that free apps are more likely to be overrated than paid ones. We notice that the negative effect of price on overrating was found to be stronger for Android-only apps than for cross-platform apps.

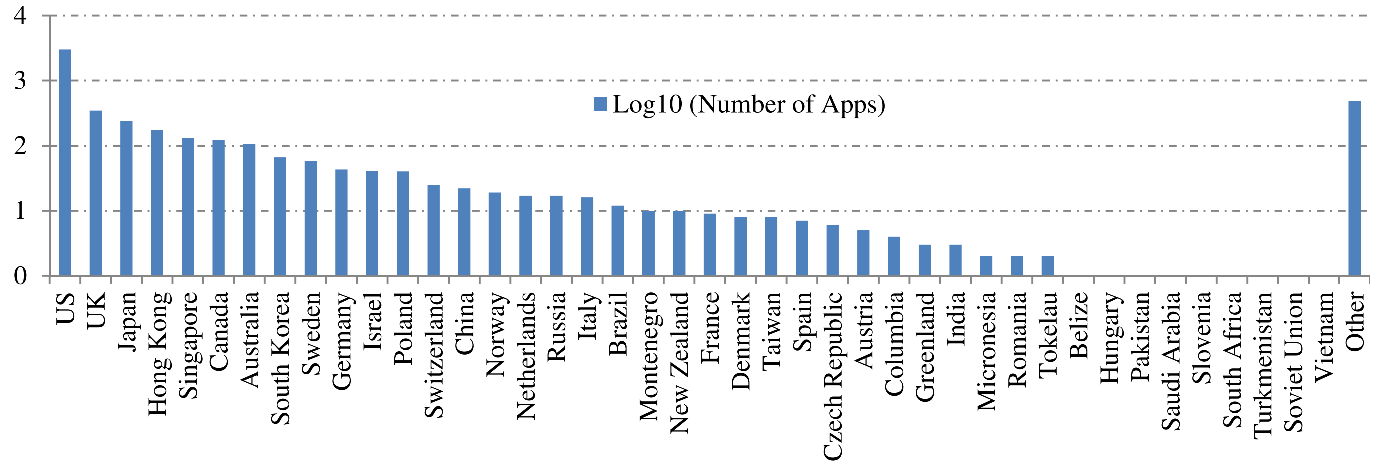


Figure 12. Distribution of developers' countries

In other words, free apps which are only available on the Android platform are more likely to be overrated. This result is interesting. Given that Android-only apps are often newly developed or published by small companies or individual developers, we argue that overrating may serve as a strategy to further attract users' attentions to those free apps. This is because, for free apps which do not require purchase to download, advertisements are the primary revenue sources. Developers are therefore more eager to promote their apps, even by giving apps the same or look-alike names as the well-known apps.

Table 12. Significant factors that affect overrating (n=265)

Impact		**CP**	**AO**	**AOSL**
Price→Overrating		-0.05^{***}	-0.25^{***}	-0.26^{***}
Overrating → Danger Permission		0.57^{***}	0.45^{***}	0.37^{***}
Impact	**Paid**	**Free**	**HR**	**CT**
Price→ Overrating	-	-	-0.21^{***}	-0.33^{***}
Overrating→Danger Permission	0.44^{***}	0.62^{***}	0.43^{***}	0.47^{***}

$^{***}p<0.001$ [*Note*. CP = cross-platform apps; AO = Android-only apps; AOSL = Android-only apps which have same or look-alike names with iOS apps; HR = human rights countries; CT = contract term countries.]

As shown in Table 12, the negative effect of price on overrating was found to be stronger in *contract term* countries than in *human rights* countries. In other words, free apps developed in *contract term* countries are more likely to be overrated than those developed in *human rights* countries.

In addition, the result showed that overrating had a significant positive effect on requesting *dangerous* permissions. That is to say, overrated apps were more likely to request more *dangerous* permissions. Such an effect was stronger for *cross-platform* apps than for *Android-only* apps, and stronger for *free* apps than for *paid* apps. These results suggest that those overrated cross-platform apps for free tend to be more data-hungry by requesting more data permissions across different platforms. Moreover, the positive effect of overrating on requesting *dangerous* permissions was stronger for apps developed in *contract term* countries than those developed in *human rights* countries.

Table 13. Significant factors affect underrating (n=142)

Impact	**CP**	**AO**	**Paid**	**Free**	**HR**	**CT**
Trust→Underrating	**-0.26^{*}**	**-0.04^{*}**				
Privacy→ Underrating	**-0.23^{*}**	0.03				
Child Safety →Underrating	**-0.07^{*}**	-0.03				
Popularity→ Underrating			0.01	**0.10^{**}**		
Price→Underrating	-0.01	**0.10^{**}**	-	-	0.01	**0.13^{*}**

$^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$ [*Note: The insignificance was not shown.*]

Data analyses were further conducted to explore factors that may lead to underrated maturity ratings. As shown in Table 13, the trustworthiness of a developer's website (evaluated by WOT) had significant negative effect on underrating. That is to say, the less trustworthy the developer's website is, the more likely that the app may be underrated. Such effect was found to be stronger for cross-platform apps than for Android-only apps.

It was found that privacy awareness level of a developer's website (evaluated by WOT) had negative association with underrating for cross-platform apps. This finding indicates that those developers with lower levels of privacy awareness are more likely to underrate the cross-platform apps, for the purpose of reaching wider user population and harvesting users' personal information.

As shown in Table 13, the child safety score of a developer's website (given by WOT) had significant negative association on underrating for cross-platform apps. That is to say, if a developer's website has a lower safety score for children, the app developed by this specific developer is more likely to be underrated.

For free apps, popularity was found to be positively associated with underrating. Such an effect was not significant for paid apps. This result suggests that popular free apps are more likely to be underrated. In addition, we found that paid apps are more likely to be underrated when developers are from *contract term* countries than those from *human rights* countries.

4.5 Conclusion

As discussed earlier, no research has systematically uncovered the extent and severity of these unreliable app maturity ratings, nor has any research shown the types of apps that are most often mis-categorized. To bridge this gap, our study develops mechanisms to verify the maturity ratings of mobile apps and investigates the possible reasons behind the incorrect ratings. Specifically, we develop a text-mining algorithm "Automatic Label of Maturity ratings" (ALM) to verify mobile apps' maturity ratings on the Android platform compared to Apple's iOS platform. ALM discovered that 27.8% of Android apps have unreliable maturity ratings, among which 18.1% apps are overrated and 9.7% apps are underrated.

Our research has several contributions. First, we practically examine the maturity rating policies on both Android and iOS platforms, and discover the inconsistencies and ambiguities from both policies. Second, based on app descriptions and user reviews, the algorithm ALM is developed to automatically verify Android apps' maturity ratings that were based on developers' self-disclosure. Experimental results show that ALM has advanced performance on detecting objectionable content in any maturity levels in terms of precision, recall and f-score. Third, we conduct some preliminary analyses to explore the factors that may lead to untruthful maturity ratings in Android apps. We believe that our findings have important implications for platform providers (e.g., Google or Apple) as well as for regulatory bodies and application developers.

Chapter 5 Exposure to Problematic Content—Mobile In-App Ads

5.1 Introduction

With the rapid adoption of smartphones, tablets, and mobile apps, more and more people use these personal digital devices for communication, entertainment, and professional activities. Among smartphone and tablet operating systems, Google's Android and Apple's iOS dominate the U.S. smartphone market by 52.5 and 34.3 percent, respectively (Graziano, 2012). The sweeping popularity of smartphones and tablets also affects the user population of children and adolescents. It has been reported that 25% of toddlers used their parents' smartphones in 2011 (Carmichael, 2011), and 23% of children and teens between the ages of 12 and 17 owned their own smartphones in 2012 (EnterpriseAppsTech, 2012). Meanwhile, more and more apps that are specifically designed for children appear on smart devices. As a result, parents start worrying about the content appropriateness of mobile apps for their children.

In order to help parents choose age-appropriate mobile apps for their children, both Android and iOS apps have maturity ratings that are similar to the movie and video game industries. Such maturity ratings examine the existence and intensity of mature themes such as violence, offensive language, sexual content, and reference of drugs within each app. Android maturity rating policy contains four maturity-rating levels: "Everyone", "Low Maturity", "Medium Maturity", and "High Maturity", while iOS's policy provides four different maturity-rating levels based on the suitable age of audience: "4+", "9+", "12+", and "17+". Higher maturity levels represent apps containing more objectionable contents.

These maturity rating policies on Android and iOS platforms may prevent children from being exposed to the inappropriate contents. However, neither mobile platforms nor advertising networks apply these maturity policies to restrict the contents of in-app advertisements. As a result, children may still be able to view high maturity contents from in-app advertisements within those apps rated with low maturity. For example, as a *4+* app on iOS platform, *Angry Birds* should "*contain no objectionable materials*" (Apple, 2012b). However, it allows a full-screen advertisement (Fig.13) with bloody scenes from a high maturity app appearing inside the app (the source of this bloody scene comes from the *9+* app *Blood Brothers* containing "*Infrequent/Mild Cartoon or Fantasy Violence*"). Further, *Angry Birds* also allows sexual banner advertisements shown on the up-right corner of the screen (Fig.14). As pointed out by the Washington Post, "*there have been complaints that violent and sexual ads pop up in some apps aimed at children*" (Rasmussen, 2011). However, according to our best knowledge, no systematic research has been conducted to analyze the content appropriateness of in-app advertisements for children's protection. This work is designed to bridge the gap in the literature. We aim to contribute the following:

- This is the first study to examine the risks of in-app ads on mobile platforms from the perspective of children's online safety. Previous work rarely considers children as a special group of mobile users. Thus there is no study measuring the content appropriateness of the in-app advertisements. This research aims to fill in this gap.

- We aim to examine how the advertising networks regulate and monitor the content appropriateness on their advertisements. Our preliminary findings suggest that none of the advertising networks integrates the parental control settings from mobile devices to deliver the age-appropriate in-app advertisements to end-users.

- We propose methods to calculate ads' maturity levels, and uncover the age appropriateness for different types of in-app advertisements.

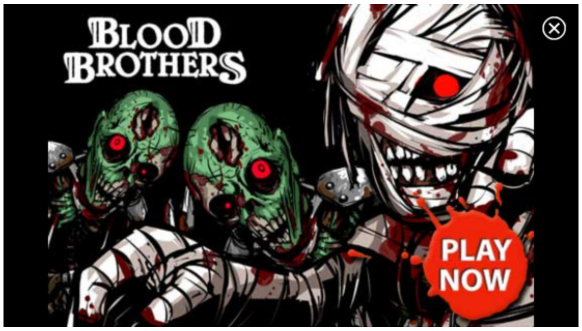


Figure 13. Sample full-screen ad in “Angry Birds”.

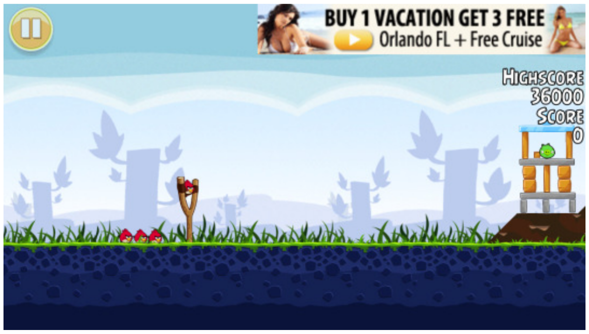
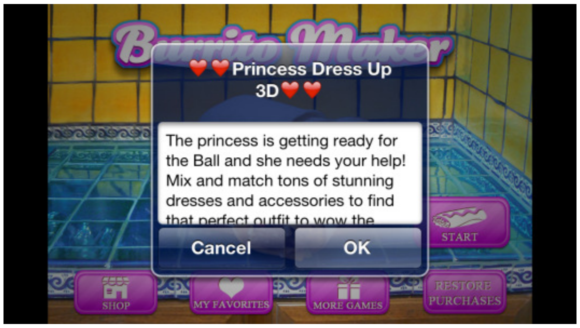


Figure 14. Sample banner ad in “Angry Birds”.



❤❤Princess Dress Up 3D❤❤

The princess is getting ready for the Ball and she needs your help! Mix and match tons of stunning dresses and accessories to find that perfect outfit to wow the

Figure 15. Sample pop-up ad in “Burrito Maker”.

This paper is structured as follows: we first review the relevant works, followed by proposing our research questions and describing the research methodology. Then we present our experimental design and findings. We summarize our findings in the discussion section and then conclude this work. (In this paper, “advertisement” and “ad” may be further used interchangeably).

5.2 Research Questions

As discussed earlier, we aim to examine the content appropriateness of the in-app advertisements on mobile platforms for children's protection. Our first research question is to ascertain:

1. Is the placement of in-app advertisements a common practice in free apps designed for kids?

In addition, we seek to find out whether the advertising networks regulate and monitor objectionable contents on the in-app advertisements. Therefore, the second research question is raised as:

2. What types of objectionable contents on the in-app advertisements are regulated and monitored by the advertising networks?

There are different types of in-app advertisements available on mobile platforms, such as full-screen ads (Fig. 13), banner ads (Fig. 14), pop-up ads (Fig. 15). The visual effects vary among different types of in-app advertisements. For example, full-screen ads are visually more significant than banner and pop-up ads. If the full-screen ads contain more severe age-inappropriate contents, they probably will be more harmful to children. In addition, the in-app advertisements may also have different targets. For example, there are advertisements promoting other mobile apps that are linked to external websites. Given the difference in visual effects for different types of in-app advertisement, our last research question is to examine:

3. How does age appropriateness vary among different types of in-app advertisements?

5.3 Methodology

In order to answer our research questions, we first collected the in-app advertisements from both Android and iOS platforms. Since this study was for children's protection, we focused on those apps which were specifically developed for children (further denoted as "kid-specific apps"), and conducted content analysis on their in-app advertisements.

5.3.1 Kid-specific Applications

To select kid-specific apps, we searched "app for kids" and "app for children" on both Android and iOS app stores. In order to compare the content appropriateness of the in-app advertisements on these two platforms, we focused on the intersection of the result sets and downloaded the free apps that were available on both platforms.

To confirm the intended audience of the kid-specific apps, we adopted FTC's approach to search keywords on the app descriptions (FTC, 2012), such as "infant", "toddler", "child", "kid", "preschool", "elementary school", "parent", "teacher", and "family". Once acquiring the general age groups of apps' intended audiences, we follow the FTC's approach (FTC, 2012) to search keywords such as "age", "year old", and "grade" to specify the age ranges (e.g. 3~8 years old) of the apps' intended audiences. With the general age groups and specific age ranges of apps' intended audiences, we could select the apps specifically developed for kids.

5.3.2 Advertising Networks Used by Applications

After downloading the kid-specific applications, we identified the advertising-supported apps by applying static analysis ("Static program analysis," 2013) to locate the libraries of the advertising networks from these apps.

Android apps were packaged as *.apk* files. After decompiling, each app folder contained all the resources used in the app, as well as the disassembled codes (with all function names and the parameters). By searching through the disassembled codes, especially the *AndroidManifest.xml* file, we found the libraries of the advertising networks used in the apps.

Similarly, iOS apps were packaged as *.ipa* files. Those files could be automatically decompiled once installed on iOS devices. We then utilized the device manger to download the decompiled folders from the devices, and searched for the libraries of the advertising networks in those folders.

5.3.3 In-app Advertisements on Mobile Platforms

Once advertising-supported apps were selected, we could collect the advertisements from those apps. On the Android platform, we utilized dynamic analysis ("Dynamic program analysis," 2013) to collect the in-app advertisements. Android is an open source platform, and it provides various debugging tools to test the functionalities of the apps. The app packages downloaded from Google Play app store were installed by *Android Debug Bridge (ADB)* (Google). To ensure collecting most of the advertisements shown in apps, we manually opened every app and clicked on the in-app advertisements to trigger the redirect events. Android logging tool *logcat* (Google) automatically recorded the URLs of the advertisements. On iOS platform, we manually recorded the types and the URLs of the in-app advertisements.

5.3.4 Content Analysis on In-app Advertisements

After collecting the in-app advertisements, we could analyze their content appropriateness. Firstly, the in-app advertisements were categorized based on their targets (i.e., the contents promoted by the advertisements). There were three types of ad targets: 1) iOS apps; 2) Android apps; 3) external websites. The maturity levels of the advertisements were equivalent to the maturity levels of their targets. For instance, if an advertisement promoting a 9+ iOS app appeared in a 4+ app, the maturity level of the advertisement was determined as 9+, for reflecting the maturity level of its target. We observe the maturity levels of the 98% ads' images reflected the maturity levels of their targets, while the maturity levels of the remaining 2% ads' images even exceeded the maturity levels of their targets to attract attention (this was an ad hoc, author-based assessment that is not replicable). Therefore, the maturity levels of ads' targets were used to present ads' maturity levels in this study. We proposed the following formula to calculate ads' maturity levels:

$$L_{ad} = \begin{cases} L_{iOS} & if\ ad \rightarrow iOS\ app \\ L_{Android} & if\ ad \rightarrow Android\ app \\ L_{ALM} & if\ ad \rightarrow web\ page \end{cases} \qquad (1)$$

For advertisements promoting Android and iOS apps, their maturity levels are deemed to be equivalent to the maturity levels of the target apps (Google and Apple provides a maturity rating for each app on its description page). In addition, for advertisements promoting external websites, we adopted the ALM (stands for "Automatic Label of Maturity Ratings for Mobile Apps") algorithm (Chen, Xu, et al., 2013) to calculate their maturity levels. The ALM algorithm was originally developed to infer mobile apps' maturity levels based on their descriptions, it provided detection accuracy over 96%. Here ALM was used to analyze the contents on external website pages and measured their maturity levels. The limitation of the ALM algorithm is that it can only determine content maturity based on text information.

5.4 Experiment

In this section, we described our experimental dataset, design and results.

5.4.1 Data Collection

By searching "app for kids" in both the Google Play and Apple iTune Stores, 405 free apps were found available on both Android and iOS platforms. APK Downloader *(Evozi, 2012)* was used to download apps on Android platform, while apps on iOS platform were downloaded manually.

In total, 3921 advertisements were collected from the apps. Most advertisements frequently appeared in those apps during April and May 2013. There were 1521 in-app advertisements collected from the Android platform, and 2447 in-app advertisements collected from the iOS platform.

5.4.2 Experiment 1: Kid-specific Applications

We crawled the app descriptions from both the iTunes app store and the Google Play app store, and found that apps' Android version and iOS version had the same descriptions. Furthermore, we searched through app descriptions based on three types of keywords: 1) children: "infant", "child", "kid", "preschool", and "elementary school"; 2) adults: "parent", "teacher", and "adult"; 3) family: "family". These three types of keywords represented the general age groups of the apps' intended audiences. The distribution of the apps based on the general age groups of intended audience was presented in Table 14.

Table 14. Intended audience—general age groups (n=405).

General age group	% of apps	Combined % of apps
Infant/toddler	7.9%	73.6%
Child	23.5%	
Kid	34.3%	
Preschool	7.4%	
Elementary school	0.5%	
Parent	10.6%	18.5%
Teacher	2.2%	
Adult	5.7%	

Family	12.8%	12.8%

Table 15. Intended audience—specific age ranges.

		Maximum recommended age						**% of apps with this min. age**
		0-2	**3-4**	**5-6**	**7-8**	**9-12**	**13+**	
Minimum recommended age	**0-2**	11	3	15	2	5	36	18%
	3-4	-	46	2	6	1	3	14%
	5-6	-	-	11	2	-	4	4%
	7-8	-	-	-	3	1	-	1%
	9-12	-	-	-	-	5	2	2%
	13+	-	-	-	-	-	1	0%
% of apps with this max. age		3%	12%	7%	3%	3%	11%	**n=405**

Some descriptions contained combination of keywords, and keywords representing adults were always mentioned with the keywords representing children. According to Table 14, 73.6 percent of the apps were specifically developed for children, and 7.9 percent of the apps—32 apps—were even designed for infants and toddlers.

Furthermore, we searched through the app descriptions based on the age keywords such as “age”, “year old”, and “grade”, to find out the suggested age ranges of the apps. After converting the grade levels to ages, we categorized the results and listed them in Table 15. Similarly, 60 out of 405 (14.8%) apps were designed only for toddler and infants (i.e., under 4 years old) to use, and approximately 18 percent of the apps indicated they were expecting newborns to use.

Interestingly, according to Apple’s maturity policy (Apple, 2012b), iOS apps are only for 4 years and older. However, according to our results and FTC’s report (FTC, 2012), Apple had approved many apps which were developed specifically for toddlers and infants, and placed them on the iTunes app store for users to download. Moreover, according to a 2011 report (Carmichael, 2011), there were already 25% of toddlers using their parents’ smartphones. Therefore, Apple should adjust its maturity rating policy to better monitor the contents of the apps and the in-app advertisements for toddlers and infants' protection.

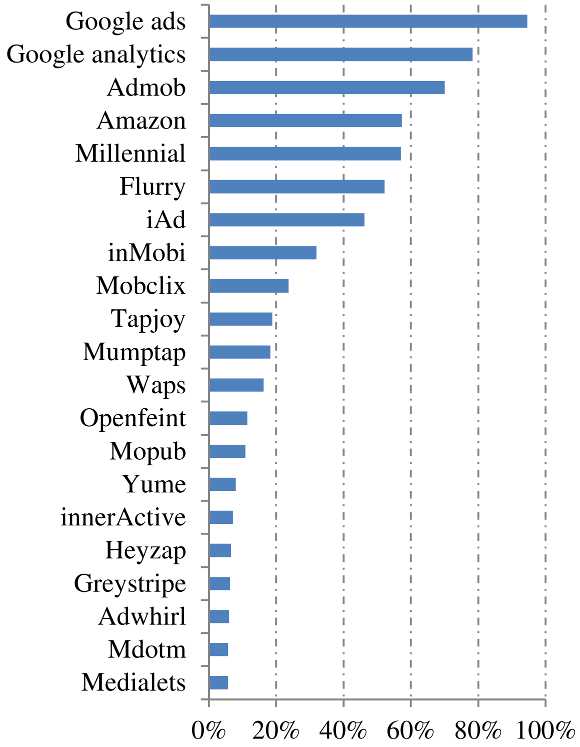


Figure 16. Distribution of advertising networks in Android apps.

5.4.3 Experiment 2: Advertising Networks

5.4.3.1 Usage of Advertising Networks in Applications

To further select the advertisement-supported apps, we searched the libraries of the advertising networks inside the 405 apps.

For Android apps, *android-apktool* (Google) was used to decompile the *.apk* packages (Android app packages). For iOS apps, *iTools* (Thinksky) was used to download the related decompiled folders of the installed *.ipa* packages (iOS app packages). Once the apps' decompiled files were ready, static analysis was applied to search the embedded libraries of the advertising networks. The distributions of the advertising networks used by Android apps and iOS apps were shown in Fig.16 and Fig. 17, respectively. We only listed the advertising networks used by at least 5 percent of the apps (i.e., used by at least 20 apps). *Google ads* advertising networks included Google's *AdSense* and *AdWords* advertising networks.

According to Fig. 16, 351 out of 405 Android apps (i.e., 86.7%) embedded advertising networks, and surprisingly, on average, each app embedded 6.97 advertising networks. One reason why Android apps embedded many advertising networks was that app developers allowed mediation options when they registered apps on the advertising networks. The mediation options were provided by most of the advertising networks; the advertising networks centrally controlled the integration of the advertisements from other advertising networks into their own ad pools. Once mediation had been enabled, other alliance advertising networks could also display their advertisements inside the apps. For example, Google *Admob* advertising network allowed mediation with 16 other advertising networks: *Adfonic*, *Drawbridge*, *Flurry*, *Hunt Mobile Ads, InMobi, Jumptap, LG U+AD, MdotM, Medialets*, *Millennial Media*, *MobFox*, *Mojiva*, *Nend*, *TapIt*, *i-mobile*, and *iAd.*

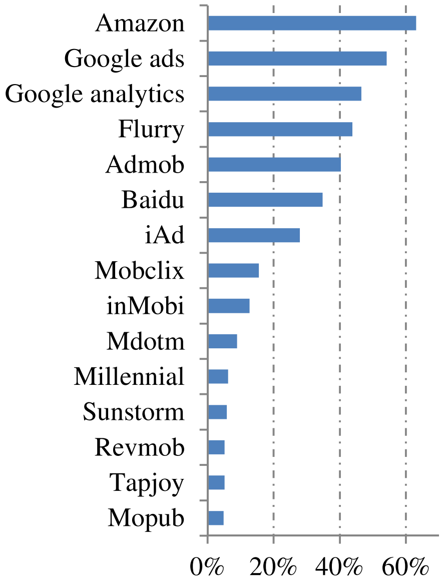


Figure 17. Distribution of advertising networks in iOS apps.

As shown in Fig.16, the distribution of the advertising networks used by the kid-specific Android apps were similar to the distribution of the advertising networks used by the general Android apps (Grace et al., 2012a). We noticed that the top three advertising networks used by Android apps were all from Google. Therefore, Android developers had strongly skewed towards Google advertising networks.

According to Fig.17, the distribution of the advertising networks used by iOS apps was slightly different from the distribution of the advertising networks used by Android apps. Firstly, only 290 out of 405 iOS apps (i.e., 71.6%) embedded advertising networks, and the average number of advertising networks embedded in apps was 4.13, lower than that on Android platform. Secondly, the selection of the advertising networks on the iOS platform was more diverse. There was no dominating advertising network on iOS platform.

Since the advertising networks were widely used on both platforms, and most of the kid-specific apps embedded many different advertising networks, the contents of advertisements could be frequently viewed by children. Therefore, if the in-app advertisements contain objectionable contents, children will have a large chance to be exposed to those inappropriate contents.

5.4.3.2 Advertising Networks' Policies Regarding Content Appropriateness

We further examined the policies of the advertising networks to check whether they considered and monitored the content appropriateness of the in-app ads. We analyzed the *terms of use* of the top 50 advertising networks; however, we were limited to only 39 ad networks. Results of other advertising networks were missing either because the *terms of use* information was not available online, or was written in foreign languages. We considered four types of objectionable contents: 1) violence and horror themes; 2) sexual and nudity contents; 3) offensive and hateful language; 4) references to gambling, alcohol, tobacco, and drugs. These types of contents were considered harmful for children on both Android and iOS platforms (Apple, 2012b; Google, 2012). The results of policy analysis were summarized in Table 16.

Table 16. Forbidden contents at top 39 advertising networks.

Ad network	**Violence, horror**	**Sexual, nudity**	**Offensive language**	**Gambling, alcohol, tobacco, drugs**
Admob	X	X	X	X
Google ads		X		X
Flurry	X	X	X	X
Google analytics		X	X	X
Millennial media				
Mobclix				
Adwhirl	X	X	X	X
iAd				
YuMe				
Mobfox				
Komli mobile	X	X	X	X
inMobi	X	X	X	X
Wooboo	X	X	X	X
AdMarvel				
Smaato	X	X	X	X
Airpush	X	X	X	X

Ad network	**Violence, horror**	**Sexual, nudity**	**Offensive language**	**Gambling, alcohol, tobacco, drugs**
MdotM				
Vdopia	X	X	X	
Wiyun		X		X
Adhubs				
Madhouse		X		X
Pontiflex		X	X	X
innerActive	X	X		X
Mocean mobile				
Casee				
Greystripe				
Adform				
Guohead	X	X	X	X
Domob	X	X	X	X
Tapjoy	X	X	X	X
Adchina	X	X	X	X
Jumptap				
Medialets	X	X	X	
Waps		X		X
Vpon	X	X	X	X
Iconosys				
Smartadserver				
Airad				
Mopub	X	X		X
iconosys				
Total	17(43.6%)	23(59.0%)	17(43.6%)	21(53.8%)

According to Table 16, 41 percent of the advertising networks did not regulate any types of inappropriate contents in their advertisements. They either did not provide specific remarks regarding the content appropriateness (e.g., *iAd*), or passed the responsibility to ad designers and end-users. For instance, *Mobfox* provided the following statement in the *terms of use*:

MobFox may have no control over any mobile websites or resources which are provided by companies or persons other than MobFox.

And *Mobclix* stated:

Our websites target an audience that is over the age of 18 and some content may not be appropriate for all ages. We recommend that minors over the age of 13 ask their parents and/or legal guardians for permission before sending any information about themselves to anyone over the Internet.

Similarly, *YuMe* stated:

The YuMe Services are not for persons under the age of 13. If you are under 13 years of age, please visit another site – there are lots of other great web sites for you.

Moreover, the advertising network *Jumptap* even allowed adult contents (i.e., sexual and nudity contents) to appear on their advertisements.

Among the remaining advertising networks (59%), all of them forbade pornography and adult contents to appear on their ads. For other types of inappropriate contents such as violence, offensive language, and gambling, fewer advertising networks forbade them to appear on the advertisements.

Although some advertising networks monitored ad contents and provided setting options on their dashboards for app developers to manage the categories of the expecting advertisements, they failed to provide fine-grained maturity levels for developers to filter out inappropriate contents. For example, *Admob* advertising network stated that they reviewed ads' contents and classified ads into two categories: "all ages" and "adult". App developers could filter out ads in the "adult" category. However, since *Admob* did not classify the ads to more detailed categories, such as "4+" and "9+", infants and toddlers still could be exposed to inappropriate contents. Other advertising networks only provided vague statements regarding how they monitor ad contents. For example, *Adform* stated that they "*monitor unsafe content*".

Since many advertising networks did not consider the content appropriateness of their advertisements, and provided no fine-grained opt-out options for app developers to filter out inappropriate ads, those inappropriate contents in advertisements were delivered to mobile devices and eventually reached children.

5.4.4 Experiment 3: Age Appropriateness of In-app Advertisements

In this experiment, we aim to analyze the content appropriateness of the in-app advertisements. We notice that the advertising networks provided targeting options to deliver advertisements to different groups of users. In order to collect an un-biased sample of the in-app advertisements, we first analyzed the targeting options provided by the advertising networks.

5.4.4.1 Advertising Networks' Targeting Options

The advertisement targeting options from the top 39 advertising networks were analyzed and the results were summarized in Fig. 18.

According to Fig.18, the advertising networks delivered their advertisements by users' demographic information, app category, and device information/usage. Users' demographic information included gender, age, income level, education level, language, and relationship status (e.g., single, married). The advertising networks could deliver advertisements by users' demographic information if it was available on users' mobile devices. Some of them also allowed to deliver ads to audiences by certain age ranges, such as 18~24, 25~34, 35~44, 45~54, 55~64, and 65+. Currently there were no rules designed to deliver ads to an audience less than 18 years old. In addition, the advertising networks could also deliver ads to certain categories of apps or a specific app. Furthermore, they could deliver ads by device information: day parts (e.g., weekend 10:00am~10:00pm), location, Internet connection (e.g., wifi or 3G), carrier (e.g., AT&T, Verizon), device (e.g., iPhone 4GS), and user events/timing. User events/timing included the activities users conducted on devices, such as the keywords users searched on devices, the time users spent on apps, users' purchasing and transaction history, and etc.

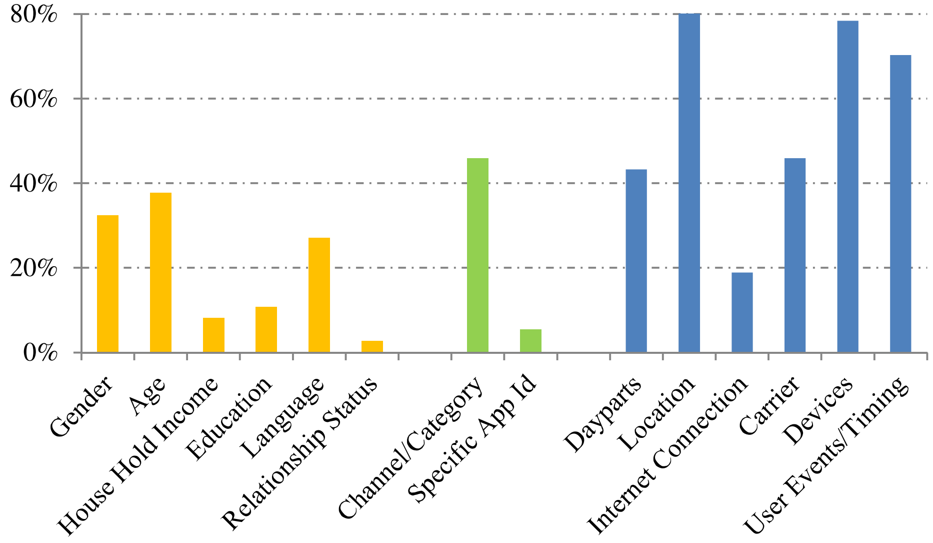


Figure 18. Targeting options provided by top 39 advertising networks.

Table 17. Apps' features on Android and iOS platforms.

	Android	**iOS**
Total	405	
With no ads	36	114
Not compatible with device	17	0
Cannot use	1	1
With ads	351	290
With (→app) ads	159	217
With (→website page) ads	157	66
With banner ads	172	203
With full-screen ads	89	105
With pop-up ads	23	18

Table 18. Features of the in-app ads on Android platform.

Ad types	**# of ads**	**# of ads:** **ad maturity > app maturity**	
Total	1521	546	35.9%
(→App) ad	645	295	**45.7%**
(→Website page) ad	876	251	28.6%

According to Fig.18, the advertising networks delivered their advertisements primarily by location, device, demographic information, and etc. To eliminate the influence of those factors and collect an un-biased sample of advertisements, the apps were installed on 4 different devices. The Android apps were installed on a Nexus S device (system version is 2.3.6), while iOS apps were installed on an iPhone 5 device, an iPhone 4 device, and an iPod touch 3 device. 2 different accounts were used to install the apps, one was a male account with age range 65+, the other was a female account with age range 18~24. We did not type in any other information into the accounts, and the devices were all reformatted before use. Because only users over 18 years old can set up mobile accounts, children have to use their parents' devices. Therefore, the results of this experiment are valid for testing the content children receive on mobile ads.

5.4.4.2 Content Analysis on In-app Advertisements

According to Table 17, three types of apps were excluded from the dataset: 1) apps without advertisements; 2) apps not compatible with our devices; 3) apps that could not be used in the location conducting the experiment. In total, there were 351 Android apps and 290 iOS apps in the dataset. "(→app) ads" denoted the ads promoting Android apps, and "(→website page) ads" represented the ads promoting external websites. In addition, only three types of in-app advertisements were observed on both Android and iOS platforms: *banner ads*, *full-screen ads*, and *pop-up ads*. Table 17 showed the numbers of apps with each type of in-app advertisements. The advertisements pre-integrated with the apps were not considered, for those were not dynamically pushed by the advertising networks, and app developers should have integrated the maturity levels of those advertisements into apps' maturity levels. In our experiment, none of the pre-integrated ads is observed containing objectionable content. Therefore, the pre-integrated ads should be fine.

In total, 1521 in-app advertisements were collected on Android platforms, and 2447 in-app advertisements were collected on iOS platforms. The maturity levels of the advertisements were calculated and compared with the host apps' maturity levels. The features of the in-app advertisements on Android and iOS platform were presented in Table 18 and Table 19, respectively.

According to Table 18, on Android platform, there were 645 in-app advertisements promoting Android apps, and 876 in-app advertisements promoting external websites. We adopted (1) to calculate ads' maturity levels. Furthermore, a *t-test* was performed to measure the distinction between ads' maturity levels and their host apps' maturity levels. *t-value* was used to measure the distinction, and *p-value* was used to measure the significance of the results. 45.7 percent of the ads promoting Android apps exceeded their host apps' maturity levels (t=13.97, p<0.001, within 98 apps), while 28.6 percent of the ads promoting external websites exceeded the host apps' maturity levels (t=8.73, p<0.001, within 65 apps). The ads promoting Android apps also had higher chance to exceed the host apps' maturity levels than the ads promoting external websites (t=5.20, p<0.001, within 89 apps). One explanation was that apps were developed by third party individuals and organizations, while websites were usually held by reputable corporations. Therefore, to attract attention, there were a larger percent of high maturity apps than websites.

Table 19. Features of the in-app ads on iOS platform.

Ad types	**# of ads**	**# of ads: ad maturity > app maturity**	
Total	2447	953	38.9%
(→App) ad	2251	919	**40.8%**
(→Website page) ad	196	34	17.3%
Banner ads	818	275	33.6%
Full-screen ads	1609	678	**42.1%**
Pop-up ads	20	0	0%

Since *logcat* was used on Android to record the in-app advertisements' urls, it failed to record whether the advertisements were banner ads, full-screen ads, or pop-up ads. Therefore, their maturity levels were not compared across these ad types on the Android platform.

Similarly, according to Table 19, 40.8 percent of the ads promoting iOS apps exceeded their host apps' maturity levels (t=17.42, p<0.001, within 151 apps), while 17.3 percent of the ads promoting external websites exceeded the host apps' maturity levels (t=4.51, p<0.001, within 21 apps). The ads promoting iOS apps also had higher chance to exceed the host apps' maturity levels than the ads promoting external websites (t=7.56, p<0.001, within 8 apps).

There were 818 banner ads, 1609 full-screen ads, and only 20 pop-up ads collected on iOS platform. The banner ads normally appeared for most of the time while apps were open on the foreground, and usually stayed for 20~30 seconds before refresh (developers could change the settings on display time, interval and refreshing rate). The full-screen ads normally appeared when users triggered certain events inside the apps. The events could be "played the app for over 1 minute", "finished a level", "restarted a level", and etc. The pop-up ads normally appeared when apps started, and never displayed again unless the apps are re-opened. Both full-screen ads and pop-up ads required immediate response.

In total, 33.6 percent of the banner ads exceeded their host apps' maturity levels (t=14.53, p<0.001, within 130 apps), while 42.1 percent of the full-screen ads exceeded their host apps' maturity levels (t=13.46, p<0.001, within 77 apps). The full-screen ads also had a higher chance to exceed the host apps' maturity levels than the banner ads (t=2.87, p=0.003, within 57 apps). For the pop-up ads, none of them exceeded the host apps' maturity levels. Since the full-screen ads also had a more significant visual impression than other types of ads, the high maturity full-screen ads might be the most harmful to children's mental health. In contrast, the pop-up ads only contained text. And according to Table 19, their maturity levels were consistent with the host apps' maturity levels. Therefore, the pop-up ads were the least harmful to children.

From the results on both platforms, a large proportion of the in-app advertisements had higher maturity levels than the host apps. It was highly possible that children would be exposed to inappropriate contents while they are playing with those apps. The results also showed that the advertisements promoting other apps and the full-screen advertisements tended to appear on lower maturity apps. These results could help parents, mobile platforms, and advertising networks to improve their efforts to protect children using the apps.

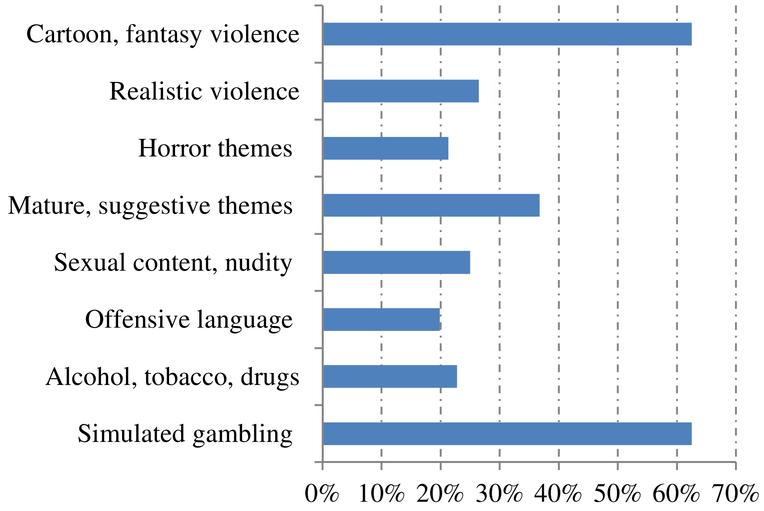


Figure 19. Types of objectionable contents in high maturity in-app advertisements on iOS platform. (n=136)

5.4.4.3 Types of Inappropriate Contents in Mobile Advertisements

Further, we investigated that the in-app advertisements contained higher maturity levels of objectionable contents than that of their host apps. Apple provided detailed reasons for each app's maturity rating on the iTunes app store if it was higher than 4+.For example:

Rated 9+ for the following:

- *Frequent/Intense Cartoon or Fantasy Violence*

Therefore, for the advertisements promoting iOS apps with maturity levels higher than the host apps, the reasons of their maturity levels were collected and the results were presented in Fig. 19.

According to Fig. 19, three types of objectionable contents mostly appeared in the high maturity in-app ads: 1) cartoon, fantasy violence; 2) simulated gambling; 3) mature/suggestive themes. We believe that the results could bring more insights to parents about what types of objectionable contents to expect in the advertisements exceeded the maturity levels of the host apps.

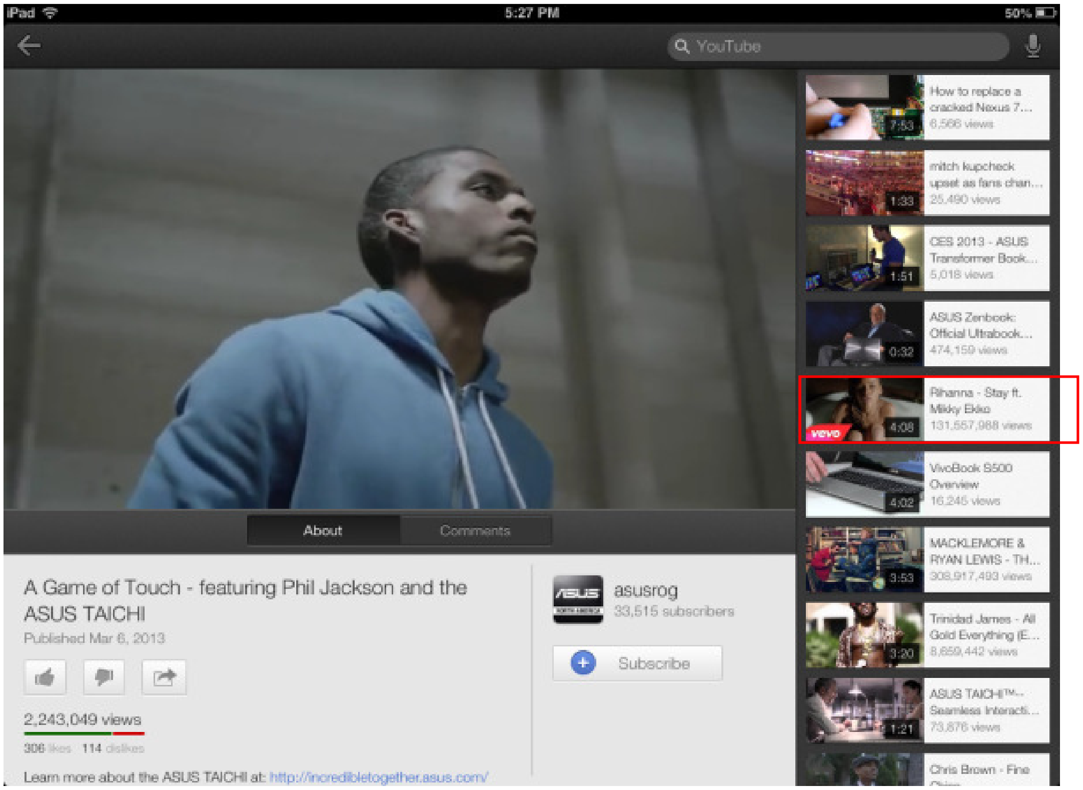
5.5 Discussion

As discussed earlier, there are many in-app advertisements that exceed their host apps' levels of maturity ratings. Given the fact that more and more toddlers and infants are actually using mobile devices, and that there are many apps specifically designed for them to use, the iOS platform should develop corresponding maturity rating policy to monitor contents for both apps and in-app advertisements in order to protect users under 4 year old.

Further, we find that regardless of the ad types and the ad targets, a large proportion of the ads exceed their hosts' maturity levels, especially the full-screen ads and the ads promoting other mobile apps. In addition, among those advertisements which exceed their hosts' maturity levels, most of them contain cartoon, fantasy violence, simulated gambling, and mature/suggestive themes. However, neither Android nor iOS has maturity policies to restrict the contents of in-app advertisements.

Our findings suggest that only 59 percent of the advertising networks regulate the content appropriateness of their advertisements, and few of them actually monitor ads' contents and provide filtering options on their dashboards for app developers to filter ads based on the content appropriateness. Besides, currently none of the advertising networks integrate the parental control settings on mobile devices to deliver the age-appropriate advertisements to end-users.

The surrounding contents of the advertisements on social media can also be harmful, and it is more difficult to monitor the user-generated contents on social media. For example, sometimes ads are embedded in social media, such as Youtube, Facebook, and Twitter. Social media recommend materials to users based on their viewing histories. Fig.20 is an example that although the advertisement contains no objectionable contents, the suggested material recommended by social media does. An advertisement shown inside a 4+ app on iOS platform has been redirected to a Youtube video. Among the suggested videos, there is one (Fig.21) with a sexual cover screen—a nude woman sitting in a tub.


iPad
5:27 PM
50%
YouTube
How to replace a cracked Nexus 7...
6,566 views
7:53
mitch kupchack upset as fans chan...
25,490 views
1:33
CES 2013 - ASUS Transformer Book...
5,018 views
1:51
ASUS Zenbook: Official Ultrabook...
474,159 views
0:32
Rihanna - Stay ft. Mikky Ekko
131,557,988 views
4:08
vevo
VivoBook S500 Overview
16,245 views
4:02
MACKLEMORE & RYAN LEWIS - TH...
308,917,493 views
3:53
Trinidad James - All Gold Everything (E...
8,659,442 views
3:20
ASUS TAICHI™ - Seamless Interacti...
73,876 views
1:21
Chris Brown - Fine
About
Comments
asusrog
33,515 subscribers
Subscribe

A Game of Touch - featuring Phil Jackson and the ASUS TAICHI

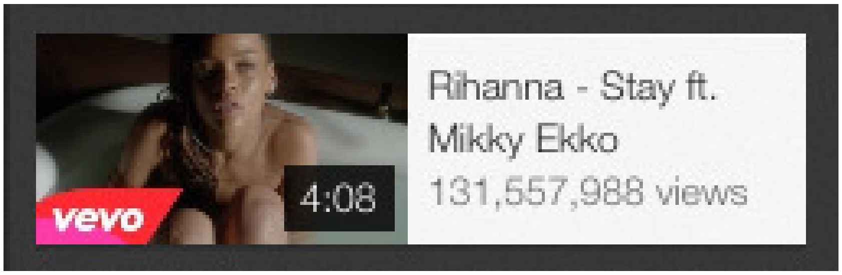
Published Mar 6, 2013

2,243,049 views

306 likes 114 dislikes

Learn more about the ASUS TAICHI at: http://incredibletogether.asus.com/

Figure 20. An advertisement linked to a Youtube video.



Rihanna - Stay ft.
Mikky Ekko
131,557,988 views

Figure 21. A suggested video with a sexual cover screen.

The challenges regarding the content appropriateness of the in-app advertisements cannot simply be tackled by one entity. Therefore, to better protect children, we suggest advertisement providers, advertising networks, app developers, and mobile platforms collaborate in developing policies and mechanisms to monitor and control the content appropriateness of the in-app advertisements.

5.6 Conclusion

Both Android and iOS allow app developers to link their free apps with advertising networks to increase revenue. Behavioral research on users' privacy practices suggests that mobile users would rather engage with in-app ads than pay for no-ad apps (MarketingPilgrim, 2013). As a result, mobile in-app advertising has become a common practice. However, little research has systematically uncovered the extent and severity of the inappropriate contents on the in-app advertisements. To bridge this gap, our study explores the content appropriateness of the in-app advertisements on both Android and iOS platforms.

Our research has several contributions. Firstly, the advertisement content regulations and the monitoring methods provided by the major advertising networks were practically examined. Only a small percentage of the advertising networks considered the content appropriateness of their advertisements, and provided opt-out options for app developers to filter out inappropriate advertisements. Secondly, experimental results showed that regardless of the advertisement types and the advertising targets, a large percentage of the in-app advertisements exceeded their host apps' maturity levels, especially the full-screen ads and the ads promoting other mobile apps. Therefore, both Android and iOS should develop corresponding maturity rating policies to restrict the content appropriateness of the in-app advertisements. Lastly, we point out that while viewing the contents of advertisements, users may also be affected by the objectionable user-generated contents on social media. We believe that our findings provide insights for parents, advertising networks, platform providers (e.g., Google or Apple) as well as for regulatory bodies and application developers to better collaborate in protecting children's online safety.

Chapter 6 Sexual Solicitation and Internet-Initiated Offline Encounters

6.1 Introduction

Online solicitation of minors is a violation of state and federal law. However, numerous children have experienced cyber-solicitations (referring to online sexual solicitations and Internet-initiated offline solicitations). Wolak et al [1] found that one in seven kids received at least one online sexual solicitation in 2006. Efforts such as detection of online sexual solicitations from youth mainly focus on online chat-rooms, instant messaging, and gaming devices, where cyber-solicitations mostly occur [1]. Nowadays, mobile devices have integrated all these communication functions and become the new and the major channel for online solicitations. Julie Miller, a consultant pediatrician said [2]:

> *"We have witnessed the rapid uptake of broadband, the increasing ownership of camera phones, many with video capability, the arrival of 3G bringing improved speeds and services to the mobile and making the mobile internet a reality, and the continued popularity of services—and not just chat-rooms—that can put children in touch with people that they do not know, such as instant messenger and online games."*

However, children and even parents are rarely aware of the potential risks of information disclosure through mobile devices. On the contrary, more and more young children become fans of mobile devices. It has been reported that 25% of toddlers used their parents' smartphones in 2011 (Carmichael, 2011), nearly 10% children get first mobile phone by the age 5 (Theguardian, 2013), and 48% of American children aged between 6 and 12 want Apple’s iPad for Christmas gifts (Olson, 2012).

There are two major types of behaviors on mobile devices that expose children to online sexual solicitation (Ybarra et al., 2007): posting personal information and interacting with online strangers. Unfortunately, there are no existing regulations guarding children’s personal information on mobile devices. The Children’s Online Privacy Protection Act (COPPA) details the requirements of a website operator to seek verifiable consent from a parent or guardian, and the responsibilities an operator has to protect children's online privacy including restrictions on the marketing to those under 13. However, it does not consider mobile devices. Personal information on mobile devices, such as locations, photos, and calendar, which can be easily retrieved but hard to get from other digital devices, is also not considered by COPPA. Furthermore, only 13% of the mobile apps have privacy policies posted on app developers’ websites (FTC, 2012) regarding the collection and distribution of users’ personal information. In general, regulations on disclosure of users’—especially children’s—personal information on mobile devices are lacking.

Sharing children’s personal information (CPI) on mobile devices to third-parties exposes children to cyber-solicitations; however, there is no existing policy regulating the collection and distribution of CPI on mobile devices, so this study seeks to fill the gap.

This study contributes the following:

- We use the contextual integrity theory [8] to develop quantitative measures of privacy risks on mobile apps.
- We propose an automatic system with user interface to assist parents in being aware of apps’ privacy risks, therefore, to make informed decisions when choosing apps for their children.

The result of this study explains the boundaries for collecting and distributing children's personal information on mobile platforms, which provides guidelines for policy makers to quantitatively measure mobile apps' privacy risk levels.

6.2 Methodology

Nissenbaum's theory of contextual integrity (Nissenbaum, 2004) argues that privacy concerns are not absolute but largely depend on the context. For example, revealing the content of personal belongings to strangers may not be acceptable in offices, but it is fine at airports. In a mobile environment, disclosing children's photos and locations to strangers is much more dangerous in outdoor environments than in classrooms.

In general, contextual integrity (Nissenbaum, 2004) conceptualizes privacy as:

> In a *context*, the flow of information of a certain type (*attributes*) about a *subject* (acting in a particular capacity/role) from a *sender* (possibly the subject, acting in a particular capacity/role) to a *recipient* (acting in a particular capacity/role) is governed by a particular *transmission principle*".

Therefore, contextual integrity is characterized by four elements: *contexts*, *actors*, *attributes*, and *transmission principles*. Context has been generally defined as "stimuli and phenomena that surround and, thus, exist in the environment external to the individual" (Mowday & Sutton, 1993). The privacy concerns for children to disclose CPI change dynamically when children use them in different context (e.g., location, time). However, context cannot help to measure an app's static privacy risk when parents shop on an app store and search apps for their children. Therefore, context is not used to measure an app's static privacy risks in this study. It will be left for future work.

6.2.1 Actors

In contextual integrity theory (Nissenbaum, 2009), information norms have three placeholders for actors: *subjects*, *senders*, and *recipients*. In a mobile environment, information subjects are mobile users. Information senders can be the mobile users, the applications, or the in-app advertisements. Mobile users can send their personal information out through browsers, SMS, instant messaging (IM)/chatting apps, social networking sites/apps, and etc. Applications granted with certain permissions (e.g., READ_CONTACTS and INTERNET permissions on Android/iOS) can fetch users' personal information from the mobile devices and send it out through the Internet. To better target the customers, in-app advertisements also fetch users' personal information from mobile devices and send it to the advertising networks.

Information recipients on mobile devices consist of other mobile users (i.e., friends, strangers), the advertising networks, and the analytic third parties. Friends/strangers receive users' personal information during the following circumstances: 1) using social networking sites/apps, 2) using chatting/dating apps, 3) exchanging SMS/MMS/IM, 4) competing/coordinating in apps (e.g., word with friends), and 5) playing massive-multiplayer online (MMO) games/apps. Assuming friends are generally benign while sharing CPI and interacting with online strangers is risky, detection of mobile-web predators is critically needed. There are many methods to detect online predators (Kontostathis, 2009). By applying those methods in a mobile environment, one can evaluate the trustworthiness of information recipients of CPI and effectively protect children from cyber-predators.

Besides friends/strangers, other parties also receive CPI (e.g., the advertising networks, and the analytic third parties). Advertising networks receive CPI from apps to provide targeting advertisements. Analytic third parties also receive CPI from apps for marketing purposes. The trustworthiness and privacy awareness of these parties can be measured by trusted third parties (e.g., Web of Trust (WOT)).

6.2.2 Attributes (Information Types)

Attributes circumscribe different types of information revealed in a particular context (Nissenbaum, 2009). In this study, children's trace information (CTI) on mobile devices is considered as attributes. CTI—such as location, contacts, calendars, photos, audios, videos—can be retrieved from mobile devices. Revealing the trace information exposes children's daily routines and contacts to strangers. Particularly for the location information, Barkhuus (Barkhuus, 2012) emphasizes that the risk level varies when disclosing the location information with different granularities. For example, it is less dangerous to give out information about children's resident city than the restaurant address where they just walked in. It is also less dangerous to disclose user-entered place names than the exact geo-referenced locations (based on latitude and longitude) or the community-named locations (Barkhuus, 2012) (e.g., commercial 'check-in' services such as Gowalla, foursquare and Facebook's check-in).

6.2.3 Transmission Principles

Transmission principle refers to "a constraint on the flow (distribution, dissemination, transmission) of information from party to party in a context" (Nissenbaum, 2009). Originally defined by Nissenbaum (Nissenbaum, 2009), re-framed in this study to the mobile world, transmission principles include instances such as *confidentiality*: information recipients are prohibited from sharing the information with others; *compulsion*: information subjects are compelled or mandated to reveal information; *need* (i.e., deserve, entitlement): information recipients need/deserve/entitled to know a particular kind of information.

An information flow is considered confidential when the information senders (i.e., mobile apps) provide privacy policies regarding the distribution of collected CPI. An information flow is considered compulsive when the information subjects have to disclose information (e.g., asked to login) or accept permission requests for personal information access before using the services/apps. An information flow is considered needed when the information recipients need certain types of information to provide corresponding services.

In general, in this study, privacy is measured as the dimensions as shown in Fig. 22.

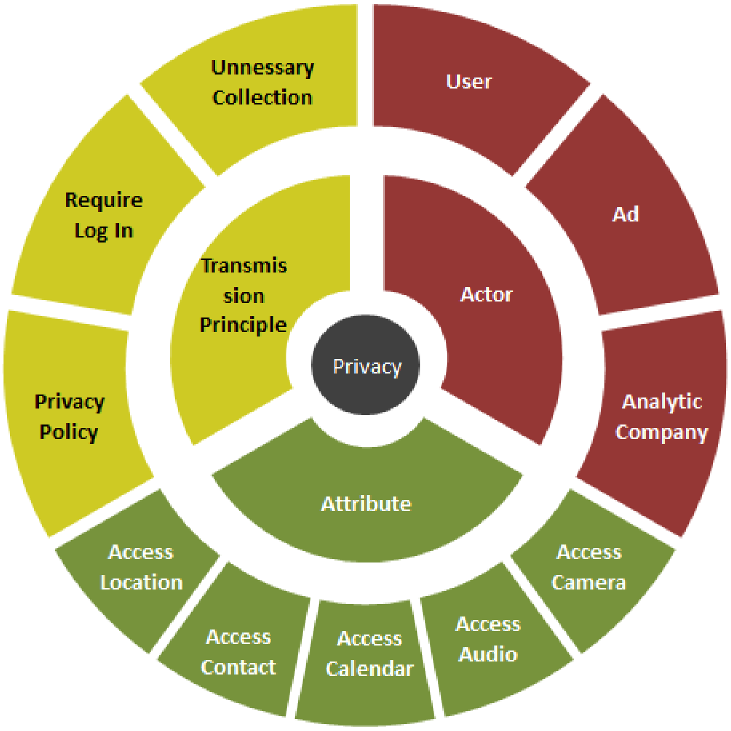


Figure 22. Mobile privacy measurement

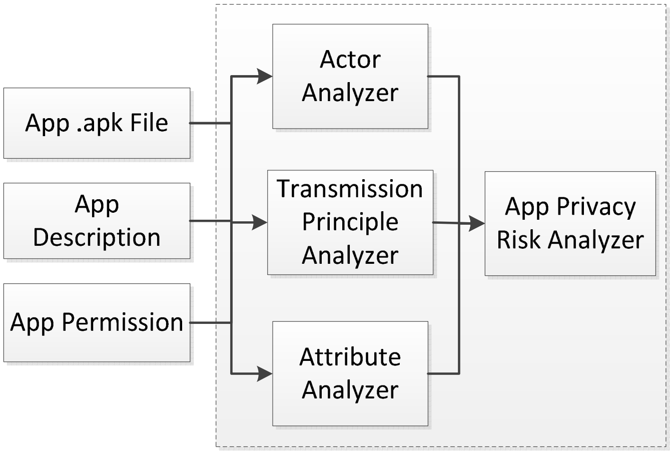


Figure 23. System design framework

6.3 System Design

Researchers have proposed several methods to measure mobile apps' security levels (e.g., for malware detection (Peng et al., 2012a)). Although it is discovered that users' personal information has been aggressively collected by different parities from mobile devices (Grace et al., 2012b), currently there is no quantitative approach to measure the privacy risk levels of mobile apps. In this study, we propose exploiting contextual integrity theory to quantitatively measure the privacy risks of mobile apps.

The framework is presented as an *App privacy risk analyzer*, which scores each app's privacy risk level. With it, parents can make informed decisions when shopping for apps for their children. The *App privacy risk analyzer* measures the privacy risk levels of mobile apps with three dimensions in contextual integrity (i.e., actors, attributes, transmission principles).

6.3.1 Actor Analyzer

As discussed above, actors (i.e., actor-recipients) includes other users and third parties. Most common third parties are ad networks and analytic parties.

Apps can disclose users' information to other users when it 1) embeds a visible "share buttons" (Android) allowing users to send out app contents (further referred as share functions), 2) allows users logging in with other web-ids, such as Facebook, Twitter, and Gmail account (further referred as login functions), 3) allows users to chat (further referred as chat functions), 4) allows users to discuss on certain topics (further referred as discussion functions). The chat and discussion functions are rarely observed in kids apps, so we focus detecting the share and login functions in this study.

Share functions (i.e., visible "share button" (Android) to send out app content) can be detected by searching for specific APIs inside apps, for there are standard ways to implement them. There are two standard ways to implement share-style functions in Android apps. The Sherlock action bar (JavaExperience) is used to implement share functions in old Android versions:

```
Intent shareIntent = new Intent(Intent.ACTION_SEND);
shareIntent.setType("text/plain");
startActivity(Intent.createChooser(shareIntent,"ShareVia"));
```

For Android 4.0 (API Level 14) and up (JaveExperience), Android's newly introduced action provider can be directly used to implement share functions:

```
ShareActionProvider mShare = xxx;
Intent shareIntent = new Intent(Intent.ACTION_SEND);
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.setType("text/plain");
mShare.setShareIntent(shareIntent);
```

Two implementations share some common features: 1) they both set *ACTION_SEND* intents, 2) they both set type as “text/plain”. The only difference is that the Sherlock action bar uses *createChooser* to start the intent, while in the newly introduced action provider, *ShareActionProvider* is used to start the intent. Therefore, if intents found in apps are set with action *ACTION_SEND* and type “text/plain”, and either *ShareActionProvider* or *createChooser* is used to start the intent, the apps must contain share functions.

Similarly, there are standard ways to implement login functions (i.e., logging in with other web-ids, such as Facebook, Twitter, and Gmail account). Take Facebook as an example, it provides specific instructions for developers to embed Facebook logins in Android apps (Facebook). Detecting those authentication functions can certainly discover whether or not an app allows logging with other web-ids. Other than apps’ decompiled codes, we also search simple keywords (e.g., “share”, “Facebook”, “Twitter”, “social network”) in app descriptions, to cover the case that 1) the apps’ codes are obfuscated, and 2) the share functions are not written inside the apps but on web pages.

Furthermore, we identify ad networks and analytic third parties embedded in apps by searching through the *AndroidManifest.xml* file to locate their libraries. Grace et al (Grace et al., 2012b) list the top 50 advertising libraries on mobile platforms. More advertising and analytic third party libraries can be found in (Chen, Zhu, et al., 2013). Once we identify whether or not an app allows other users or third parties to access users’ information, we can quantify the actor dimension in the app privacy measurement. Assume app a allows sharing information to other users, denote $u_a = 1$, otherwise, $u_a = 0$. Assume app a embeds third party libraries, denote $tp_a = 1$, otherwise, $tp_a = 0$. Therefore, the actor risk score of app a can be represented as: $Actor_a = (w_u * u_a + w_{tp} * tp_a)/(w_u + w_{tp})$ where $w_u > w_{tp}$. w_u is the weight of u_a, w_{tp} is the weight of tp_a. w_u is higher than w_{tp} because over exposure to other users more easily cause solicitation than over exposure to third parties.

6.3.2 Attributes (Information Types) Analyzer

We study five types of critical personal information as attributes in this study: location, contact, calendar, audio, and camera. Besides these five types of attributes, iOS (Costello, 2014) also considers "Reminder" in its privacy settings. Reminder can be accessed by *read_calendar* permission on the Android platform, so it is already included in the calendar detection. App permissions are used to determine whether certain types of personal information are collected. That is, the *access_coarse_location* and *access_fine_location* permissions are used to detect whether users' location information is collected, the *read_contact* permission is used to detect whether users' contact information is collected , the *read_calendar* permission is used to detect whether users' calendar information is collected, the *record_audio* permission is used to detect whether users' audio information is collected, and the *camera* permission is used to detect whether users' photo/video information is collected. Assume app a collect p types of personal information out of the 5 types of critic personal information, the attribute risk score of app $a$is: $Attribute_a = p/5$.

6.3.3 Transmission Principle Analyzer

As discussed above, the transmission principle includes three dimensions: *confidentiality*, *compulsion*, and *need.* Confidentiality indicates that information recipients are prohibited from sharing the information with others. It can be directly derived from apps' privacy policies. We scan apps' descriptions and search for their privacy policies. For apps with privacy policies, we further highlight the sentences containing related phrases—"children", "Children Online Privacy Protection Act (COPPA) (FTC)", "advertisement", "ad network", "third party", "share"—in their privacy policies.

Compulsion indicates that information subjects are compelled or mandated to reveal information. On mobile platforms, we check whether apps require users to register or log in before providing any services. On Android, register and log in functions normally call the function *OnEditorActionListener*, which allow users to type in using keyboard. We reconstruct Android apps' activities as trees, and functions are listed as tree note attributes. We then locate the tree nodes with the attribute *OnEditorActionListener*. If the identified nodes are within ε depth ($\varepsilon \rightarrow 0$) to the root (entry activity, can be found in the *AndroidManifest.xml* file), the app is annotated as "forcing users to register or log in".

Need (i.e., deserve, entitlement) indicates that information recipients need/deserve/entitled to know a particular kind of information. To determine whether an app needs a particular kind of information for service, we use Whyper (Pandita et al., 2013) to detect whether the app mentions the need for the information in its description. Whyper uses Natural Language Processing (NLP) techniques to examine whether the application description provides any indication for why the application needs a permission. Since Whyper only detects the needs for contact, calendar, and audio, we have to manually analyze the Android API to build the semantic graphs for location and camera permissions. The remaining components of Whyper (Whyper) are adaptable to detect needs of all 5 types of the critical permissions. Assume an app requests personal information which is not needed for service, the app is annotated as "collect unneeded personal information".

Assume app a has no privacy policy, denote $pr_a = 1$, otherwise, $pr_a = 0$. Assume app a forces users to register or log in, denote $f_a = 1$, otherwise, $f_a = 0$. Assume app a collects unnecessary PI, denote $un_a = 1$, otherwise, $un_a = 0$. Therefore, the transmission risk score of app a is: $TP_a = \sqrt{\frac{{pr_a}^2 + {f_a}^2 + {un_a}^2}{3}}$.

6.3.4 App Overall Privacy Score

In general, we measure a mobile apps' privacy risks by three values: actor risk score, attribute risk score, and transmission risk score. Therefore, the privacy risk of app a can be calculated as the quadratic mean of the three dimensions:

$$P_a = \sqrt{\frac{{Actor_a}^2 + {Attribute_a}^2 + {TP_a}^2}{3}}.$$

The reason we choose quadratic mean is because the calculated result skewed to the larger number. The general expression for power mean is as follows:

$$\bar{x}(m) = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^m\right)^{\frac{1}{m}}$$

It is defined for a set of n positive numbers x_i . By choosing different values for the parameter *m*, the following types of means are obtained:

$$\begin{array}{lll}
ForActor_a = 0.5, & Attribute_a = 0.7, & TP_a = 0.3 \\
m \to \infty, & max(x_i) & \\
m = 2, & Quadratic\ mean\ (QM), & P_a = 0.53 \\
m = 1, & Arithmetic\ mean\ (AM), & P_a = 0.5 \\
m \to 0, & Geometric\ mean\ (GM), & P_a = 0.47 \\
m = -1, & Harmonic\ mean\ (HM), & P_a = 0.44 \\
m \to -\infty, & \min(x_i) &
\end{array}$$

Because we want the overall privacy score P_a reflect the most risky dimension (i.e., the dimension with largest number), only QM satisfy the requirement. Therefore, QM is chosen to calculate the apps' overall privacy scores.

6.4 Application Study

This section reports an application study using our privacy analysis framework to analyze app privacy risks. For app privacy risk analysis, we randomly select 30 apps from Google Play store's *Family* category (later refer as family dataset). Apps are placed into the *Family* category based on the developers' best judgments. Almost all the apps in *Family* category are designed for kids to play. We find that the average privacy risk for *Family* apps is 0.16. We also present a user interface helping parents to visualize apps' privacy risks.

6.4.1 Findings

6.4.1.1 Actor Analyzer

According to Table 20, 11 of the 30 family apps disclose information to other users. 9 of the 11 apps allow logging in with a Facebook account; 5 of the 11 apps allow sharing information on Facebook and Twitter through a browser; 4 of the 11 apps allow logging in with a Facebook account, and sharing information on Facebook and Twitter through a browser; and 1 of the 11 apps allows sharing information through email and SMS.

Table 20. Actor--other users

Apps (Family Category)	Share via email, SMS	Share to Facebook, Twitter	Log in with Facebook
Dress Up - Doll Salon, Monsterama Park, Pet Food Train, Ice Pops Maker Salon, Libii Hospital			√
Paint Joy - Color & Draw		√	
Clumsy Doctor, 3D Model Dress Up Girl Game, Cookie Deluxe - Cooking Games, Little Girl Salon		√	√
Cute Girl Summer Dress Up	√		

25 of the 30 *Family* apps embed ad networks and each of them embeds 1.5 ad networks on average. According to Fig. 24, Google ads is the most popular ad network embedded in the Android kids apps, and Google analytic is the most popular analytic third party embedded in the Android kids apps.

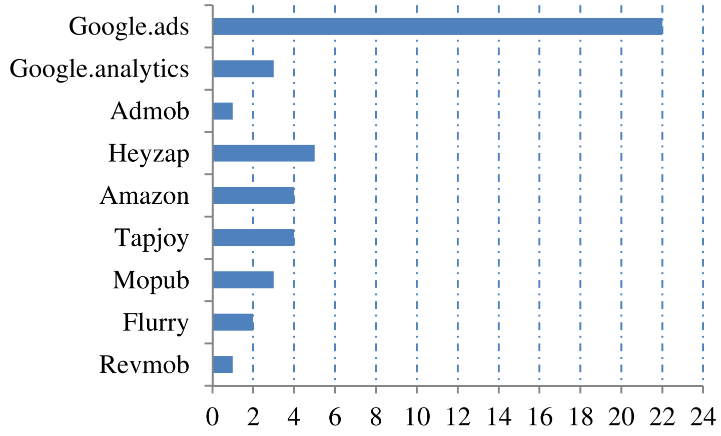


Figure 24. Actor--third parties

In total, 27 of the 30 *Family* apps disclose information to either other users or third parties.

6.4.1.2 Attribute Analyzer

According to Table 21, 12 of the 30 *Family* apps collect sensitive personal information from users. 8 of the 12 apps collect users' location information, 1 of the 12 apps records users' audio, 4 of the 12 apps access users' photo/video gallery and are able to take photos/videos.

Table 21. Transmission Principle— Sensitive information collection (Family)

Apps (Family)	**L**	**A**	**Cam**
Fashion Girls Nail Salon		√	
Make Jelly Candy, Sally Spa Salon-Fashion Games, Chocolate Maker, Draw Robots Transformers, Cute Girl Summer Dress Up, Ice Candy Maker, Cookie Deluxe - Cooking Games	√		

Dress Up - Doll Salon, Ice Pops Maker Salon, Little Girl Salon			√
3D Model Dress Up Girl Game	√		√

6.4.1.3 Transmission Principle Analyzer

To determine whether the apps violate private information transmission principles, we first check for confidentiality. 17 of the 30 *Family* apps have privacy policies.

For the privacy policies provided by the *Family* apps, 14 of the 17 policies mention the apps are particularly designed for kids, 5 of the 17 policies mention the apps comply with the Children Online Privacy Protection Act (COPPA), 1 of the 17 policies mentions its embedded ad networks complying with COPPA, 1 of the 17 policies mentions its embedded third parties (not ad networks) complying with COPPA, and 5 of the 17 policies mention they only send users' information to third parties when business transfers, in partnership, and under legal requirements.

According to Table 22, 3 of the 17 *Family* apps with privacy policies are ad free, 2 of the 17 apps with privacy policies are third-party free, and 2 of the 17 apps with privacy policies do not send information to third parties. For ad networks and other third parties, if the policies do not mention that they complying with COPPA, users need to check the privacy policy of each third party for their information security.

Table 22. Transmission Principle—Confidentiality (Family)

Apps (Family)	**Kid***	**App**	**Ad**	**TP**	**Send**
Make Jelly Candy, Ice Candy Maker	√	√			
Webkinz™	√	√			√
Dress Up - Doll Salon, Monsterama Park	√				√
Pet Food Train, Ice Pops Maker Salon, Libii Hospital, Little Girl Salon	√				
Toddler Cars: ABCs & Numbers, Animals for Toddlers	√		/	/	
Cookie Maker, Chocolate Maker					/
Super Chief Cook -Cooking game	blank privacy page				
Princess Crush	√		√		√
Puppy Doctor	√	√	/		
Clumsy Doctor	√	√		√	
Cookie Deluxe - Cooking Games					√

*Kid: mention the app is directed towards children under 13; App: the app complies with Children Online Privacy Protection Act (COPPA); Ad: Ad networks comply with COPPA; TP: Third party comply with COPPA; Send: the app send kids' personal information to third party only when business transfers, in partnership, and under legal requirements; Slash: no ads, no third party included, or not send information to third party.

To determine whether the apps violate private information transmission principles, we secondly check for compulsion. That is, whether apps force users to reveal information before providing the service. 1 of the 30 *Family* apps (i.e., the app *Webkinz*™) requires user to log in, and 1 of the 30 *Family* apps (i.e., the app *Fruit Juice Maker*) writes in the privacy policy that it requires user to log in but actually the app does not.

To determine whether the apps violate private information transmission principles, we lastly check for necessity. That is, whether the apps retrieve/store sensitive information from users, which is unnecessary for the service providing. 9 of the 30 *Family* apps retrieve/store unnecessary sensitive information from users. Comparing Table 23 with Table 21, we find that none of the *Family* apps retrieves the location/audio information for service, 4 of the 4 *Family* apps accessing camera need the permission for service. Therefore, we conclude that the location information requested by the kids apps are mostly not used for service.

Table 23. Transmission Principle—Unnecessary collection (Family)

Apps (Family)	**Location**	**Audio**
Fashion Girls Nail Salon		√
Make Jelly Candy, Sally Spa Salon- Fashion Games, Chocolate Maker, Draw Robots Transformers, 3D Model Dress Up Girl Game, Cute Girl Summer Dress Up, Ice Candy Maker, Cookie Deluxe - Cooking Games	√	

6.4.1.4 App Privacy Risk Analyzer

Since we have measured the actor risks, attribute risks, and transmission risks in the *Family* apps, we can measure their overall privacy risks. Using the algorithms proposed in the system design section, apps' overall privacy risks are presented in Fig. 25. Apps with actor risk scores or attribute risk scores equal to 0 are excluded. An app's privacy risk is 0 when it neither retrieves sensitive information from users nor embeds any third parties or allows other users to view users' information.

According to Fig. 25, the app *3D Model Dress Up Girl Game* is the most risky general kids app. Its overall privacy risk score is 0.73. In particular, the actor risk score is 1, the attribute risk score is 0.4, and the transmission risk score is 0.66. The details of the risk scores are represented in Fig. 26. According to Fig. 26, it is easy to figure out how apps retrieve and distribute their information, and also to find out why one app is more risky than the other. For app information, Fig. 26 reveals that 6 kids apps only embed ad networks, 2 kids apps only allow other users to access the information, 4 kids apps embed ad networks and also allow other users to access the information (e.g., the app *3D Model Dress Up Girl Game*); 11 kids apps only collect one type of sensitive information, 1 kids app collects two types of sensitive information; 3 kids apps violate one type of transmission principles, 4 kids apps violate two types of transmission principles, and 5 kids apps violate all three types of transmission principles. For app comparison, it is clear that the app *Cute Girl Summer Dress Up* is more risky than the app *Little Girl Salon* because the former one violates more types of transmission principles.

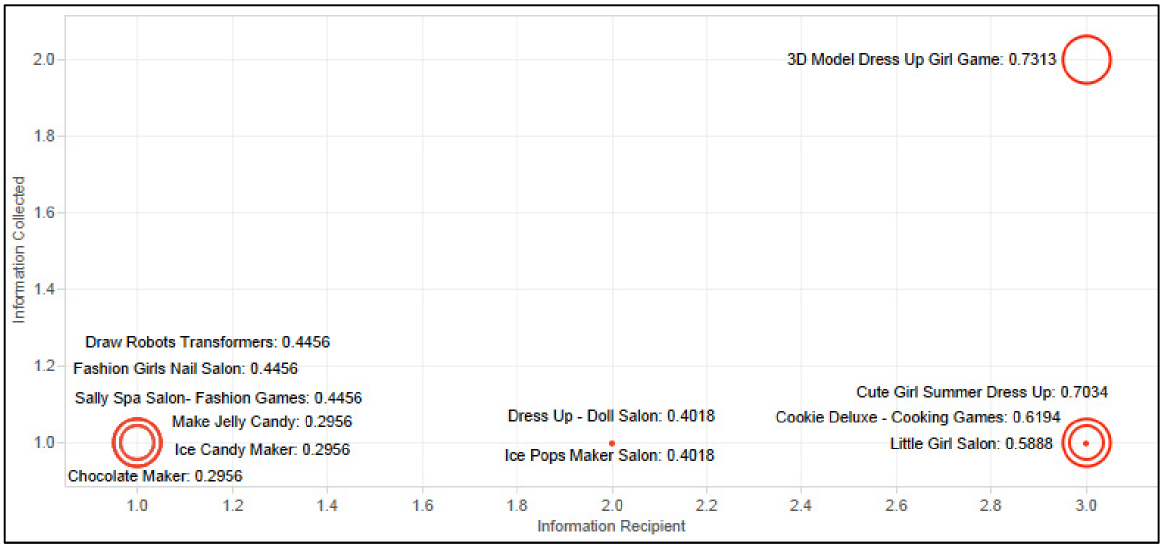
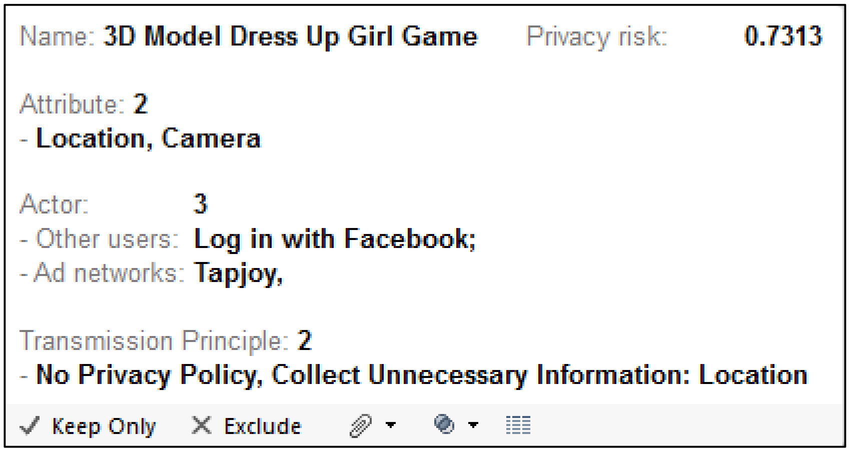


Figure 25. Bubble chat of app privacy risks (Family)



Name: **3D Model Dress Up Girl Game** Privacy risk: **0.7313**

Attribute: **2**
- **Location, Camera**

Actor: **3**
- Other users: **Log in with Facebook;**
- Ad networks: **Tapjoy,**

Transmission Principle: **2**
- **No Privacy Policy, Collect Unnecessary Information: Location**

Figure 26. Privacy risk report for the app *3D Model Dress Up Girl Game*

6.4.2 User Interface Design

We further develop a user interface to assist parents choosing apps for their children (as Fig. 27). The user interface is similar to the current user interface of Google Play store. We add two components to explicitly show apps' privacy ratings: a bubble chart (on the left side), and a colored circle (on the upper-left corner of each app).

The bubble chart helps users to visualize an app's privacy risks in three dimensions: actor, attribute, and transmission principle. The x axis represents apps' actor risk scores, the y axis represents apps' attribute risk scores, and the bubble size represents apps' transmission risk scores. The apps on the bottom left corner will be less risky than the apps on the upper right corner, because the former have lower actor and attribute risk scores. Similarly, the apps with smaller bubble size are less risky than the ones with larger bubble size. Parents can drag the mouse on the bubble chart to select a zone, the selected apps will be placed in a "Selection" box on the right side for closer checking, and the original search results are listed below the "Selection" box.

For each app, there is a colored circle, called the "privacy icon", attached to its upper left corner, representing the app's general privacy risk score. Apps' privacy scores are real numbers between 0~1. When an app's privacy score is between [0, 0.2), it is considered as a safe app, and its privacy icon is green. When an app's privacy score is between [0.2, 0.5), it is considered as a medium risky app, and its privacy icon is yellow. When an app's privacy score is between [0.5, 1), it is considered as a dangerous app, and its privacy icon is red.

When mouse over the "privacy icon", the "privacy dialog" appears, listing the detailed risk scores of each dimension. With it, users can easily find out why an app is risky. If users are confused about what the colors and dimensions mean, they can click on the hyperlink "click to view details" on the "privacy dialog", then more details are revealed (as Fig. 29). With such design, parents can choose apps for their children based on their privacy needs.

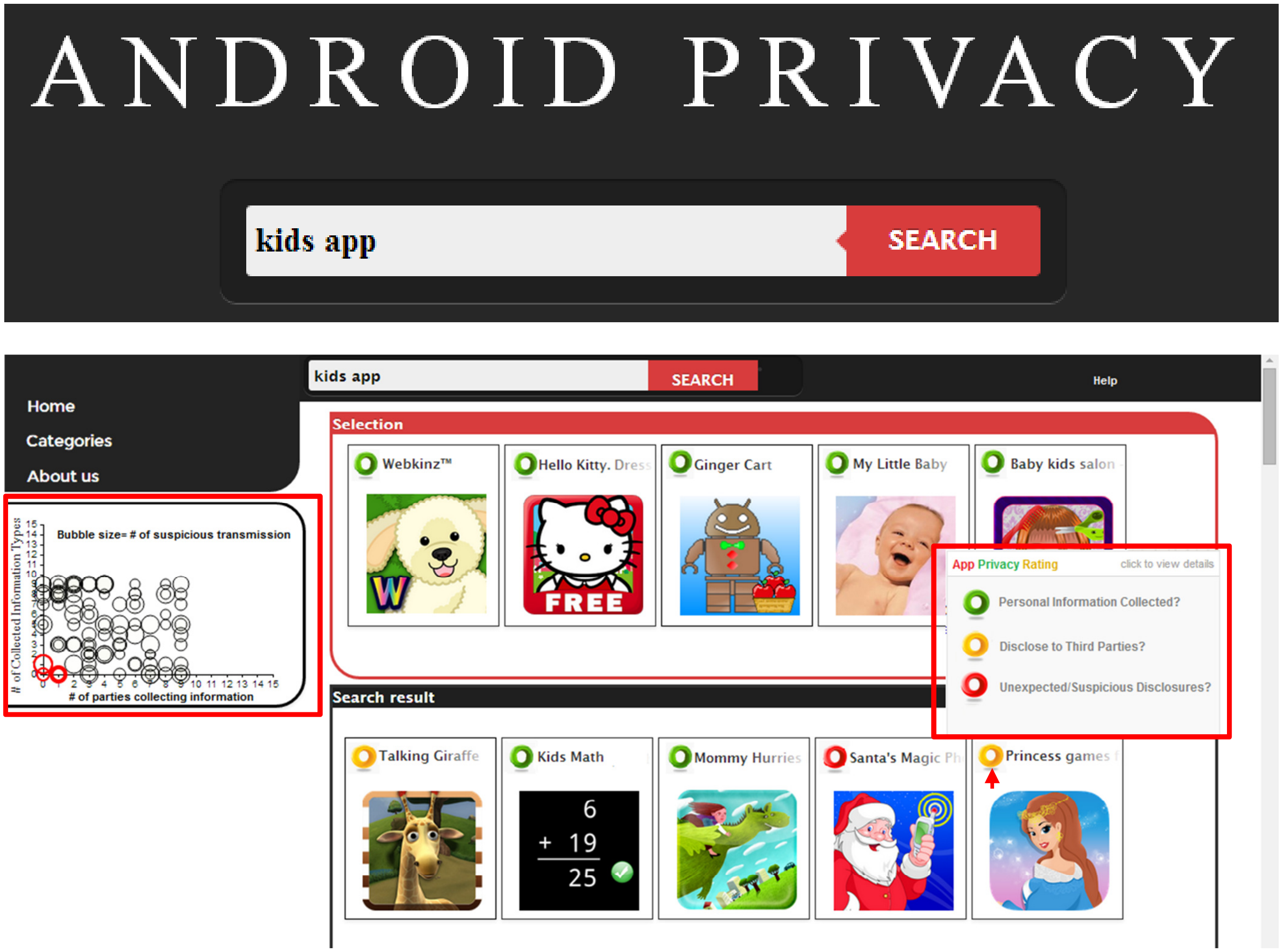
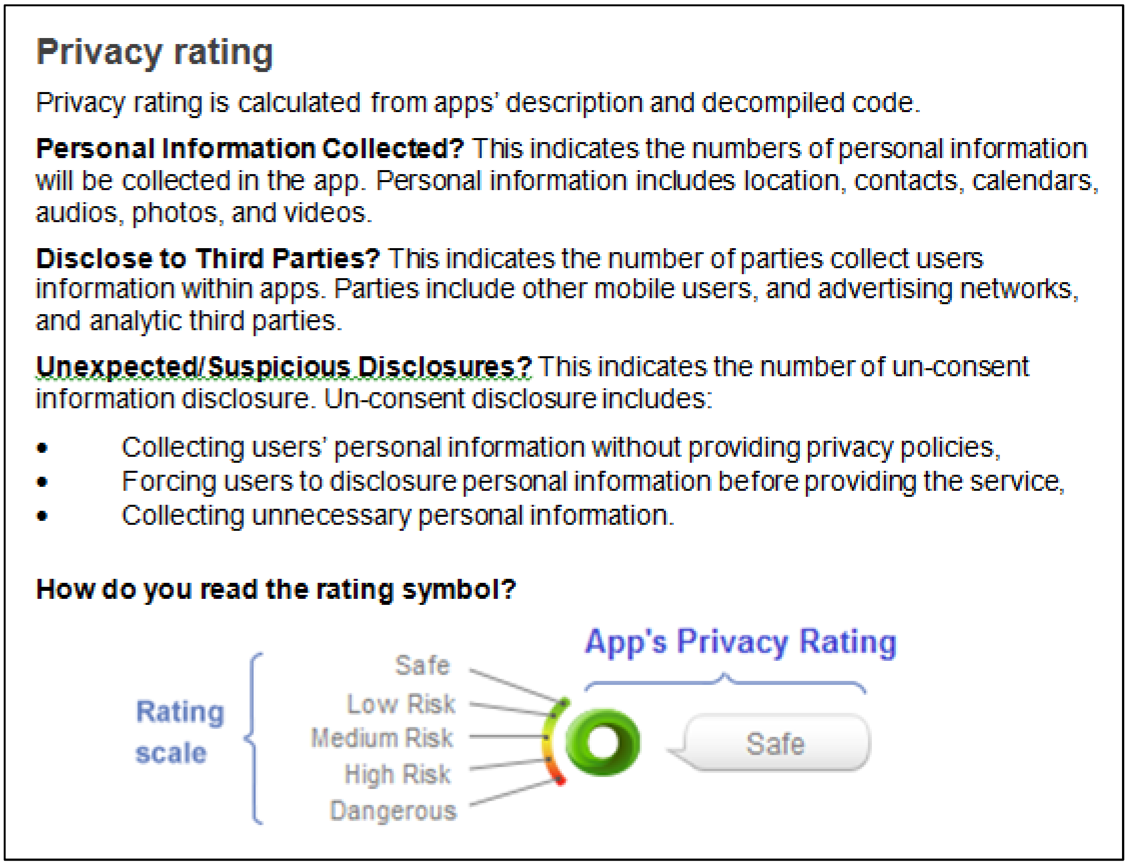


Figure 27. App privacy rating user interface



Privacy rating

Privacy rating is calculated from apps' description and decompiled code.

Personal Information Collected? This indicates the numbers of personal information will be collected in the app. Personal information includes location, contacts, calendars, audios, photos, and videos.

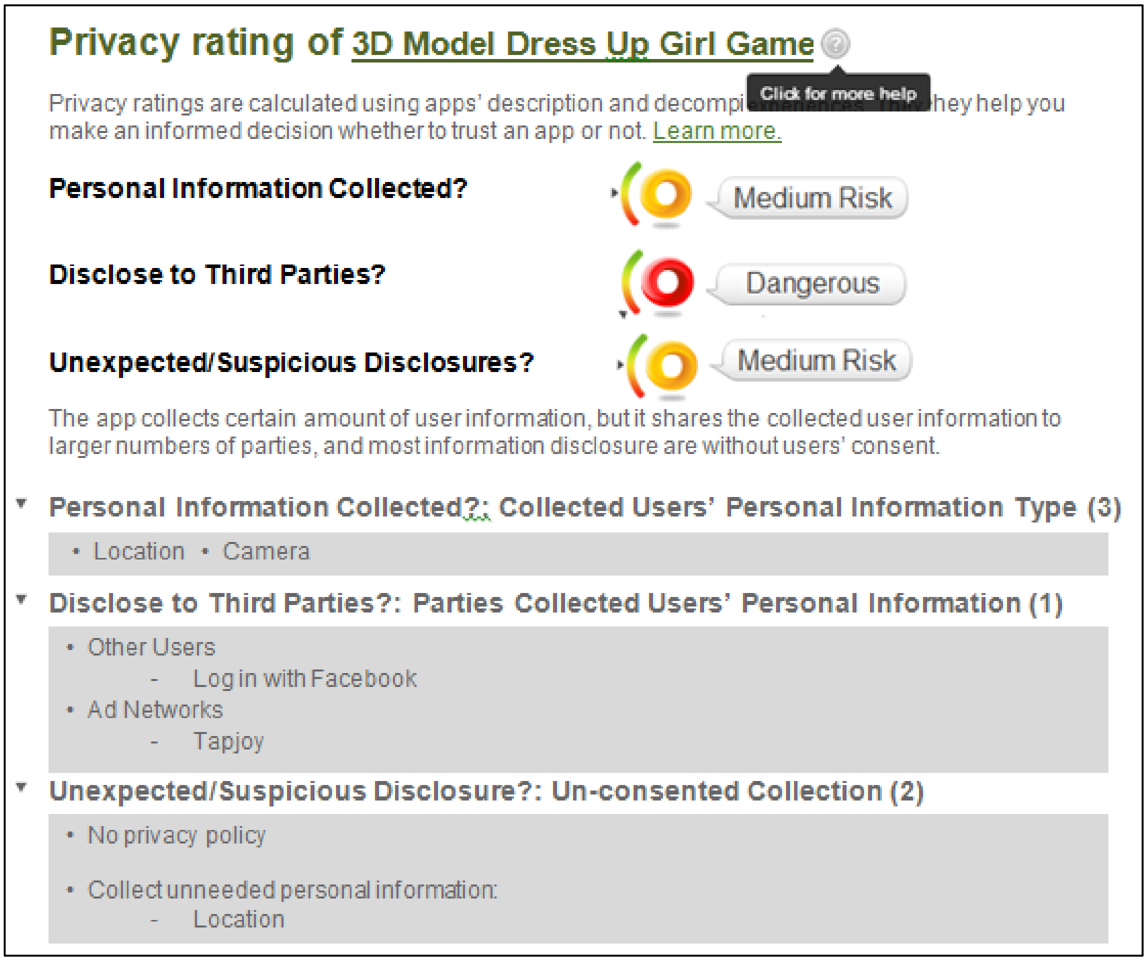
Disclose to Third Parties? This indicates the number of parties collect users information within apps. Parties include other mobile users, and advertising networks, and analytic third parties.

Unexpected/Suspicious Disclosures? This indicates the number of un-consent information disclosure. Un-consent disclosure includes:

- Collecting users' personal information without providing privacy policies,
- Forcing users to disclosure personal information before providing the service,
- Collecting unnecessary personal information.

How do you read the rating symbol?

Figure 28. Privacy rating help page



Privacy rating of 3D Model Dress Up Girl Game

Click for more help

Privacy ratings are calculated using apps' description and decompi hey help you make an informed decision whether to trust an app or not. Learn more.

Personal Information Collected?	Medium Risk
Disclose to Third Parties?	Dangerous
Unexpected/Suspicious Disclosures?	Medium Risk

The app collects certain amount of user information, but it shares the collected user information to larger numbers of parties, and most information disclosure are without users' consent.

Personal Information Collected?: Collected Users' Personal Information Type (3)

- Location
- Camera

Disclose to Third Parties?: Parties Collected Users' Personal Information (1)

- Other Users
  - Log in with Facebook
- Ad Networks
  - Tapjoy

Unexpected/Suspicious Disclosure?: Un-consented Collection (2)

- No privacy policy
- Collect unneeded personal information:
  - Location

Figure 29. App privacy rating detail page

6.5 Discussion and limitation

There are no specific privacy regulations for acquiring children's personal information from mobile devices. In this chapter, we adopt Nissenbaum's contextual integrity theory to examine information boundaries on mobile platforms and to quantitatively measure mobile privacy risks. Nissenbaum argues that privacy depends on four parameters: context, actor, attribute, and transmission principle. To measure mobile privacy risks, we need to measure *where/when* (i.e., context) and *what* types of information (i.e., attribute) is disclosed to *who* (i.e., actor) in *which* (i.e., transmission principle) way. For each app, the attribute, actor, and transmission principle is static. However, context is dynamic. Therefore, we first use attribute, actor, and transmission principle to measure apps' static privacy risks. We then dynamically calculate context-based privacy risks, which is presented in the next section.

To measure apps' static privacy risks, we first extract actor, attribute, and transmission principle features. As to the actor (i.e., information recipient) features, we consider other users and ad networks. We detect whether the apps disclose users' information to other users and whether the apps embed advertising networks' libraries. As to the attribute (i.e., information) features, we consider five types of sensitive information on mobile devices: location, contact, calendar, audio, and camera. As to the transmission principle (i.e., information flow) features, we consider confidentiality, compulsiveness, and necessity. We detect whether the apps have privacy policies, whether the apps force users to disclose information, whether the apps only collect necessary information. As a result, we implement an automatic system which not only can calculate apps' privacy risks, but also provides clear indicators to parents assisting the decision making process.

With the developed automatic app privacy risk measurement system, we perform a case study on the 30 apps in the *Family* category on Google Play store. In the actor risk analysis, 36.7% of the *Family* apps are found disclosing information to other users, 83.3% *Family* apps are found embedding ad networks and each of them embedding 1.5 ad networks on average. In the attribute risk analysis, 40% of the *Family* apps collect sensitive personal information from the users. In the transmission principle analysis, 43.3% of the *Family* apps do not have privacy policies, 1 of the 30 *Family* apps (i.e., the app *Webkinz*™) forces users to log in, and 30% of the *Family* apps retrieve/store unnecessary sensitive information from users. Therefore, the privacy indicators are critical for parents to choose apps for their children.

6.6 Context Risk Analyzer

In this section, we discuss how to dynamically calculate context-based privacy risks on the mobile platforms. We only have preliminary design and results. Complete system relies on the future work.

6.6.1 Contexts

Context has been generally defined as “stimuli and phenomena that surround and, thus, exist in the environment external to the individual” (Mowday & Sutton, 1993). Researchers have suggested considering context information (e.g., location, time, temperature, noise, and light) in designing security policies on mobile devices (Conti et al., 2011). In our research, we interpret contexts by two components: *temporal status, location*. The concern for collecting and distributing CPI differs when those parameters change. *Temporal status* represents the date (i.e., weekday, weekend) and time (i.e., daytime, nighttime) when children use mobile devices. *Location* represents children’s geographic locations while they use mobile devices. Locations/routes include *protected* locations (e.g., school, home) and *un-protected* locations/routes (e.g., casino, bars). While at protected locations/routes at protected time, children are considered to be protected from solicitations.

6.6.2 System Design

We design the *context risk analyzer*, which determines the risk level of children’s physical surrounding using mobile devices’ times and locations, incorporating with public business and crime reports. With the context risk analyzer and access control rules, usability and privacy can be well balanced on mobile devices. In a risky context, the disclosure of personal information from mobile devices should be restricted to prevent cyber-initiated-offline-solicitation. That is, offenders should not be able to identify that a child is around using Bluetooth or context-aware apps (i.e., listing nearby mobile users), such as WeChat and Waze. The personal information on children devices can only be disclosed in safe time and location. A sample access control rule can be defined as:

R = {<time, location >, <disclose>}

where <disclose>::={allowed | denied}. Parents should be allowed to add or configure the rules. For example, they can specify the rule *{<8:00~15:00, school>, <allowed>}* to grant disclosure of children's location information in the protected environment, and add the rule *{<18:00~23:59, any>, <denied>}* to restrict disclosure of children's location information while they are out during the nighttime. With the context-based access control, parents can better protect their children from solicitations.

It is relatively easy for parents to generate access control policies for contexts such as home, school or school bus route whose locations or trajectories are known in advance. However, often the locations and trajectories are not so clear and cannot be determined beforehand. Hence, it is very desirable to have flexible policies based on the context risks. To automatically determine the context risks, we differentiate temporal status into four ranges – weekday day, weekday night, weekend day, and weekend night - to represent differential temporal sensitivities (Benisch et al., 2011). The context risk analyzer can automatically score the risk of each location as a real number in the [0, 1] range. Parents can just list <*context risk less than 0.3*, *allowed*> in the access control rule to protect their children's personal information. The details of the context risk analyzer are explained in the following. With the context risk analyzer, the access control unit of mobile devices can enforce different policies and dynamically protect children's personal information based on the contexts.

There are some academia attempts studying how to score context privacy risks. For general publics, the CMU location privacy group (Cranshaw et al., 2010; Toch et al., 2010) has demonstrated that entropy can be used to represent the location privacy sensitivity. Locations with less number of people are more sensitive. However, for children, two types of locations are risky: age-inappropriate locations and high crime areas.

To determine whether or not a location is age-appropriate, we need to know whether it is surrounded by "good for kids" businesses. Yelp publishes a dataset (Yelp) containing the metadata of 15,585 businesses. In the dataset, some businesses (36.9%, e.g., 5761 businesses) are annotated by Yelp as "true" or "false" in the "good for kids" measurement. For the rest 63.1% businesses, we do not know whether or not they are appropriate for kids. Therefore, for any business, if we want to measure whether or not it is "good for kids", we need to firstly find out whether or not its' category is "good for kids". Each business is categorized by tags. For example, "Rite Aid" is categorized as [["Drugstores", "drugstores"], ["Convenience Stores", "convenience"]]. With the "good for kids" labels and the category tags, we can measure categories' age-inappropriateness for children, and eventually measure businesses' age-inappropriateness. Here is how we measure businesses' age-inappropriateness. Assume the business b is with the category tags as $C = \{c_0, c_1, \dots, c_i, \dots, c_n\}$ $(0 \le i \le n)$, in total N_{c_i} businesses with the category tag c_i are annotated as "true" or "false" for "good for kids", among them n_{c_i} businesses with the category tag c_i are annotated as "false" for "good for kids", then the possibility of the category tag c_i being bad for kids is n_{c_i}/N_{c_i} (further referred as weight). For any location $L = \{latitude, longitude\}$, Yelp search API (YelpSearchAPI) returns 20 businesses within r radius of the location L. We aggregate the weight of the category tags for each business. Businesses weight higher than 0 are determined as bad for kids. Assume there are m "bad for kids" businesses within r radius of the location L, the probability of the location L being age-inappropriate is $B_L = m/20$.

To determine whether or not a location is a high crime area, we use the public crime records for training. The crime map in the whole US area is available online[7]. We use the Baltimore crime dataset (Data.gov, 2014) as an example in this study. The dataset lists all the crime incidents in the Baltimore area from 2009/1/1 to 2014/5/3 (i.e., 1958 days). Each incident is listed with timestamp, coordinate, crime type, and other metadata. For the location $L = \{latitude, longitude\}$, we aggregate the crime incidences happen within r radius. There are 9 types of crime: burglary, robbery, arson, assault, shooting, larceny, auto theft, rape, homicide. Assume these crimes $S = \{s_1, s_2, \ldots, s_j, \ldots, s_9\}$ $(1 \leq j \leq 9)$ are identified within r radius of the location L, s_j is the number of the jth type of crime, so that the crime rate of the location L can be presented as: $C_L = \sqrt{\frac{\sum_{j=1}^{9}\left(\frac{s_j}{1958}\right)^2}{9}}$. When $C_L > 1/3$, it means every single day within r radius of the location L, there are at least one type of crime happening. When $C_L = 1$, it means every single day within r radius of the location L, all 9 types of crime are happening.

In general, the context risk of the location L can be calculated as the quadratic mean of its age-inappropriateness level and crime rate: $\sqrt{\frac{{B_L}^2 + {C_L}^2}{2}}$.

---

[7] http://www.crimemapping.com/

6.6.3 Evaluation and Preliminary Results

We set the testing context as {Time= School time, Location= The top (rating ≥ 8.0) elementary schools in the Baltimore area}, and we measure the age-inappropriate levels and the crime rate of the context.

To measure the context age-inappropriate levels, 5,761 businesses on Yelp with the “true” or “false” labels on the “good for kids” evaluation are used for training. We only count the categories appearing for more than 5 times. In total, 24 categories of businesses are found bad for kids. The sample categories and weights are presented in Table 24.

Category	**Bad for kids Weight**
Dive Bars	0.625
Wineries	0.75
Jazz & Blues	0.8
Dance Clubs	0.88235
Casinos	1

Table 24. “Bad for kids” categories of businesses

With the category weights, we can further determine businesses’ age-inappropriate levels. The annotated businesses in the training set are used as the ground truth. We use the calculated category weights to score the businesses’ age-inappropriate levels, and compare the results with the ground truth. According to Table 25, the precision of the businesses’ age-inappropriate level determination is 86.5%, and the recall is 96.1%. In general, the accuracy of detecting businesses’ age-inappropriateness is 91% (f-score).

N	**I**		**TP**	**TN**	**FP**	**FN**
5761	5204		4500	374	704	183
T	**F**		**Acc (%)**	**P (%)**	**R (%)**	**F_S (%)**
4683	1078		84.6	86.5	96.1	91.0

N: total # of businesses; **I**: # of businesses identified as “good for kids”; **T**: # of businesses annotated as “good for kids”; **F**: # of businesses annotated as “bad for kids”; **TP**: true positive; **TN**: true negative; **FP**: false positive; **FN**: false negative; **Acc**: short for Accuracy, presents the ratio of sum of true positives and true negatives to the total number of instances; **P**: precision, presents the percent of identified apps that are truly positive; **R**: recall, represents the percent of actual positive instances that are correctly identified; $\mathbf{F_S}$: F-score (Yin et al., 2009), represents the weighted harmonic mean of precision and recall.

Table 25. Accuracy of predict businesses' age-inappropriate levels

We then measure the context age-inappropriate levels for the top 21 elementary schools in the Baltimore area. For each location, we use Yelp API to find out the closest 20 businesses within 20 minutes walking distance (r =1.2 miles). The result is presented in Table 26. Empty cells mean the values are equal to 0. According to Table 26, many elementary schools are within a walking distance to bars and clubs. Especially for the *Midtown Academy* elementary school, 4 out of 20 businesses within a walking distance are pubs.

Elementary schools	**S**	**P**	**Y**	**M**	**D**	**L**	**N**	B_L
Chadwick Elementary								
Hillcrest Elementary								
Rodgers Forge Elementary	√	√						0.1
Stoneleigh Elementary	√		√					0.1
Roland Park Elementary/Middle		√		√				0.1
Midtown Academy		√√√√			√	√		**0.3**
Carney Elementary		√√√	√					**0.2**
Lansdowne Elementary								
Pine Grove Elementary		√	√					0.1
Cromwell Valley Elementary Technology	√			√		√		0.15
Summit Park Elementary								
Joppa View Elementary		√						0.05
West Towson Elementary								
The Mount Washington School		√	√√	√				0.2
Chesapeake Terrace Elementary								
Fullerton Elementary								
Gunpowder Elementary	√							0.05
Middleborough Elementary							√	0.05
Relay Elementary		√√						0.1
Westchester Elementary						√		0.05
Woodholme Elementary								

S: Sports Bar; P: Pub, Bar, Brewery; Y: Yoga; M: Massage; D: Dive bar; L: Lounge; N: Nightlife; B_L: Business age-inappropriateness.

Table 26. School context age-inappropriate levels

We also measure the crime rate around those elementary schools. We aggregate the crimes within 1.2 miles from the schools' locations (Table 27), and find that crimes frequently happen within a walking distance of the elementary schools. From 2009/1/1 to 2014/5/3, in total

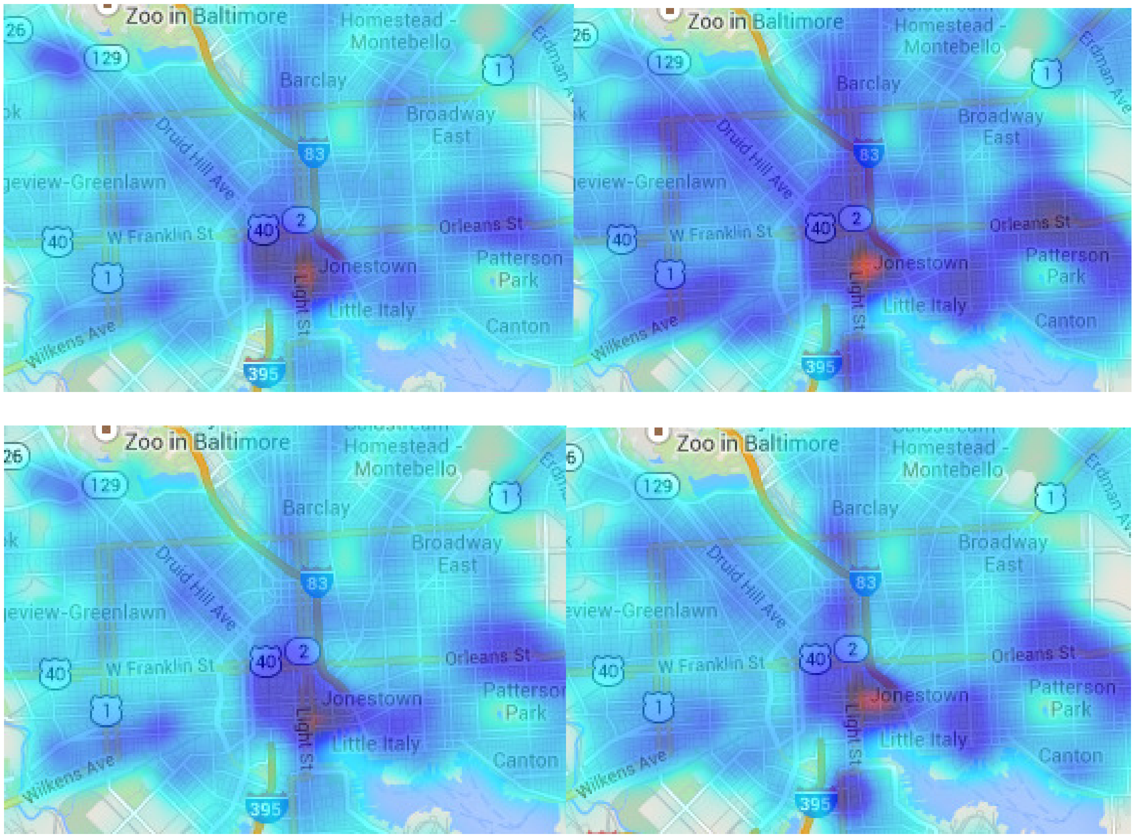
1958 days, 3 elementary schools—Roland Park Elementary/Middle, Midtown Academy, Lansdowne Elementary—have more than one crime incidents on average every single day. Especially for the school “Midtown Academy”, there are 20 crime instances on average every day within a walking distance. With such information, parents really need to think twice before sending their children over.

Elementary schools	**B**	**Ro**	**Ar**	**As**	**S**	**La**	**Au**	**Ra**	**H**	**# of Crime**	C_L	B_L	**Context risk**
Chadwick Elementary													
Hillcrest Elementary													
Rodgers Forge Elementary	385	107	4	228	0	578	118	4	385	1426	0.14	0.1	0.12
Stoneleigh Elementary	398	139	10	387	4	549	223	10	398	1724	0.16	0.1	0.13
Roland Park Elementary/Middle	707	123	8	295	0	1534	205	6	707	2878	**0.32**	0.1	0.24
Midtown Academy	4847	3517	184	11802	340	15933	2391	257	4847	39463	**3.65**	**0.3**	**2.59**
Carney Elementary												0.2	0.14
Lansdowne Elementary	754	268	44	1207	22	1634	395	24	754	4357	**0.40**		0.28
Pine Grove Elementary												0.1	0.07
Cromwell Valley Elementary Technology												0.15	0.11
Summit Park Elementary	121	15	1	53	0	210	59	3	121	462	0.05		0.03
Joppa View Elementary												0.05	0.04
West Towson Elementary													
The Mount Washington School	403	89	5	272	4	947	177	8	403	1906	0.20	0.2	0.20
Chesapeake Terrace Elementary													
Fullerton Elementary	340	106	4	237	7	425	120	2	340	1241	0.12		0.08
Gunpowder Elementary												0.05	0.04
Middleborough Elementary												0.05	0.04
Relay Elementary												0.1	0.07
Westchester Elementary												0.05	0.04
Woodholme Elementary													

# of Crime: Number of crime happen in 2km radius since 2009; B: Burglary; Ro: Robbery; Ar: Arson; As: Assault; S: Shooting; La: Larceny; Au: Auto theft; Ra: Rape; H: Homicide; C_L: Crime rate; B_L: Business age-inappropriateness.

Table 27. School context crime rate

Besides location, we also measure temporal factors on the context risk analysis. Fig. 30 presents the crime heat map in the Baltimore area. According to Fig. 30, we find that weekday nighttime is the most dangerous time to go out in the Baltimore area, especially the downtown area. Therefore, for weekday nighttime, parents should be more careful to let their children out alone without adults' companions. In contrast, weekend daytime is the safest time to go out.



(a) Weekday daytime (b) Weekday nighttime

(c) Weekend daytime (d) Weekend nighttime

Figure 30. Crime heat map in Baltimore, MD area from 2009/1/1 to 2014/5/3

Since we study the context risks on children's protection, when we incorporate time into consideration, we find that around the class dismissal time (15:00~16:00), the crime rate within a walking distance to the *Midtown Academy* elementary school is still as high as 0.33. That is, around the class dismissal time, the *Midtown Academy* elementary school has at least one crime instance every day. Since those crimes are happen right after class dismissal, when children are out waiting parents to pick them up, parents really need to choose schools wisely.

	# of Crime 15:00~16:00	**B**	**Ro**	**Ar**	**As**	**La**	**Au**	**Ra**	C_L
Rodgers Forge Elementary	119	30	3		15	65	6		0.01
Stoneleigh Elementary	125	26	7	1	28	53	10		0.01
Roland Park Elementary/Middle	299	73	10	1	36	167	12		0.04
Midtown Academy	3003	431	186	6	794	1416	158	12	**0.33**
Lansdowne Elementary	338	65	12		72	166	22	1	0.04
Summit Park Elementary	33	8	2		3	16	4		0.00
The Mount Washington School	177	34	2	1	21	102	17		0.02
Fullerton Elementary	90	19	11		17	40	3		0.01

# of Crime 15:00~16:00: Number of crime happen in 15:00~16:00pm in 2km radius since 2009.

Table 28. Crimes happen on class dismissal

According to above analysis results, we find that the context risk analyzer can inform parents about the dynamic context risks of their children.

In the future work, we will implement the context risk analysis algorithm into mobile platforms. In addition, we plan to implement the corresponding access control system to dynamically protect children's personal information on mobile platforms, which can automatically grant or deny information disclosure based on the context risks and the app risks.

6.7 Conclusion

Recently, mobile devices have become a major platform for cyber-solicitation. However, there are no specific privacy regulations for acquiring and using children's personal information from mobile devices. In this study, we adopt Nissenbaum's contextual integrity theory

(Nissenbaum, 2009) to examine privacy risks of mobile applications for the purpose of protecting children against cyber-solicitation. In particular, we specify contexts, actors, attributes, and transmission principles for collecting and distributing children's personal information on mobile platforms, which can further be used to quantitatively measure the static privacy risk levels of mobile apps and the dynamic privacy risk levels of children's physical surroundings. We develop user interface to assist parents making informed decisions when choosing mobile apps. We also propose to develop the context risk analyzer and the access control rules to help dynamically protect children's privacy on different time and locations.

References

148Apps.biz. (2012). App Store Metrics. from http://148apps.biz/app-store-metrics/

Agten, T. (2012). Same Apps Have Different Age Rating in Apple App Store and Google Android Market. from http://www.distimo.com/blog/2012_01_same-apps-have-different-age-rating-in-apple-app-store-and-google-android-market/

Android. Adding an Easy Share Action. from http://developer.android.com/training/sharing/shareaction.html

Apple. (2012a). App Store Review Guidelines. from https://developer.apple.com/appstore/guidelines.html

Apple. (2012b). Application Ratings. from http://itunes.apple.com/WebObjects/MZStore.woa/wa/appRatings

Barkhuus, L. (2012). *The mismeasurement of privacy: using contextual integrity to reconsider privacy in HCI.* Paper presented at the Proc CHI 2012.

Benisch, M., Kelley, P.G., Sadeh, N., & Cranor, L.F. (2011). Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal and Ubiquitous Computing, 15*(7), 679-694.

Carmichael, M. (2011). Stat of the Day: 25% of Toddlers Have Used a Smartphone. from http://adage.com/article/adagestat/25-toddlers-a-smartphone/229082/

Chen, Y., Xu, H., Zhou, Y., & Zhu, S. (2013). *Is This App Safe for Children? A Comparison Study of Maturity Ratings on Android and iOS Applications.* Paper presented at the Proc. WWW.

Chen, Y., Zhu, S., Xu, H., & Zhou, Y. (2013). *Children's Exposure to Mobile In-App Advertising: An Analysis of Content Appropriatenessh.* Paper presented at the Proc. of SocialCom, Washington D.C., USA.

Cheng, J. (2007). Report: 80 percent of blogs contain "offensive" content. *ars technica* Retrieved 11/7, 2011, from http://arstechnica.com/security/news/2007/04/report-80-percent-of-blogs-contain-offensive-content.ars

Chia, P.H., Yamamoto, Y., & Asokan, N. (2012). *Is this app safe?: a large scale study on application permissions and risk signals.* Paper presented at the Proceedings of the 21st international conference on World Wide Web.

CNN. (2013). When bullying goes high-tech. from http://edition.cnn.com/2013/02/27/health/cyberbullying-online-bully-victims/

Conti, M., Nguyen, V.T.N., & Crispo, B. (2011). CRePE: Context-related policy enforcement for Android *Information Security* (pp. 331-345): Springer.

Costello, S. (2014). Using the iPhone Privacy Settings in iOS. from http://ipod.about.com/od/UsingiOS6/ss/Using-Iphone-Privacy-Settings-In-Ios.htm

Cranshaw, J., Toch, E., Hong, J., Kittur, A., & Sadeh, N. (2010). *Bridging the gap between physical location and online social networks.* Paper presented at the Proceedings of the 12th ACM international conference on Ubiquitous computing.

Data.gov. (2014). Baltimore Victim Based Crime Data. from https://catalog.data.gov/dataset/bpd-part-1-victim-based-crime-data

Dynamic program analysis. (2013). *Wikipedia*, from http://en.wikipedia.org/wiki/Dynamic_program_analysis

Egele, M., Kruegel, C., Kirda, E., & Vigna, G. (2011). *PiOS: Detecting privacy leaks in iOS applications.* Paper presented at the Proceedings of the Network and Distributed System Security Symposium.

eMarketer. Hesitant parents can't keep kids away from smartphones. from http://www.emarketer.com/Article/Hesitant-Parents-Cant-Keep-Kids-Away-Smartphones/1010075

Enck, W., Gilbert, P., Chun, B.-G., Cox, L.P., Jung, J., McDaniel, P., & Sheth, A. (2010a). *TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones.* Paper presented at the Proc. OSDI.

Enck, W., Gilbert, P., Chun, B.-G., Cox, L.P., Jung, J., McDaniel, P., & Sheth, A.N. (2010b). *TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones.* Paper presented at the Proceedings of the 9th USENIX conference on Operating systems design and implementation.

EnterpriseAppsTech. (2012). Smartphone Usage Growing Amongst Teenagers. from http://www.appstechnews.com/blog-hub/2012/mar/26/smartphone-usage-growing-amongst-teenagers/

Evozi. (2012). APK Downloader. from http://apps.evozi.com/apk-downloader/

Facebook. Log in with Facebook. from https://developers.facebook.com/docs/android/scrumptious/authenticate

Felt, A.P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). *Android permissions demystified.* Paper presented at the Proc. CCS.

Felt, A.P., Greenwood, K., & Wagner, D. (2011). *The effectiveness of application permissions.* Paper presented at the Proceedings of the 2nd USENIX conference on Web application development.

Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D. (2012). *Android permissions: User attention, comprehension, and behavior.* Paper presented at the Proceedings of the Eighth Symposium on Usable Privacy and Security.

Forbes. (2013). more than half of parents plan to buy tech gifts for young children.

Frank, M., Dong, B., Porter Felt, A., & Song, D. (2012). *Mining Permission Request Patterns from Android and Facebook Applications.* Paper presented at the Proc. of 12th ICDM.

FTC. COPPA - Children's Online Privacy Protection Act. from http://www.coppa.org/

FTC. (2012). Mobile Apps for Kids: Current Privacy Disclosures are Disappointing *Federal Trade Commission*.

FTC. (2013). *Mobile Privacy Disclosures: Building Trust Through Transparency*.

Google. android-apktool - A tool for reverse engineering Android apk files. from https://code.google.com/p/android-apktool/

Google. Android Debug Bridge | Android Developers. from http://developer.android.com/tools/help/adb.html

Google. logcat | Android Developers. from http://developer.android.com/tools/help/logcat.html

Google. (2012). Application Content Ratings. from http://support.google.com/googleplay/bin/answer.py?hl=en&answer=1075738

Grace, M.C., Zhou, W., Jiang, X., & Sadeghi, A.-R. (2012a). *Unsafe exposure analysis of mobile in-app advertisements.* Paper presented at the Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks.

Grace, M.C., Zhou, W., Jiang, X., & Sadeghi, A.-R. (2012b). *Unsafe exposure analysis of mobile in-app advertisements.* Paper presented at the Proc Wisec 2012.

Graziano, D. (2012). Android and iOS Still Rule the Mobile World; Microsoft and RIM Have Long Roads Ahead. from http://bgr.com/2012/11/02/android-ios-market-share-dominate-microsoft-rim/

Gwenn, S.O.K., Kathleen, C.-P., & MEDIA, C.O.C.A. (2011). Clinical report--the impact of social media on children, adolescents, and families. *Pediatrics*.

Haddadi, H., Hui, P., Henderson, T., & Brown, I. (2011). Targeted advertising on the handset: Privacy and security challenges *Pervasive Advertising* (pp. 119-137): Springer.

Hansen, J.V., Lowry, P.B., Meservy, R.D., & McDonald, D.M. (2007). Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. *Decision Support Systems, 43*(4), 1362-1374.

Hardawar, D. (2012, March, 29). The magic moment:Smartphones now half of all U.S. mobiles. *venturebeat.com*, from http://venturebeat.com/2012/03/29/the-magic-moment-smartphones-now-half-of-all-u-s-mobiles/

Hornyack, P., Han, S., Jung, J., Schechter, S., & Wetherall, D. (2011). *These aren't the droids you're looking for: retrofitting android to protect data from imperious applications.* Paper presented at the Proceedings of the 18th ACM conference on Computer and communications security.

JavaExperience. How to share data using sherlock action bar in Android. from http://www.javaexperience.com/how-to-share-data-using-sherlock-action-bar-in-android/

JaveExperience. How to share data in Android. from http://www.javaexperience.com/how-to-share-data-in-android/

Jay, T., & Janschewitz, K. (2008). The pragmatics of swearing. *Journal of Politeness Research. Language, Behaviour, Culture, 4*(2), 267-288.

Johnson, T., Shapiro, R., & Tourangeau, R. (2011). National survey of American attitudes on substance abuse XVI: Teens and parents. *The National Center on Addiction and Substance Abuse.* Retrieved 11/7, 2011, from http://www.casacolumbia.org/templates/NewsRoom.aspx?articleid=648&zoneid=51

Kang, C. (2011). Inappropriate content making its way to mobile apps. from http://www.washingtonpost.com/business/economy/inappropriate-content-making-its-way-to-mobile-apps/2011/10/05/gIQAnYB4kL_story.html

Kelley, P.G., Consolvo, S., Cranor, L.F., Jung, J., Sadeh, N., & Wetherall, D. (2012). *A Conundrum of Permissions: Installng Applications on an Android Smartphone.* Paper presented at the Proc. of Workshop on USEC.

Kontostathis, A. (2009). *ChatCoder: Toward the tracking and categorization of internet predators.* Paper presented at the Proc Text Mining Workshop 2009.

Kontostathis, A., Edwards, L., & Leatherman, A. (2009). Chatcoder: Toward the tracking and categorization of internet predators. *In Proc. Text Mining Workshop 2009 held in conjunction with the Ninth SIAM International Conference on Data Mining*.

Leontiadis, I., Efstratiou, C., Picone, M., & Mascolo, C. (2012). *Don't kill my ads!: balancing privacy in an ad-supported mobile application market.* Paper presented at the Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications.

Lin, J., Amini, S., Hong, J., Sadeh, N., Lindqvist, J., & Zhang, J. (2012). *Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing.* Paper presented at the Proc. of 14th Ubicomp.

Lin, J., Sadeh, N., Amini, S., Lindqvist, J., Hong, J.I., & Zhang, J. (2012). *Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing.* Paper presented at the Proc. of 14th ACM Ubicomp.

Ma, J., Teng, G., Chang, S., Zhang, X., & Xiao, K. (2011). Social network analysis based on authorship identification for cybercrime investigation. *Intelligence and Security Informatics*, 27-35.

Mahmud, A., Ahmed, Kazi Zubair, and Khan, Mumit (2008). *Detecting flames and insults in text.* Paper presented at the Proc. of 6th International Conference on Natural Language Processing (ICON' 08).

Malhotra, N.K., Kim, S.S., & Agarwal, J. (2004). Internet users' information privacy concerns (IUIPC): the construct, the scale, and a causal model. *ISR, 15*(4), 336-355.

MarketingPilgrim. (2013). Majority of Mobile Users Would Rather Engage an Ad Than Pay for an Upgrade. *Marketing Pilgrim*, from http://www.marketingpilgrim.com/2013/05/majority-of-mobile-users-would-rather-engage-an-ad-than-pay-for-an-upgrade.html

Marneffe, M.-C.d., MacCartney, B., & Manning, C.D. (2006). *Generating typed dependency parses from phrase structure parses*. Paper presented at the LREC.

Mazurek, M.L., Arsenault, J., Bresee, J., Gupta, N., Ion, I., Johns, C., Lee, D., Liang, Y., Olsen, J., & Salmon, B. (2010). *Access control for home data sharing: Attitudes, needs and practices.* Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

McEnery, A., Baker, J., & Hardie, A. (2000). *Swearing and abuse in modern British English*. Paper presented at the Practical Applications of Language Corpora.

Mitchell, A., Rosenstiel, T., Santhanam, L.H., & Christian, L. (2012). Future of Mobile News. from http://www.journalism.org/analysis_report/future_mobile_news

Möller, A., Diewald, S., Roalter, L., Michahelles, F., & Kranz, M. (2012). Update Behavior in App Markets and Security Implications: A Case Study in Google Play. *LARGE*, 3.

Mowday, R.T., & Sutton, R.I. (1993). Organizational behavior: Linking individuals and groups to organizational contexts. *Annual review of psychology, 44*(1), 195-229.

Nauman, M., Khan, S., & Zhang, X. (2010a). *Apex: extending android permission model and enforcement with user-defined runtime constraints.* Paper presented at the Proc. of 5th ACM ASIACCS.

Nauman, M., Khan, S., & Zhang, X. (2010b). *Apex: extending android permission model and enforcement with user-defined runtime constraints.* Paper presented at the Proc. of 5th ASIACCS.

NielsenWire. (2011). Play Before Work: Games Most Popular Mobile App Category in US. from http://blog.nielsen.com/nielsenwire/online_mobile/games-most-popular-mobile-app-category

Nissenbaum, H. (2004). Privacy as contextual integrity. *Wash. L. Rev., 79*, 119.

Nissenbaum, H. (2009). *Privacy in context: Technology, policy, and the integrity of social life*: Stanford University Press.

O'Neill, T., & Zinga, D. (2008). *Children's rights: multidisciplinary approaches to participation and protection*: Univ of Toronto Pr.

Olson, P. (2012). 5 Eye-Opening Stats That Show The World Is Going Mobile. from http://www.forbes.com/sites/parmyolson/2012/12/04/5-eye-opening-stats-that-show-the-world-is-going-mobile/

Orebaugh, A., & Allnutt, D.J. (2010). Data mining instant messaging communications to perform author identification for cybercrime investigations. *Digital Forensics and Cyber Crime*, 99-110.

Palfrey, J., Sacco, D., Boyd, D., DeBonis, L., & Tatlock, J. (2008). *Enhancing Child Safety & Online Technologies.* Paper presented at the Final Report of the Internet Safety Technical Task Force, Berkman Center for Internet and Society, Harvard University.

Pandita, R., Xiao, X., Yang, W., Enck, W., & Xie, T. (2013). *WHYPER: towards automating risk assessment of mobile applications.* Paper presented at the Proc. of 22nd USENIX.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. *In EMNLP'02: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, 79-86.

Pazienza, M., & Tudorache, A. (2011). Interdisciplinary contributions to flame modeling. *AI* IA 2011: Artificial Intelligence Around Man and Beyond*, 213-224.

Pearce, P., Felt, A.P., Nunez, G., & Wagner, D. (2012). *AdDroid: Privilege separation for applications and advertisers in Android.* Paper presented at the Proceedings of AsiaCCS.

Pendar, N. (2007). *Toward spotting the pedophile telling victim from predator in text chats.* Paper presented at the Proceedings of the First IEEE International Conference on Semantic Computing.

Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012a). *Using probabilistic generative models for ranking risks of android apps.* Paper presented at the Proc CCS 2012.

Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012b). *Using probabilistic generative models for ranking risks of android apps.* Paper presented at the Proc. of 19th ACM CCS.

Rasmussen, R.H. (2011). Unreliable ratings on mobile apps. from http://kidsandmedia.org/unreliable-ratings-on-mobile-apps/

Razavi, A., Inkpen, D., Uritsky, S., & Matwin, S. (2010). Offensive language detection using multi-level classification. *Advances in Artificial Intelligence, 6085/2010*, 16-27.

readwrite. (2009). How do iphone users find new apps. from http://readwrite.com/2009/08/11/how_do_iphone_users_find_new_apps

Sadeh, N., Hong, J., Cranor, L., Fette, I., Kelley, P., Prabaker, M., & Rao, J. (2009). Understanding and capturing people's privacy policies in a mobile social networking application. *Personal and Ubiquitous Computing, 13*(6), 401-412.

Shekhar, S., Dietz, M., & Wallach, D.S. (2012). Adsplit: Separating smartphone advertising from applications. *CoRR, abs/1202.4030.*

Siegler, M. (2009). Here's How iPhone App Store Ratings Work. Hint: They Don't., from http://techcrunch.com/2009/06/29/heres-how-iphone-app-store-ratings-work-hint-they-dont/

Smith, H.J. (2004). Information privacy and its management. *MIS Quarterly Executive, 3*(4), 201-213.

Spertus, E. (1997). Smokey: Automatic recognition of hostile messages. *Innovative Applications of Artificial Intelligence (IAAI) '97*.

Static program analysis. (2013). *Wikipedia*, from http://en.wikipedia.org/wiki/Static_program_analysis

Statista. (2012). Number of available applications in the Google Play Store from December 2009 to September 2012. from http://www.statista.com/statistics/74368/number-of-available-applications-in-the-google-play-store/

Symonenko, S., Liddy, E.D., Yilmazel, O., Del Zoppo, R., Brown, E., & Downey, M. (2004). Semantic analysis for monitoring insider threats. *Intelligence and Security Informatics*, 492-500.

Ter Louw, M., Ganesh, K.T., & Venkatakrishnan, V. (2010). *Adjail: Practical enforcement of confidentiality and integrity policies on web advertisements.* Paper presented at the 19th USENIX Security Symposium.

Theguardian. (2013). Nearly one in 10 children gets first mobile phone by age five, says study. from http://www.theguardian.com/money/2013/aug/23/children-first-mobile-age-five

Thinksky. iTools One-stop manager of your iPhone, iPad & iPod Touch. from http://itools.hk/en_index.htm

Toch, E., Cranshaw, J., Drielsma, P.H., Tsai, J.Y., Kelley, P.G., Springfield, J., Cranor, L., Hong, J., & Sadeh, N. (2010). *Empirical models of privacy in location sharing.* Paper presented at the Proceedings of the 12th ACM international conference on Ubiquitous computing.

Tsou, B.K.Y., Yuen, R.W.M., Kwong, O.Y., Lai, T.B.Y., & Wong, W.L. (2005). Polarity classification of celebrity coverage in the Chinese press. *Paper presented at the International Conference on Intelligence Analysis*.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *In Proceedings of the Association for Computational Linguistics (ACL)*, 417-424.

Tynes, B., Reynolds, L., & Greenfield, P.M. (2004). Adolescence, race, and ethnicity on the Internet: A comparison of discourse in monitored vs. unmonitored chat rooms. *Journal of Applied Developmental Psychology, 25*(6), 667-684.

Web of Trust (WOT). from http://www.mywot.com

Whitelaw, C., Garg, N., & Argamon, S. (2005). *Using appraisal groups for sentiment analysis.* Paper presented at the Proceedings of the 14th ACM International Conference on Information and Knowledge Management NY, USA.

Whyper. from https://sites.google.com/site/whypermission/

Wolak, J., Mitchell, K., & Finkelhor, D. (2006). *Online victimization of youth: Five years later*: National Center for Missing & Exploited Children.

Xu, Z., & Zhu, S. (2010). *Filtering offensive language in online communities using grammatical relations.* Paper presented at the Proceedings of The Seventh Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference (CEAS'10).

Ybarra, M.L., Mitchell, K.J., Finkelhor, D., & Wolak, J. (2007). Internet prevention messages: Targeting the right online behaviors. *Archives of Pediatrics & Adolescent Medicine, 161*(2), 138.

Ye, Q., Shi, W., & Li, Y. (2006). *Sentiment classification for movie reviews in Chinese by improved semantic oriented approach.* Paper presented at the HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on System Sciences.

Yelp. from http://www.yelp.com/dataset_challenge/

YelpSearchAPI. from http://www.yelp.com/developers/documentation/v2/search_api

Yin, D., Xue, Z., Hong, L., & Davison, B. (2009). *Detection of harassment on Web 2.0.* Paper presented at the the Content Analysis in the Web 2.0 Workshop.

Zhang, C., Zeng, D., Li, J., Wang, F.Y., & Zuo, W. (2009). Sentiment analysis of Chinese documents: from sentence to document level. *Journal of the American Society for Information Science and Technology, 60*(12), 2474-2487.

Zheng, R., Li, J., Chen, H., & Huang, Z. (2006). A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society of Information Science and Technology, 57*(3), 378-393.

Zheng, R., Qin, Y., Huang, Z., & Chen, H. (2010). Authorship analysis in cybercrime investigation. *Intelligence and Security Informatics*, 959-959.

Vita

Ying Chen

Ying Chen received the B.E. degree in Information Security from Beijing University of Posts and Telecommunications, Beijing, China, in the years of 2008. She enrolled in the Ph.D. program of Computer Science and Engineering at The Pennsylvania State University in August 2008. On December 2011, she received a M.S degree in Computer Science and Engineering at The Pennsylvania State University. She is a student member of the IEEE.

Publications during the Ph. D. study:

- Ying Chen, Sencun Zhu, Heng Xu, and Yilu Zhou. “Children’s Exposure to Mobile In-App Advertising: An Analysis of Content Appropriateness.” Proceedings of the ASE/IEEE International Conference on Social Computing (SocialCom), 2013.

- Ying Chen, Heng Xu, Yilu Zhou and Sencun Zhu. “Is This App Safe for Children? A Comparison Study of Maturity Ratings on Android and iOS Applications.” Proceedings of the 22nd International World Wide Web conference (WWW), 2013.

- Ying Chen, Yilu Zhou, Sencun Zhu and Heng Xu. “Detecting Offensive Languages in Social Media to Protect Adolescent Online Safety.” Proceedings of The ASE/IEEE International Conference on Social Computing (SocialCom), 2012.

- Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. "Poster: oFBI: Detecting Offensive Language in Social Networks for Protection of Youth Online Safety." Proceeding of Symposium On Usable Privacy and Security (SOUPS), 2011.