Protecting Your Children from Inappropriate Content in Mobile Apps: An Automatic Maturity Rating Framework

Bing Hu
Samsung Research America
binghu2006@gmail.com

Bin Liu
Rutgers University
binben.liu@rutgers.edu

Neil Zhenqiang Gong
ECpE, Iowa State University
neilgong@iastate.edu

Deguang Kong
Samsung Research America
doogkong@gmail.com

Hongxia Jin
Samsung Research America
hongxia.jin@samsung.com

ABSTRACT

Mobile applications (Apps) could expose children or adolescents to mature themes such as sexual content, violence and drug use, which harms their online safety. Therefore, mobile platforms provide rating policies to label the maturity levels of Apps and the reasons why an App has a given maturity level, which enables parents to select maturity-appropriate Apps for their children. However, existing approaches to implement these maturity rating policies are either costly (because of expensive manual labeling) or inaccurate (because of no centralized controls). In this work, we aim to design and build a machine learning framework to automatically predict maturity levels for mobile Apps and the associated reasons with a high accuracy and a low cost.

To this end, we take a multi-label classification approach to predict the mature contents in a given App and then label the maturity level according to a rating policy. Specifically, we extract novel features from App descriptions by leveraging *deep learning* techniques to automatically capture the semantic similarity between words and adapt Support Vector Machine to capture label correlations with *pearson correlation* in a multi-label classification setting. Moreover, we evaluate our approach and various baseline methods using datasets that we collected from both App Store and Google Play. We demonstrate that, with only App descriptions, our approach already achieves 85% Precision for predicting mature contents and 79% Precision for predicting maturity levels, which substantially outperforms baseline methods.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Measurement, Experimentation

CIKM'15, October 19–23, 2015, Melbourne, VIC, Australia.

DOI: http://dx.doi.org/10.1145/2806416.2806579.

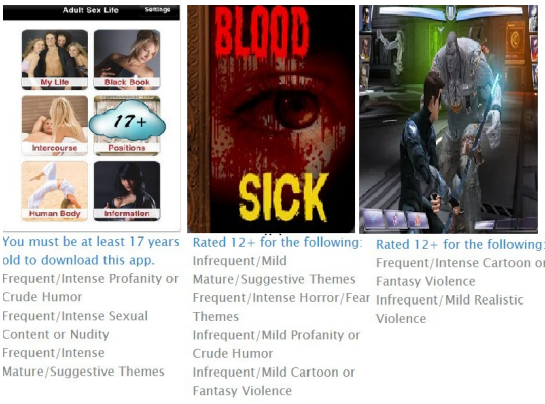


Figure 1: The mature contents and maturity levels of three Apps on App Store.

Keywords

Mobile Apps; Content Rating; Text Mining; Deep Learning; Pearson correlation.

1. INTRODUCTION

Mobile devices are becoming more and more popular in the past few years. However, Apps could expose children or adolescents to mature themes such as sexual content, violence, and drug use, which are harmful to their growth and development. Indeed, research from psychology has long established that teenagers who are exposed to content that glamorizes drug use, sex, or violence tend to engage in those activities themselves [27, 13, 28].

Therefore, similar to the conventional video game and movie industry, mobile platforms provide mechanisms to rate the maturity levels of Apps, which enables parents to select maturity-appropriate mobile Apps for their children. For instance, App Store has a maturity rating policy, which consists of four maturity levels, i.e., *4+, 9+, 12+*, and *17+*. Apps with different maturity levels are suitable for users with different ages, e.g., Apps with 17+ maturity level are appropriate for users who are at least 17 years old. In addition to the maturity level, App Store also identifies the detailed mature contents which make an App be rated as a specific maturity level. These mature contents are helpful for users to better understand the App, and to guide developers to modify their Apps in order to increase their audience population. For instance, Figure 1 shows the mature

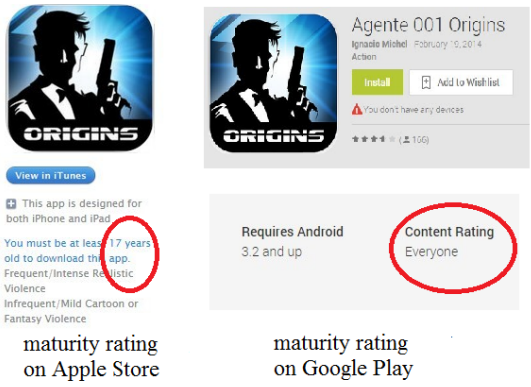


Figure 2: Inconsistency of maturity ratings for the same app in App Store and Google Play.

contents and maturity levels of three Apps on App Store. Google Play also establishes a rating policy that includes four maturity levels, i.e., *Everyone, Low Maturity, Medium Maturity,* and *High Maturity*, and they are corresponding to the four maturity levels on App Store.

Existing approaches to implement these maturity rating policies are either costly or inaccurate. Specifically, Apple Inc. hires employees to manually examine each submitted App in order to identify its maturity level and the associated reasons. Given the large amount of new Apps, e.g., 20,000 new Apps were submitted to App Store per month as of 2014 [2], this manual labeling approach is very costly and time-consuming. Unlike the centralized rating service provided by App Store, Google Play requires developers to label the maturity levels for their own Apps according to the Google Play's rating policy. These self-reported maturity levels are determined at the time when developers submit their Apps to Google Play and remain unchanged until users report inappropriateness. Although Google Play's strategy is scalable and less costly, the maturity ratings reported by developers could be inaccurate. Moreover, various reports [15, 25] have shown growing concerns among parents who have experienced inaccurate maturity ratings of Apps. Figure 2 shows an example of the mislabelling of the maturity ratings by the Android developers. Chen et al. [5] rated the maturity of apps using a keyword matching method through manual identification of the sensitive words that are highly correlated with apps' maturity in App descriptions. Their method achieves limited labeling accuracy because it does not consider the semantic meanings of words. Moreover, their method cannot produce evidences about why an App has a worse rating than another one.

In this work, we aim to design a scalable framework that automatically labels App maturity level and the associated reasons accurately. Our framework, called *Automatic App Maturity Rating (AAMR)*, takes a rating policy and the description of an App as input, and predicts the mature content in the App and the maturity level of the App. We choose App description because it describes the content and functionality of an App, and thus it is a good indicator of the mature content (if any) in the App. However, App description based rating analysis faces several challenges:

⊙ Apps often have *short* and *concise* descriptions in order for users to quickly understand the Apps.

⊙ Natural languages are *ambiguous*, i.e., the same word or phrase has different meanings at different contexts; and natural languages have *synonyms*, i.e., the same mature theme can be expressed with different words or phrases.

⊙ Mature contents are correlated, i.e., some mature contents are likely to co-occur in Apps while some mature contents are mutually exclusive.

To address these challenges, we propose a two-stage machine learning approach to first predict the mature contents and then label the maturity levels. Specifically, we extract novel features from App descriptions. In particular, we use *word to vector model* [23, 24], which leverages *deep learning* technique to automatically capture the semantic similarity between words; to mitigate the language ambiguity, we use the bag-of-words feature to capture the context and global word distributions. With these features, we map the mature-content prediction to be a multi-label classification problem and we adapt SVM to capture label correlations. We choose Support Vector Machine (SVM) [7] for rating prediction since it was shown to outperform other classifiers for short text analysis [11]. We evaluate our approach and various baseline methods using large-scale datasets that we collected from both App Store and Google Play. We demonstrate that, using only App descriptions, our approach can already achieve 85% Precision on maturity content prediction and 79% Precision on maturity level prediction, and our approach substantially outperforms baseline methods including both automatic and human-labeling approaches.

The key contributions are summarized as follows:

- We propose a machine learning framework, called Automatic App Maturity Rating (AAMR), to automatically predict mature contents and label maturity levels accurately. To the best of our knowledge, this is the *first* systematic study about automatic maturity rating for mobile Apps.

- We extract novel features from App descriptions by leveraging *deep learning* techniques. Moreover, we adapt standard SVM as a *multi-label* classifier to capture label correlations using *pearson correlation*.

- We comprehensively evaluate our approach and other baseline methods using *large-scale real-world* datasets that we collected from both App Store and Google Play, and we show that our approach substantially outperforms baseline methods.

2. BACKGROUND AND DESIGN GOALS

In this section, we first introduce the maturity rating policies adopted by App Store and Google Play, and then we discuss the limitations of current approaches to implement the policies, which is followed by our design goals of our automatic App maturity rating system.

2.1 Two Maturity Rating Policies

App Store's Rating Policy: Table 1 shows the rating policy of App Store [16]. In this policy, Apps are classified into four categories, i.e., *4+*, *9+*, *12+*, and *17+*. Apps with the maturity level of *17+* are appropriate for users who are at least 17 years old. The maturity level of an App is related to the following mature contents: violence, sexual/maturity, profanity/humor, alcohol/drug/tobacco,etc. Moreover, different intensity of mature contents results in different maturity levels. For example, an App is rated as *9+* if it contains

Table 1: App Store Maturity Rating Policy

Content	Level	Rating	#
realistic violence	Infrequent/mild	9+	1
	Frequent/intense	12+	2
cartoon/fantasy violence	Frequent/intense	9+	3
horror-themed content	Infrequent/mild	9+	4
	Frequent/intense	12+	5
profanity/crude humor	Infrequent/mild	9+	6
	Frequent/intense	12+	7
sexual content/nudity	Infrequent/mild	12+	8
	Frequent/intense	17+	9
mature/suggestive content	Frequent/intense	17+	10
alcohol/tobacco/drug	Infrequent/mild	12+	11
	Frequent/intense	17+	12
gambling/contests	—	17+	13
simulated gambling	—	12+	14
treatment-focused content	Infrequent/mild	12+	15
	Frequent/intense	17+	16
unrestricted web access	—	17+	17
—	—	4+	

Table 2: Google Play Maturity Rating Policy

Content	Level	Rating	#
violence	mild cartoon	low	1
	fictional violence	low	2
	realistic	medium	3
	intense fictional	medium	4
	graphic	high	5
profanity/crude humor	frequent/intense	medium	6
hate	inflammatory cont.	medium	7
sexual/suggestive	include	medium	8
	focus	high	9
alcohol/tobacco/drug	reference	medium	10
	focus	high	11
gambling themes	—	medium	12
simulated gambling	—	medium	13
location	access	low	14
	publish/share	medium	15
UGC/social	host	medium	16
	focus	medium	17
—	—	everyone	

infrequent or *mild* "realistic violence", but it is rated as *12+* if the realistic violence is *frequent* or *intense*.

Google Play's Rating Policy: Similar to App Store, Google Play's policy also consists of four maturity levels. However, unlike App Store's policy whose maturity levels are directly related to numeric ages, Google Play's four levels are *everyone*, *low maturity*, *medium maturity*, and *high maturity* [1]. Table 2 illustrates Google Play's rating policy.

Comparing the two policies: We note that the two policies are equivalent under some conditions [5], i.e., some Apps have the same maturity level under the two rating policies. For instance, an App with only frequent profanity is labeled as *12+* by the App Store's policy and *medium* by the Google Play's policy, respectively. Thus, *12+* in the App Store's policy is equivalent to *medium* in the Google Play's policy for this App. Table 3 shows the mappings between Google Play's maturity levels and the App Store's maturity levels. The conditions for such mappings were discussed and obtained by Chen et al. [5]. However, App Store and Google Play treat different content as mature. For example, Google Play adopts "hate" and "location" as mature, but these content is not considered by App Store. Moreover, Google Play also considers the intensity of mature content when labeling maturity levels, but the definition of intensity is a slightly different from that of App Store. In our experiments, we will use these conditions to obtain groundtruth mature contents and maturity levels of Google Play Apps.

2.2 Limitations of Existing Rating Policy Implementations

An implementation of a policy is to label the maturity level of an App according to the rating policy. Existing implementations adopted by App Store and Google Play are either costly or inaccurate. On App Store, Apple Inc. hires trained employees to comprehensively evaluate each submitted App to label its maturity level, and such evaluations could include meta-data (e.g., App description, icon, and screenshots of the App) analysis and code analysis. Given the large amount of new Apps, e.g., 20,000 new Apps were submitted to App Store per month as of 2013 [2], this manual labeling approach is very costly and time-consuming.

Table 3: Mappings between maturity levels in Google Play and App Store

Maturity Levels	Google Play	App Store
1	Everyone	4+
2	Low Maturity	9+
3	Medium Maturity	12+
4	High Maturity	17+

Unlike the centralized rating service provided by App Store, Google Play requires developers to label the maturity levels for their own Apps according to the Google Play's rating policy. Although Google Play's strategy is scalable and less costly, the maturity ratings reported by developers could be inaccurate. For instance, in our dataset collected from Google Play, 45% Apps have incorrect developer-provided maturity levels.

2.3 Design Goals

To overcome the limitations of existing policy implementations, we propose an automatic App maturity rating framework. We have the following design goals for such a framework.

Policy Independent: Different platforms could have different rating policies. For instance, although Google Play's policy and App Store's policy are equivalent in some cases, they are different in general. Therefore, we aim to design our framework independent of policies. In other words, the input of our framework includes the specifications of a policy, and our framework produces the maturity level of an App under this policy.

Mature Contents to Support Maturity Levels: Predicting the mature contents in an App help users better understand the App and guide developers to modify their Apps in order to increase the number of potential users. For instance, an App is labeled as *17+* by App Store because of *frequent* sexual content. The developer could decrease the number of dirty words so that it is labeled as *12+*, which makes people whose ages are between 12 and 17 years old

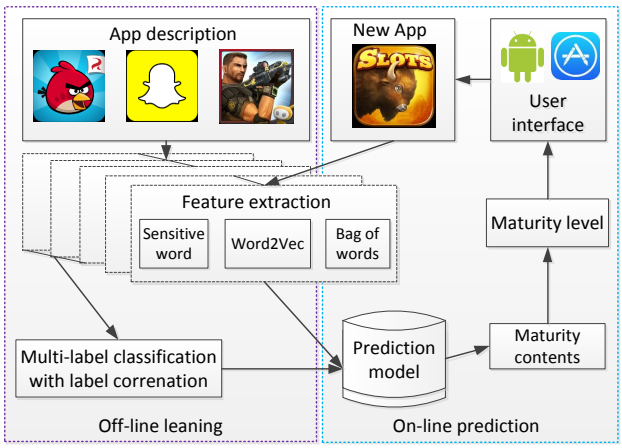


Figure 3: Overview of Automatic App Maturity Rating (AAMR) framework.

potential users of this App. Therefore, we aim to design our framework to give both the maturity levels and also the maturity words that support our rating.

Scalable and Accurate: Given the large number of new submitted Apps per month, we aim to design our framework to be scalable. In particular, our framework should produce maturity analysis results for Apps immediately after they are submitted. Moreover, our framework should produce the maturity analysis results with a high accuracy.

3. AUTOMATIC APP MATURITY RATING (AAMR) FRAMEWORK

3.1 Overview

Figure 3 illustrates the machine learning framework for automatic App maturity rating. In the off-line learning phase, we extract features from App descriptions and learn a multi-label classifier to predict the *mature content* in a given App. In the on-line prediction phase, our framework performs maturity rating in a two-stage approach, i.e., we first use the learned multi-label classifier to predict mature content in a new App and then label the maturity level according to the given rating policy.

We extract our features from *sensitive words* in maturity rating policies that directly refer to mature content, *augmented sensitive words* that are semantically similar to sensitive words, and bag-of-words model. Natural languages have synonyms, e.g., different words could represent the same mature content. Thus, we use augmented sensitive words that are synonyms of sensitive words to enrich our features. We obtain augmented sensitive words via recent *word-to-vector* techniques [24][23]. Specifically, a word-to-vector algorithm learns a vector representation for each word from a corpus of text, and two words are semantically similar if they have close vector representations. Moreover, natural languages are ambiguous, e.g., a sensitive word in an App description might not indicate that the App has mature content. Therefore, we use bag-of-words model to capture the *context* of sensitive words to mitigate the ambiguity issue.

We find that mature contents are correlated, i.e., some mature contents have very high co-occurrences in App descriptions. Therefore, we adapt multi-label Support Vector Machine (SVM) [7] to capture such correlations. We choose SVM because App descriptions are short and previous work [6] showed that SVM outperforms other classifiers for short text classification.

In the next few subsection, we will illustrate each component in more details.

3.2 Feature Engineering and Learning

We extract features from sensitive words in rating policies, augmented sensitive words that are semantically similar to sensitive words.

3.2.1 Extracting Sensitive Words

A maturity rating policy has clear definitions on maturity content. Therefore, we extract *sensitive words* from a rating policy that directly refer to mature content. For instance, the list of sensitive words we extract from the App Store's rating policy shown in Table 1 include *violence, horror, humor, profanity, sex, nudity, mature, alcohol, tobacco, drug, gambling*, and *treatment, etc.* We note that the list of sensitive words could be different for different rating policies because they could treat different contents as mature. For instance, Google Play also defines *hate* as mature content while App Store does not. We use binary features to represent sensitive words. Specifically, a feature has a value of 1 if the corresponding sensitive word appears in an App description, otherwise the feature has a value of 0.

Chen et al. [5] proposed to take these sensitive words as features and use them to directly learn a classifier to predict the maturity levels of Apps. The keyword-matching method, however, suffers from two limitations. First, natural languages have synonyms, e.g., *battle* is semantically similar to *violence*. Keyword-matching will miss these synonyms, which results in high false negatives. Second, natural languages are ambiguous, e.g., that a sensitive word appears in an App's description does not necessarily mean the App contains the corresponding mature content, which results in false positives. To address these limitations, we leverage feature augmentation to consider words that are semantically similar to sensitive words to capture the context.

3.2.2 Augmenting Features via Deep Learning

We first show an example to illustrate the issues introduced by synonyms. The following is a part of the description of an App named "Injustice: Gods Among Us":

> *Build an epic roster of DC heroes and villains and get ready for* ***battle!*** *INJUSTICE: GODS AMONG US is a free-to-play collectible card game where you build a roster of characters, moves,* ***powers****, and gear and enter the arena in touch-based 3-on-3 action* ***battle****. fight: Use the touch screen mechanics of your mobile device to* ***combat*** *your enemies in 3-on-3 action...*

In the above description, there are several words (shown in bold) strongly indicating *violence* content which is not suitable for children under a certain age. However, the sensitive words cannot represent such mature content. To address this problem, we leverage *word-to-vector (word2vec)* techniques developed by Google [24, 23] to augment sensitive words with semantically similar words.

Table 4: Top-10 words that have the highest similarities with the sensitive word *sex*.

word	cosine similarity
offend	0.4686
nudity	0.4333
intimate	0.4023
flirt	0.3612
wanted	0.3593
men	0.3326
adult	0.3323
hot	0.2974
confession	0.2898
position	0.2859

A word2vec algorithm learns a vector representation for each word from a corpus of texts in an unsupervised setting. These vector representations capture a large number of precise semantic word relationships. Specifically, two words are treated as semantically similar if they have close vector representations. As a word embedding technique, word2vec, can be viewed as a representational layer in a *deep learning* [4] architecture which transforms a word into a positional representation of the word relative to other words in the training dataset, where the position is represented as a data point in the new vector space.

In experiment, in order to get "*semantic*" meaning of words, we run the word2vec tool [32] using the app descriptions from more than 350,000 Apps in Google and App Store. For instance, Table 4 shows the top-10 words that have the highest similarities with the sensitive word *sex*. The similarity between two words is defined as the cosine similarity of the two corresponding vector representations.

For each sensitive word, we choose the top-200 words that have the highest cosine similarities, and we call these words *augmented sensitive words*. Note that some words might appear more than once in our augmented sensitive words because they might be similar to more than one sensitive word. Suppose we have n augmented sensitive words, we extract a feature vector with length n for each App. Specifically, if an augmented sensitive word appears in an App's description, the corresponding feature has a value that equals the cosine similarity between the augmented sensitive word and the corresponding sensitive word, otherwise the corresponding feature has a value of 0.

3.2.3 Bag-of-words

Given that the text data is very sparse, in some cases, App descriptions do not contain any sensitive word or augmented sensitive word. Moreover, whether a sensitive or an augmented sensitive word really indicates mature content depends on the context. Therefore, we further extract bag-of-words features from App descriptions.

We adopt *term frequency-inverse document frequency (TF-IDF)* to weight each word. TF-IDF is widely used in information retrieval and text mining [22, 30, 33]. The TF-IDF weights evaluate how important a word is to a document in a corpus of documents. Specifically, the TF-IDF weight of a word is composed by two parts. The first part computes the normalized Term Frequency (TF) and the second part is the Inverse Document Frequency (IDF). Formally, the TF-IDF weight of a word is calculated as follows, $w_{i,j} = TF_{i,j} \times log(\frac{N}{DF_i})$, where $TF_{i,j}$ is the term frequency of t_i in document d_j, N is the total number of documents in the corpus, and DF_i is the total number of documents that contain t_i.

3.2.4 Feature Concatenation

We concatenate the features extracted from sensitive words, augmented sensitive words, and bag-of-words model. For the sensitive word features, we extract twelve sensitive words from the App Store policy and thirteen sensitive words from the Google Play policy. For each sensitive word, we have the top-200 words with the highest cosine similarity. We use the most frequent 2000 words for the bag-of-words features. In total, we have 4,412 features for Apps on App Store and 4,613 features for Apps on Google Play.

3.3 Multi-label Classification

After feature extraction and feature learning from apps' descriptions, the next step is to build a machine learning classifier that can automatically classify an app into its corresponding maturity level.

More formally, let $\mathbf{x}_i \in \Re^p$ be the p-dimensional feature vector carried by each App i. Let $\mathbf{y}_i \in \{0,1\}^C$ be the C-dimensional binary vector denoting the maturity contents for App i. Then $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_n]$ corresponds to the maturity contents for all the apps. Furthermore, let $\mathbf{z} \in \Re^n$ be the set of the maturity levels of the mobile Apps, where $\mathbf{z} = [z_1, z_2, ...z_n]$ and z_i is the maturity level for each App i. More specifically, the mature content is defined by a maturity rating policy. For instance, both App Store and Google Play define 17 different mature contents, which are shown in Table 1 and Table 2, respectively, and $\mathbf{y}_i$ indicates what kind of mature content among the 17 contents the App i contains.

3.3.1 Key Idea

In the off-line training phase, we have a set of training mobile App data $\mathbf{L} = (\mathbf{x}_1, \mathbf{y}_1, z_1), (\mathbf{x}_2, \mathbf{y}_2, z_2), ...(\mathbf{x}_n, \mathbf{y}_n, z_n)$ where $\mathbf{x}_i$ is a feature vector extracted from an App description, $\mathbf{y}_i$ is the binary label vector indicating the maturity content, and $\mathbf{z}_i$ is the label of the maturity level, where $1 \leq i \leq n$.

In the on-line prediction phase, we perform maturity analysis in a *two-stage* approach, i.e.,

$$\text{App description}(\mathbf{x}) \rightarrow \text{maturity content}(\mathbf{y}) \rightarrow \text{maturity level}(z),$$

where in the first stage, the maturity contents is inferred from the app feature vectors, and in the second stage, the maturity level is predicted by combining maturity content based on the rating policy. For instance, if every element in $\mathbf{y}$ is zero, then the App has the lowest maturity level (i.e., *4+* on App Store and *Everyone* on Google Play).

Note in this procedure, an app is generally assigned to *multiple* (i.e., one or more) maturity content tags. This is known as "multi-label learning" in machine learning. Fortunately, we develop a method that can automatically adapt SVM to support multi-label classification by incorporating label correlations, with minimum efforts.

3.3.2 Multi-label Classification Using Linear SVM

As is illustrated before, to support multi-label classification task, we need to develop a method that is fast, scalable, and accurate. Linear classification method is a good fit to achieve these goals given the large number of new Apps and

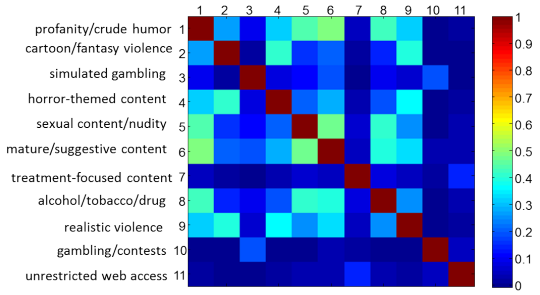


Figure 4: Correlation matrix for 11 types of maturity content, where (k, ℓ) element indicates the correlations between the tag k and ℓ. For illustration purpose, *infrequent/mild* and *frequent/intensive* are grouped together.

their meta data. Hence, in the context of app maturity level prediction, we use linear SVM as our method due to its strong capability in handling *short* text classification tasks such as [34, 29, 35, 11]. This is also further confirmed by our experiments.

Recall that in standard multi-class SVM problem, it finds the maximum-margin hyperplane [8] that has the largest separation (margin) among data points from different classes. To be exact, it optimizes:

$$\begin{aligned} \min_{\mathbf{w}_k, \xi_i} \quad & \frac{1}{2}\sum_{k=1}^{K} \mathbf{w}_k^T \mathbf{w}_k + C\sum_{i=1}^{n} \xi_i \\ s.t. \quad & \mathbf{w}_{y_i}^T \mathbf{x}_i - \mathbf{w}_k^T \mathbf{x}_i \geq 1 - \xi_i, \\ & \sum_{i=1}^{n} \xi_i \leq C, \;\; \xi_i \geq 0, \end{aligned} \tag{1}$$

where $\mathbf{w}_k$ is the decision hyperplane for k-th class ($1 \leq k \leq K$), y_i is the label for data $\mathbf{x}_i$, ξ_i is the slack variable, and C is the constant. According to "Loss + Regularization" format, Eq.(1) can be written as a sum over of a hinge loss and ℓ_2 regularization, *i.e.,*

$$\min_{\mathbf{W}} \sum_{i=1}^{n} (1 - \mathbf{w}_{y_i}^T \mathbf{x}_i + \max_{k \neq y_i} \mathbf{w}_k^T \mathbf{x}_i)_+ + \alpha \sum_{k=1}^{K} \|\mathbf{w}_k\|_2^2, \tag{2}$$

where $(x)_+ = \max\{x, 0\}$, $\mathbf{W} = [\mathbf{w}_1 \mathbf{w}_2 \cdots \mathbf{w}_K]$ is the hyperplane matrix. Let $\Omega(\mathbf{W}) = \sum_{k=1}^{K} \|\mathbf{w}_k\|_2^2$, Eq.(2) is equivalent to:

$$\min_{\mathbf{W}} f(\mathbf{W}; \mathbf{X}, \mathbf{Y}) + \alpha\Omega(\mathbf{W}), \tag{3}$$

where $f(\mathbf{W}; \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} (1 - \mathbf{w}_{y_i}^T \mathbf{x}_i + \max_{k \neq y_i} \mathbf{w}_k^T \mathbf{x}_i)_+$.

3.3.3 Adapting SVM for Label Correlations

To support multi-label classification, the idea is to transform the multi-label classification problem into multi-class problem by considering label correlations. The key observation is that many tags always co-occur together. The intuition is that the elimination of label correlations can help improve the performance, which is also confirmed in past researches [9][36][19].

In the context of maturity content rating, the two rating policies, which are shown in Table 1 and Table 2, both consider seventeen maturity content for maturity rating. Figure 4 shows the correlations among the maturity content considered by the App Store's rating policy. The figure was plotted based on the statistics from the real-world app dataset used in experiments. For illustration purpose, we combine the *infrequent/mild* and *frequent/intense* levels for the same maturity content together. However, we still treat the *infrequent/mild* and *frequent/intense* as two labels in our experiments.

We observe that there exists high correlations among some maturity content. For example, the *profanity or crude humor* and the *mature/suggestive themes* are highly positively correlated. Another example is that the *cartoon or fantasy violence* and *horror/fear theme* are highly positively correlated. In addition to the highly positive correlations, some maturity content are negatively correlated. In particular, the two levels of a maturity content are exclusive (i.e., perfectly negatively correlated). For instance, for the App shown in Figure 1.*left*, one of its maturity content is ***infrequent/mild*** *sexual content and nudity*, and thus ***frequent/intense*** *sexual content and nudity* will not be a maturity content of this App. The above observations motivate us to utilize correlations between maturity content to improve the accuracy of classification.

We adapt the standard multi-label SVM to capture label correlations. In particular, we use *pearson correlation coefficient* $\mathbf{R} = [R_{k\ell}] \in \Re^{K \times K}$ to capture the label co-occurrence, *i.e.,*

$$R_{k\ell} = \frac{\sum_{i=1}^{n} (Y_{ik} - \overline{Y_{:k}})(Y_{i\ell} - \overline{Y_{:\ell}})}{\sqrt{\sum_{i=1}^{n} (Y_{ik} - \overline{Y_{:k}})^2} \sqrt{\sum_{i=1}^{n} (Y_{i\ell} - \overline{Y_{:\ell}})^2}}, \;\; 1 \leq k, \ell \leq K, \tag{4}$$

where $\mathbf{Y} \in \Re^{n \times K}$ is the class label matrix for maturity content, and

$$\overline{Y_{:k}} = \frac{1}{n}\sum_{i=1}^{n} Y_{ik}, \quad \overline{Y_{:\ell}} = \frac{1}{n}\sum_{i=1}^{n} Y_{i\ell}.$$

The label correlation matrix $\mathbf{R}$ measures the linear correlation (dependence) between each pair of classes k and ℓ, each entry of which is a number in the interval [-1, +1]. Positive number indicates positive correlation, 0 indicates no correlation, and negative indicates negative correlation. We would like to emphasize here that without the centering of the values corresponding to each label, it is impossible to capture the negative correlations.

Then the class label matrix is modified to:

$$\tilde{\mathbf{Y}} = \mathbf{Y}\mathbf{R}, \tag{5}$$

and finally we solve the following optimization problem Eq.(6) using the same method as in Eq. (3):

$$\min_{\mathbf{W}} f(\mathbf{W}; \mathbf{X}, \tilde{\mathbf{Y}}) + \alpha\Omega(\mathbf{W}). \tag{6}$$

Please note that our method can be applied to any generic loss functions such as logistic loss, LASSO, *etc*, but not limited to hinge loss shown in Eq.(6). The *pearson correlation* is indeed a generic method to eliminate both the label correlations and the feature correlations, which can be easily adapted to solve many other correlation problems emerged in data mining and machine learning communities.

Here, we give an example to illustrate how $\mathbf{Y}$ looks like. Suppose we have the class label corresponding to app i, $Y_{i:} = [0, 1, 0, ...0, 0, 1]$. After multiplying the correlation matrix $\mathbf{R} \in \Re^{k \times k}$, $Y_{i:}$ becomes $\tilde{Y}_{i:} = [0.1, 0.75, 0.6...0.2, 0.5, -0.9]$. Using the threshold calibration in [36], we obtain a threshold t in the training phase for mapping the continuous $\tilde{Y}_{ik}$ to

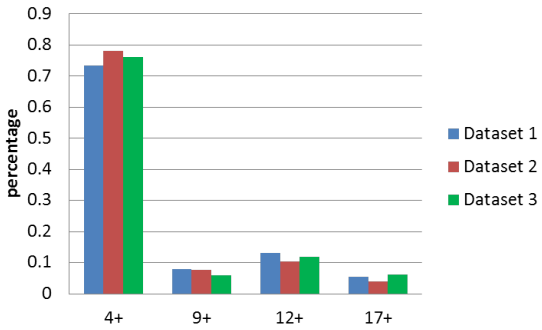


Figure 5: The distribution of ground truth maturity levels of Apps in our three datasets.

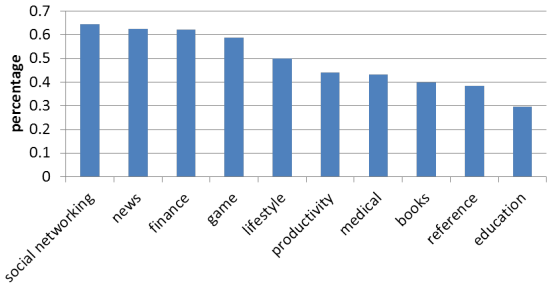


Figure 6: Top-10 categories of Google Play Apps in the Dataset 3 that have the most incorrect developer-provided maturity levels.

the binary label. In particular, if $\tilde{Y}_{ik} \leq t$, then $\tilde{Y}_{ik}$ is 0, otherwise $\tilde{Y}_{ik}$ is 1. In the training phase, we obtain $\mathbf{W}$ via solving Eq.(6) with $\mathbf{X}_{train}$ and $\tilde{\mathbf{Y}}_{train}$. In the testing phase, we first get $\tilde{\mathbf{Y}}_{test}$ with $\mathbf{W}$ and $\mathbf{X}_{test}$, and then we use t to map $\tilde{\mathbf{Y}}_{test}$ to $\mathbf{Y}$. In our experiments, we use the widely used package LibSVM [6] to implement adapted multi-label SVM classification with label correlations.

4. EXPERIMENTS

Recall that our automatic App maturity rating system not only predicts the maturity level for a given App but also provides the reasons why the App has the specific maturity level. Therefore, we evaluate both the maturity level prediction performance and the reason prediction performance. In the following, we introduce the datasets that we collected from both App Store and Google Play, evaluation metrics we adopt, training and testing, and compared approaches.

4.1 Experimental Setup

4.1.1 Data Collection

Crawling App Store and Google Play: We wrote crawlers in python to collect 105,108 free iOS Apps and 105,287 paid iOS Apps from App Store, and 261, 947 Android Apps from Google Play. Our crawls include the description and maturity level of each App. App Store also labels the mature contents that makes an App be rated as a specific maturity level. So we also crawl the mature contents for App Store Apps. We crawled our datasets between July 2014 and September 2014.

Obtaining groundtruth: Apple Inc. hires trained employees to manually perform maturity analysis for each submitted App. Therefore, for Apps from App Store, we take the maturity levels and the associated mature contents crawled from App Store as their groundtruth. Google Play Apps are rated by App developers, so their labels are not accurate. Chen et al. [5] characterized the conditions when the App Store's maturity rating policy and the Google Play's policy are equivalent, and we leverage their results to locate Apps whose Android version and iOS version have the same maturity levels and mature contents. Then, we take the maturity levels and mature contents of the iOS version as the groundtruth of the corresponding Android Apps. In summary, we have the following three datasets with groundtruth maturity information:

- **Dataset 1** consists of 105,108 free Apps in App Store.
- **Dataset 2** consists of 105,287 paid Apps in App Store.
- **Dataset 3** consists of 14,000 Apps in Google Play.

We distinguish between free Apps and paid Apps for App Store because we find that predicting their maturity levels achieves substantially different performances. However, most of Google Play Apps that have groundtruth maturity information are free, and thus we do not further classify them into free Apps and paid Apps.

Figure 5 shows the distribution of groudtruth maturity levels of Apps in our three datasets. We observe imbalanced distributions. For instance, 73% of Apps in the Dataset 1 have a maturity level of 4+, but only 5% of Apps in the Dataset 1 have a maturity level of 17+.

Unreliable Google Play ratings: A Google Play App is said to have an inaccurate maturity level if the maturity level provided by the developer does not match the groundtruth. Overall, we find that 45% of Apps in the Dataset 3 have inaccurate maturity levels. Figure 6 shows the fraction of Google Play Apps in the Dataset 3 that have inaccurate developer-provided maturity levels for top-10 App categories. We observe that Apps in the *social networking* category are most likely to be incorrectly rated by developers.

4.1.2 Evaluation Metrics

Since we classify both the maturity levels and the associated reasons, we evaluate various approaches in two aspects: 1) mature content classification, which is a multi-label task, and 2) maturity level classification, which is a multi-class task. Due to the extremely imbalanced label distributions as we show in Figure 5, we do not compare the algorithms in terms of accuracy. For instance, a classifier that always predicts 4+ as the maturity level can already achieve accuracies of more than 0.70.

Mature content classification: The mature content prediction is a multi-label classification problem. Thus, we adopt the metrics Precision, Recall, and F1-value [37], which are widely used to evaluate multi-label classification systems. Specifically, we denote by C the labels (possible mature contents). For each label $\ell \in C$, we denote by TP_ℓ, FP_ℓ, TN_ℓ, FN_ℓ the number of true positives, false positives, true negatives, and false negatives, respectively. Then, for each label $\ell \in C$, we have its precision, recall, and F1-value as $P_\ell = \frac{TP_\ell}{TP_\ell+FP_\ell}$, $R_\ell = \frac{TP_\ell}{TP_\ell+FN_\ell}$, and $F_\ell = \frac{2P_\ell R_\ell}{P_\ell+R_\ell}$, respectively. Then we compute the overall Precision, Recall, and F1-value by averaging the precisions, recalls, and F1-values over all labels, respectively.

Maturity level prediction: For maturity level prediction, we also use Precision, Recall, and F1-value as our evaluation metrics. More formally, let $TP_k, FP_k, TN_k, and FN_k$ be true positives, false positives, true negatives, and false negatives for Apps with maturity level k, respectively. Then for each level k, we have precision as $P_k = \frac{TP_k}{TP_k+FP_k}$, recall as $R_k = \frac{TP_k}{TP_k+FN_k}$, and F1-value as $F1_k = \frac{2P_kR_k}{P_k+R_k}$. The overall Precision, Recall, and F1-value are computed by averaging over all the maturity levels.

4.1.3 Training and Testing

For each of the three datasets, we sample 50% of it uniformly at random and treat them as the training data, and the rest of it is treated as the testing data. We repeat the experiments for 10 trials and average all metrics over them.[1]

4.1.4 Compared Approaches

We describe compared approaches for mature content prediction and maturity level prediction separately.

Mature content prediction: Recall that we extract features from sensitive words, augmented sensitive words, and bag-of-words model. Moreover, we consider label correlations. We aim to study the impact of each part. To this end, we add each part to our framework incrementally. Specifically, we compare the following methods:

- **AAMR-I**: Our framework AAMR with only features from bag-of-words. Label correlations are not considered.
- **AAMR-II**: Our framework AAMR with features from bag-of-words and sensitive words. Label correlations are not considered.
- **AAMR-III**: Our framework AAMR with features from sensitive words, augmented sensitive words, and bag-of-words model. Label correlations are not considered.
- **AAMR-IV**: Our framework AAMR with features from sensitive words, augmented sensitive words, and bag-of-words model. Label correlations are considered.

Maturity level prediction: We compare the following approaches to perform maturity level predictions:

- **Human Labeling (HL):** We asked 3 users (they are our colleagues) to manually label an App. Given a maturity rating policy, the users rated an App based on their experiences and their understanding of the App description. The ratings of the 3 users are aggregated in a majority voting way to get the final maturity level of the App. If majority voting does not agree upon a maturity level, we don't consider the App any more.
- **Developer Report (DR):** Google Play requires developers to report the maturity levels of their developed apps. For Apps in the Dataset 3, we compute the evaluation metrics for the ratings reported by App developers.
- **ALM [5]:** To the best of our knowledge, only Chen et al. [5] studied automatic App maturity level prediction. So we will adopt their method ALM as one baseline for comparisons. Note that the ALM method does not predict maturity contents, and thus we do not compare ALM with mature content prediction approaches.
- **Multi-Class Classification (MCC):** The multi-class classification method learns a multi-class classifier which directly maps our features extracted from App descriptions to the maturity levels. We use linear multi-class SVM as the classifier. Note that this method cannot identify mature contents in an App.
- **AAMR**: Our two-stage approach first predicts the mature contents in an App using adapted multi-label SVM with label correlations, and then labels the maturity level according to the rating policy.

[1]We find that the standard deviations of our metrics over the 10 trials are very small, and thus we do not show them for simplicity.

Table 5: Comparisons against human-based maturity rating approaches.

dataset	HL	DR	AAMR
Precision	0.43	0.68	0.77
Recall	0.34	0.57	0.76
F1_value	0.38	0.62	0.76

4.2 Results

4.2.1 Mature Content Prediction

Figure 7 shows the Precision, Recall, and F-value of the four methods AAMR-I, AAMR-I, AAMR-III, and AAMR-IV on the three datasets. We observe that feature augmentation, bag-of-words feature, and label correlation all improve mature content prediction.

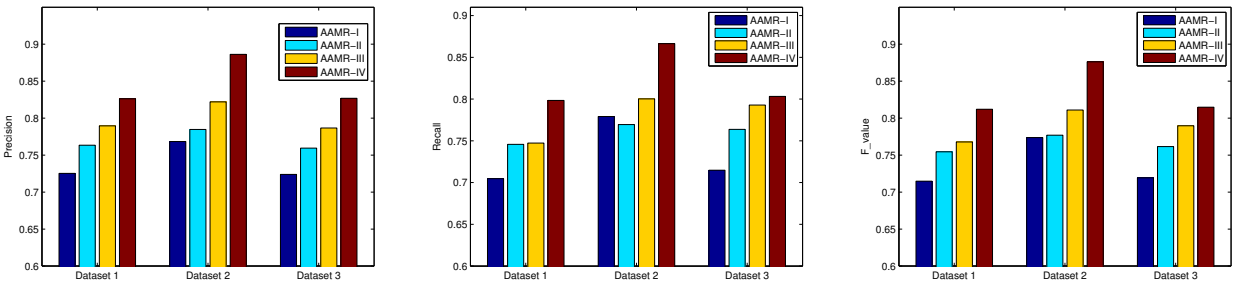
4.2.2 Maturity Level Prediction

We report results for automatic maturity level prediction approaches and human labelling approaches separately.

Comparing automatic prediction approaches: Figure 8 shows the maturity level prediction performances among the three automatic approaches, i.e., ALM, MCC, and AAMR. We observe that both MCC and AAMR achieve much better performances than ALM. Specifically, MCC achieves around 0.44 larger F-value than ALM on average, and AAMR achieves 0.39 larger F-value than ALM on average. These observations indicate that our features are much better than sensitive words which are used by ALM. MCC achieves slightly better performances than our two-stage AAMR method. This is because our AAMR makes some incorrect predictions about mature contents, which are subsequently used to label maturity levels. The two-stage approach enlarges the impact of the incorrect mature content predictions. However, MCC method cannot identify the mature contents in an App.

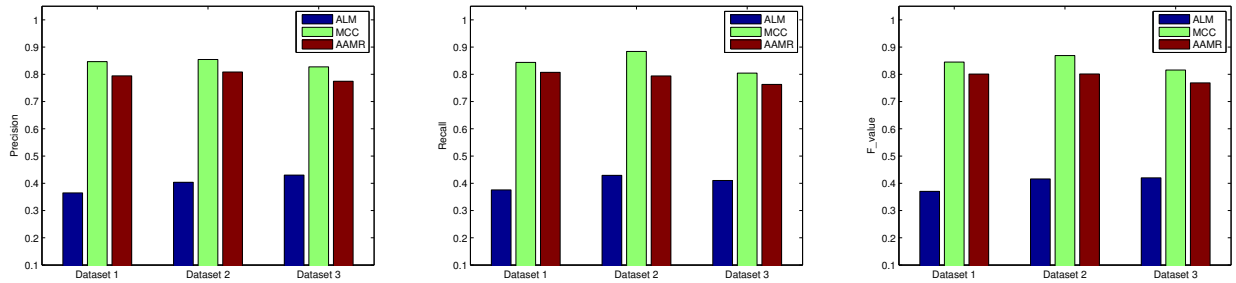
Comparing with human-based methods: We compare our method AAMR with two human-based manually labelling approaches, i.e., HL and DR. Due to the limited human resources we have, we sample 500 Apps from the Dataset 3. For each sampled App, we ask three users (our colleagues) to rate the maturity level via reading the App description. The final maturity level of an App is determined by majority voting among the labels of the three users. If the three human labels do not agree upon a maturity level, we do not consider the App. After majority voting, we obtained 441 Apps that have agreed human labels. We also predict the maturity levels for these Apps using our learned AAMR model.

Table 5 shows the Precision, Recall, and F_value of human labelling (HL), developer report (DR), and our proposed automatic method AAMR. We observe that our method AAMR achieves much more accurate results than DR (14%



(a) Precision **(b)** Recall **(c)** F1_value

Figure 7: Impact of feature augmentation, bag-of-words feature, and label correlation. We find that these parts are complementary, i.e., adding each part incrementally improves the performance of our framework.



(a) Precision **(b)** Recall **(c)** F1_value

Figure 8: Results for the three compared automatic maturity level prediction approaches. We observe that MCC and AAMR substantially outperform ALM, which indicates the effectiveness of our novel features.

improvement in terms of F_value) and HL (38% improvement in terms of F_value). Human users can hardly achieve satisfactory accuracy in rating maturity levels. In our experiments, human users only achieve 38% F_values. We speculate the reason is that developers mainly describe functionality in App descriptions, and humans might not be able to correlate non-sensitive words to maturity levels. Moreover, DR achieves better performances than HL. We speculate the reason is that App developers have better understanding about their Apps and thus could provide more accurate maturity levels.

We further study how users and Android developers label maturity levels incorrectly. An App is *underrated* if the provided maturity level is lower than its groundtruth, otherwise an App is *overrated*. We find that App developers are more likely to underrate the maturity levels. Specifically, around 80% of incorrect labels provided by App developers are underrated while 46% of users-provided incorrect labels are underrated. We speculate the reason is that App developers underrated their Apps so that more people could become their users.

4.2.3 Summary

We demonstrate that feature learning (augmentation), bag-of-words features, and label correlations are all necessary, i.e., incorporating them increases the performances of our framework. Moreover, our approach substantially outperforms previous automatic approaches and human-based manual labelling approaches.

5. RELATED WORK

We review related work on mobile App maturity rating, text analysis, and other App-related studies.

Maturity rating: To the best of our knowledge, little research has been conducted for App maturity rating. Chen et al. [5] studied the severity of unreliable maturity ratings for mobile Apps on Google Play. By comparing the maturity ratings of such Apps on Google Play and App Store, they measured the severity of inaccurate maturity ratings of Google Play apps. The reason is that Google Play requires developers themselves to rate their own Apps and developers tend to underclaim the maturity level in order to attract a boarder range of users.

Text analysis: The bag-of-words model was first documented by Harris [14] and has been widely used for documentation classification [17]. Google developed the word to vector technique [23, 24], which learns a vector representation for each word to capture the syntactic and semantic word relationships. The algorithm takes a text corpus as input and produces the word vectors as output. Fan et al. [11] demonstrated that linear SVM [7] outperforms other classifiers for short text classification. Kong et al. [20] predicted the permission required by mobile apps from app descriptions using structure feature learning technique.

Other App-related studies: Many recent app-related work focus on security and privacy issues. Specifically, they either reveal the potential security risks in the Android platform [12, 3, 18] or enhance the overall Android security via

retrofitting the android platform or adding more features into it [26, 10, 31]. In addition, users privacy preference can also be utilized in personalized mobile app recommendation [21]. These work are orthogonal to ours.

6. CONCLUSION AND FUTURE WORK

The *scope* of this work is to propose a framework that automatically predicts mature contents and maturity levels for mobile Apps from app descriptions. We map the mature content prediction to a multi-label classification problem and then use the predicted mature contents to label the maturity levels. First, we extract novel features from App descriptions using deep learning techniques by considering the semantics of words. Second, we adapt SVM to capture label correlations in a multi-label setting. The experiment results on real-world datasets demonstrate that our approach can achieve relatively high accuracies with only App descriptions, and that our approach substantially outperforms baseline methods,.

A few interesting directions include incorporating more features from information sources such as user comments, UI screenshots, and dynamic running behaviors of Apps into our framework, which makes our system more robust to attacks such as app description obfuscations, as well as rating maturity levels of dynamic contents such as advertisements in Apps.

Acknowledgment: We thank reviewers for their valuable comments.

7. REFERENCES

[1] Andriod App Ratings 2014. `https://support.google.com/googleplay/android-developer/answer/188189`, 2014.
[2] Apple Store new Apps. http://goo.gl/ojrhp2.
[3] L. Cen, D. Kong, H. Jin, and L. Si. Mobile app security risk assessment: A crowdsourcing ranking approach from user comments. In *SDM*, pages 658–666, 2015.
[4] H. Chang, Y. Zhou, P. Spellman, and B. Parvin. Stacked predictive sparse coding for classification of distinct regions in tumor histopathology. In *ICCV*, December 2013.
[5] Y. Chen, H. Xu, Y. Zhou, and S. Zhu. Is this app safe for children?: A comparison study of maturity ratings on android and ios applications. WWW '13, pages 201–212, 2013.
[6] C.-J. L. Chih-Chung Chang. Libsvm: A library for support vector machines. In *ACM Transactions on Intelligent Systems and Technology*, 2011.
[7] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.
[8] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2, Mar. 2002.
[9] A. C. P. L. F. de Carvalho and A. A. Freitas. A tutorial on multi-label classification techniques. In *Foundations of Computational Intelligence (5)*, volume 205. 2009.
[10] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. S. Wallach. Quire: Lightweight provenance for smart phone operating systems. In *Usenix Security*, 2011.
[11] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008.
[12] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: Attacks and defenses. In *Usenix Security*, 2011.
[13] D. M. Fergusson, J. M. Boden, and L. J. Horwood. Exposure to childhood sexual and physical abuse and adjustment in early adulthood. *Child abuse & neglect*, 32(6), 2008.
[14] Z. S. Harris. Distributional structure. *Word*, 1954.
[15] Inappropriate content making its way to mobile apps. http://goo.gl/dqi1cg.
[16] iOS App Ratings 2014. `https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/iTunesConnect_Guide.pdf`, 2014.
[17] T. Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002.
[18] D. Kong, L. Cen, and H. Jin. AUTOREB: Automatically understanding the review-to-behavior fidelity in android applications. In *CCS*, 2015.
[19] D. Kong, C. H. Q. Ding, H. Huang, and H. Zhao. Multi-label relieff and f-statistic feature selections for image annotation. In *CVPR*, pages 2352–2359, 2012.
[20] D. Kong and H. Jin. Towards permission prediction on mobile apps via structure feature learning. In *SDM*, pages 604–612, 2015.
[21] B. Liu, D. Kong, L. Cen, N. Z. Gong, H. Jin, and H. Xiong. Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. In *WSDM*, pages 315–324, 2015.
[22] C. Manning and H. Schutze. Foundation of statistical natural language processing. 1999.
[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
[25] Mobile Apps for Kids: Current Privacy Disclosures are Disappointing. http://goo.gl/xr1udt.
[26] D. Octeau, P. McDaniel, S. Jha, A. Bartel, E. Bodden, J. Klein, and Y. L. Traon. Effective inter-component communication mapping in android withepicc: An essential step towards holistic security analysis. In *Usenix Security*, 2013.
[27] R. E. O'Hara, F. X. Gibbons, M. Gerrard, Z. Li, and J. D. Sargent. Greater exposure to sexual content in popular movies predicts earlier sexual debut and increased sexual risk taking. *Psychological science*, 23(9), 2012.
[28] Overexposed and Under-Prepared: The Effects of Early Exposure to Sexual Content. http://goo.gl/x2whgr.
[29] A. Sadilek, S. P. Brennan, H. A. Kautz, and V. Silenzio. nemesis: Which restaurants should you avoid today? In *HCOMP 2013*, 2013.
[30] K. Spärck-Jones. A statistical interpretation of term specificity and its application in retrieval. pages 11–21, 1972.
[31] N. Wang, B. Zhang, B. Liu, and H. Jin. Investigating effects of control and ads awareness on android users' privacy behaviors and perceptions. In *MobileHCI 2015*. ACM, 2015.
[32] Word2Vec Tool for Computing Continuous Distributed Representations of Words. `https://code.google.com/p/word2vec/`.
[33] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML 1997*, 1997.
[34] H.-F. Yu, C.-H. Ho, P. Arunachalam, M. Somaiya, and C.-J. Lin. Product title classification versus text classification. 2012.
[35] G. Yuan, C. Ho, and C. Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100(9):2584–2603, 2012.
[36] M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *IEEE TKDE*, 26(8), 2014.
[37] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 26(8):1819–1837, Aug 2014.