Macro-scale Mobile App Market Analysis using Customized Hierarchical Categorization

Xi Liu  
Michigan State University  
liuxi4@cse.msu.edu

Han Hee Song  
Cisco  
hansong@cisco.com

Mario Baldi  
Cisco & Politecnico di Torino  
mariobal@cisco.com

Pang-Ning Tan  
Michigan State University  
ptan@cse.msu.edu

Abstract—Thanks to the widespread use of smart devices, recent years have witnessed the proliferation of mobile apps available on online stores such as Apple iTunes and Google Play. As the number of new mobile apps continues to grow at a rapid pace, automatic classification of the apps has become an increasingly important problem to facilitate browsing, searching, and recommending them. This paper presents a framework that automatically labels apps with a richer and more detailed categorization and uses the labeled apps to study the app market. Leveraging a fine-grained, hierarchical ontology as a guide, we developed a framework not only to label the apps with fine-grained categorical information but also to induce a customized class hierarchy optimized for mobile app classification. With the classification accuracy of 93%, large-scale categorization conducted with our framework on 168,000 Google Play apps discovers novel inter-class relationships among categories of Google Play market.

I. INTRODUCTION

Smartphones, tablets and other mobile devices have transformed the way we communicate, manage, and perform our daily tasks. A key driver to the adoption of these devices is the proliferation of mobile apps developed for personal use, business, education, and other purposes. App markets, such as Google Play and Apple iTunes, have provided a convenient platform for developers to market their creations and for users to search and download new apps. However, with over three million existing apps and 600,000 new apps being introduced to the markets every year [1], organizing and managing them has become a huge challenge.

A key step towards organizing mobile apps is to assign appropriate categories to each of them. By grouping apps that are closely related to each other, categorization can help facilitate browsing, searching, and recommending apps to accommodate the greatly varying taste of users. While each app market provides its own list of app categories, there are several issues with the existing market categorizations.

Firstly, existing categorizations have limited granularity. The Google Play market, for example, provides a flat categorization of 25 class labels, with an exception of the Games category, which has 6 subclasses. The Apple iTunes store employs a similar, largely flat categorization as well. While the number of categories has remained mostly unchanged since the introduction of app markets, the explosive growth of mobile apps has resulted in an average of about 15,000 apps per category, rendering the task of searching for an appropriate app in a category to be laborious and time consuming. The granularity of current mobile app categorization is also too coarse to effectively distinguish an app from other apps assigned to the same category. The lack of details in existing categorization is detrimental to several analysis efforts, whether focusing on users or app markets.

The second issue with the current categorization is its lack of objectivity. The classification of mobile apps in online app stores is typically done manually, often based on the subjective judgment, personal viewpoints, and opinions of the app developers. Occasionally, the categories provided by the developers may not agree with the actual use of the apps. Given the large amount of apps on the markets, it is infeasible to impose strict guidelines or a revision process to the categorization. As a result, the apps are not guaranteed to be accurately classified and to share common, undivided semantics.

The last issue has to do with limited expressiveness. To prevent developers from overly promoting their apps by assigning them to multiple classes, markets restrict apps to be exclusively in a single category. As pointed out in [2], the hard, exclusive labeling results in a large number of multi-category apps missing from their appropriate categories. For example, the Instagram app [3] can be found only in the social networking category but not in the photography category, even though it has functionalities related to both categories.

While the app markets strive to provide the most generic categorization, the current market categorization limits its utility in catering the needs of different use cases requiring different viewpoints. For example, to precisely target user groups, app recommendation needs fine-grained information on functional similarity among apps (e.g., separating air travel apps from bus and rail; currently all in travel category). On the other hand, to prioritize traffic and provision the network, corporate network management focuses on separating business apps from personal-use apps (e.g., separating Cisco Jabber from Facebook messenger, both in communicator category).

Having a well-designed customized categorization will flexibly accommodate different taxonomic relations and in turn, will greatly improve our ability to learn what an app is about for the purpose of app market analysis and user interest research. To overcome these limitations, in this paper, we propose a framework that automatically labels apps with a richer and more flexible categorization and use the labeled apps to study the app markets.
In order to label apps with a finer-grained ontology than the original categories the app market provides, we leverage a detailed categorization from an application domain that closely resembles that of mobile apps (e.g., Google Ad Preference ontology [4]). Unlike the flat structure of the original market categorization, our proposed framework is hierarchical so that both the generality and specificity between classes are encoded in ancestor-successor node relationships, while classes with comparable level of generality are encoded in sibling relationships in the class hierarchy. Additionally, to accommodate apps with properties spanning over different classes, our framework provides multi-class labeling so that such apps can be accurately labeled. Our categorization framework employs a semi-supervised non-negative matrix factorization approach to simultaneously learn (i) the categories of unlabeled apps and (ii) the inter-class relationships. A hierarchical clustering algorithm is then applied to the estimated inter-class relationship matrix to induce the class hierarchy.

While the finer-grained categorization should be carefully chosen to overcome the limitation of original market categories, it is not guaranteed to perfectly represent mobile apps. Because its taxonomy is not designed for mobile apps, even with an extensive sub-categorization, the fine-grained classes might not cover all possible concepts and aspects of mobile apps. Similarly, some classes might not be independently associate with the apps nor mutually exclusive of each other. In order to narrow the gap between the apps and the pre-built input categorization, our framework re-organizes the inter-class structures of the input categorization and outputs an acclimatized class relationship customized to the mobile apps.

We evaluate the classification accuracy of our framework on Google Play market. With 1,065 manually labeled ground truth, our framework with soft (multi-class) classification achieves the maximum accuracy of 93.0% with 100% coverage whereas a hard classification with Logistic regression has 81.4% accuracy at the same coverage level. We also evaluated the customized inter-class relationship induced by our framework. Given Google Ad preference tree [4] as an instance of the fine-grained input categorization, the framework makes an interesting suggestion on building another layer of hierarchy by grouping subset of input classes. The finding is corroborated with a business market research on mobile apps. Even though the discussion here focuses on Android apps, our methodology is also applicable to apps from Apple iTunes.

### II. BACKGROUND

To overcome the limitations of existing categorizations provided by app markets, we propose a framework to relabel the apps into finer categories. The framework considers the following sources of information as input: (i) app related information that can be used to derive the attributes for classification, (ii) a new set of fine-grained custom categories obtained from an external source, and (iii) a method to reclassify a subset of the apps into the custom categories. The re-labeled apps along with their corresponding features will be used to create a training set for building our app classifier. The sources of information along with their characteristics will be described in further details in the following subsections. Even though the discussion here focuses on Android apps, our methodology is also applicable to apps from Apple iTunes.

#### A. Metadata from Google Play Market

We wrote a script program to collect information on 466,082 mobile apps from the Google Play Market [5] using its Web API. The API enables us to gather information about each app, including the app ID, title, description, and original market category assigned by the developers. The app title and description were used to derive the textual features for classifying the apps. Among all the mobile apps we collected, about 20% of the apps either have no description or are written in a language other than English. After removing such apps, we were left with 374,182 apps. Each of the apps retrieved using Google Play API was assigned to one of 25 market categories by the app developers (see Table I). Most of the market categories do not have sub-categories with the exception of Games, which has a two-level categorization.

<table>
<thead>
<tr>
<th>Original Google Play Market Categorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book &amp; Reference</td>
</tr>
<tr>
<td>Media &amp; Video</td>
</tr>
<tr>
<td>Travel &amp; Local</td>
</tr>
<tr>
<td>Music &amp; Audio</td>
</tr>
<tr>
<td>Health &amp; Fitness</td>
</tr>
<tr>
<td>News &amp; Magazines</td>
</tr>
</tbody>
</table>

#### B. Fine-Grained Categorization

Previous studies on document categorization have shown that acclimatizing a pre-built ontology by human experts is more effective in real-world applications than building categorizations from scratch [6]. In our research, we use the categories that appear in the Google Ad tree obtained from the Google Ads Preference Manager [4] as our input custom categorization. The tree is originally designed to help online advertisers to setup targeted ad campaigns based on browsing activities of Google and its licensed service users. The tree organizes user interests into a hierarchical structure of up to seven levels of depth. Because many (but not all) of today’s mobile apps are device-specific versions of online services, the tree can serve as a good example of finer-grained, customized categorization for the apps.

The scale of the tree, however, poses a challenge in our study, specifically, in the validation of our results. While our algorithm has no restriction in the size of input categorization, finding enough amount of apps for ground truth labeling onto a specific category (e.g., Chrysler, Women’s apparel, Gymnastics) was not financially feasible (Section II-C explains our method for ground truth building). For the purpose of research, therefore, we resorted to use 82 categories (10
intermediate and 72 leaf nodes) from the top two levels (for the above three example categories, we use their parent categories of Cars, Shopping, Individual sports, respectively). While the tree has been dramatically pruned from the original one, it still is twice larger than the original market categories.

Figure 1 shows a snippet of the tree structure. Notice that some intermediate nodes are semantically equivalent to classes in the original Google Play market category (e.g., News). However, their sub-categories further divides the apps into finer-grained classes (e.g., Weather, Shopping search, and Gossip & tabloid). A complete listing of the categories is shown in Table II.

![Google Ad Tree](image)

**Fig. 1. A snippet of a custom input categorization**

### TABLE II
FINE-GRAINED CUSTOM CATEGORIZATION FROM GOOGLE AD TREE.

<table>
<thead>
<tr>
<th>Level 1 Category Label</th>
<th>Level 2 Category Label</th>
<th>Level 1 Category Label</th>
<th>Level 2 Category Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arts &amp; Entertainment</td>
<td>News</td>
<td>Games</td>
<td>News</td>
</tr>
<tr>
<td>Music &amp; Audio</td>
<td>Shopping search</td>
<td>Beauty &amp; Fitness</td>
<td>Shopping search</td>
</tr>
<tr>
<td>TV &amp; Video</td>
<td>Online communities</td>
<td>Computers &amp; Electronics</td>
<td>Online communities</td>
</tr>
<tr>
<td>Comedy</td>
<td>Business News</td>
<td>Travel</td>
<td>Business News</td>
</tr>
<tr>
<td>Movies</td>
<td>Technology News</td>
<td>Travel</td>
<td>Technology News</td>
</tr>
<tr>
<td>Comedy</td>
<td>Health News</td>
<td>Travel</td>
<td>Health News</td>
</tr>
<tr>
<td>TV &amp; Video</td>
<td>Dance &amp; Performing Arts</td>
<td>Travel</td>
<td>Dance &amp; Performing Arts</td>
</tr>
<tr>
<td>Sports</td>
<td>Health News</td>
<td>Travel</td>
<td>Health News</td>
</tr>
<tr>
<td>Music &amp; Audio</td>
<td>Beauty &amp; Fitness</td>
<td>Travel</td>
<td>Beauty &amp; Fitness</td>
</tr>
<tr>
<td>Music &amp; Audio</td>
<td>Dating &amp; Personals</td>
<td>Travel</td>
<td>Dating &amp; Personals</td>
</tr>
<tr>
<td>TV &amp; Video</td>
<td>Entertainment</td>
<td>Travel</td>
<td>Entertainment</td>
</tr>
<tr>
<td>Comedy</td>
<td>Reference</td>
<td>Travel</td>
<td>Reference</td>
</tr>
<tr>
<td>TV &amp; Video</td>
<td>TV &amp; Video</td>
<td>Travel</td>
<td>TV &amp; Video</td>
</tr>
<tr>
<td>TV &amp; Video</td>
<td>Travel</td>
<td>Travel</td>
<td>Travel</td>
</tr>
</tbody>
</table>

![Heat map of cosine similarities among categories](image)

**Fig. 2. Heat map of cosine similarities among categories**

We randomly selected 2,514 apps from the 25 major categories and 6 sub-categories for Games for labeling by the crowd workers. Each worker must assign an app to one of the custom categories described in Section II-B. Of them, we find 1,065 apps (42%) have all three workers unanimously agree on a single label. In addition, there were 688 apps (27%) in which two of the three workers agree on a single label. For the remaining 761 apps (30%), all three workers assign different labels. To obtain the most reliable results, we primarily use the 1,065 apps with unanimous agreements for our modeling unless stated otherwise. Although there were 82 custom categories available, these 1,065 apps were assigned to only 49 categories by the workers.

Next, we verify whether the apps labeled into the same custom category share semantically common terms. To do this, we represent the textual feature of each app as a TF-IDF vector and compute their pairwise cosine similarities. Figure 2 shows the cosine similarities among keywords between every pair of the unanimously labeled apps. We represent the similarity in terms of the intensity of a heat map (lighter color being lower similarity, darker color being higher similarity). From the fact that the diagonal components being highlighted, we infer that the apps labeled in the same custom category tend to share stronger semantic similarity. For example, apps in News::Weather category share several common terms in their descriptions — temperature and thermometer. There are, however, some apps from the same category that do not share much common terms (i.e., categories with light color). These are apps from categories that have broader and more abstract meanings, e.g., Reference::General Reference. This analysis allows us to not only determine the categories in which our algorithm will not perform well, but also identifies categories with clearer, more concrete semantics.

C. Re-labeling of Apps using Crowdsourcing

To train a classifier for labeling the apps into fine-grained, custom categories, we need example mappings between the apps and their custom categories (referred to as ground truth from here on). We conducted a crowdsourcing campaign on Amazon Mechanical Turk [7] to obtain the ground truth labels. For each app, we display its title and the entire text of app description. A worker chooses a label in our custom category that she thinks is the best match to the app. To ensure the labels are reliable, we present each app to three different workers.

We randomly selected 2,514 apps from the 25 major categories and 6 sub-categories for Games for labeling by the crowd workers. Each worker must assign an app to one of the custom categories described in Section II-B. Of them, we find 1,065 apps (42%) have all three workers unanimously agree on a single label. In addition, there were 688 apps (27%) in which two of the three workers agree on a single label. For the remaining 761 apps (30%), all three workers assign different labels. To obtain the most reliable results, we primarily use the 1,065 apps with unanimous agreements for our modeling unless stated otherwise. Although there were 82 custom categories available, these 1,065 apps were assigned to only 49 categories by the workers.

Next, we verify whether the apps labeled into the same custom category share semantically common terms. To do this, we represent the textual feature of each app as a TF-IDF vector and compute their pairwise cosine similarities. Figure 2 shows the cosine similarities among keywords between every pair of the unanimously labeled apps. We represent the similarity in terms of the intensity of a heat map (lighter color being lower similarity, darker color being higher similarity). From the fact that the diagonal components being highlighted, we infer that the apps labeled in the same custom category tend to share stronger semantic similarity. For example, apps in News::Weather category share several common terms in their descriptions — temperature and thermometer. There are, however, some apps from the same category that do not share much common terms (i.e., categories with light color). These are apps from categories that have broader and more abstract meanings, e.g., Reference::General Reference. This analysis allows us to not only determine the categories in which our algorithm will not perform well, but also identifies categories with clearer, more concrete semantics.
The categories provided by app markets are often inadequate to meet the needs of various applications. This section presents a framework to create a custom classification with hierarchical, multi-class labeling. As schematically represented in Figure 3, our approach consists of the following three main steps: (i) Data collection in which we collect the following three data: app metadata from app markets, a pre-built fine-grained ontology as custom categorization, and the labels of apps onto the custom categorization using crowdsourcing, (ii) Feature extraction in which we convert unstructured natural language text into a structured feature set, and (iii) Classifier construction, in which we develop a hierarchical learning algorithm. Details of each step are given below.

A. Data collection

As shown in the top part of Figure 3, our categorization framework has three types of input: app description keywords, pre-built custom categorization, and ground truth labels. We extracted app keyword features from the app title and app description.

The custom categories from Google Ad tree (Section II-B) are used to provide finer-grained categorization than the original market categories.

Considering the semantics of the input app description keywords, our classifier modifies the taxonomy of the input categories and suggests a customized hierarchy. An equal number of apps from every market class are labeled using crowdsourcing (see Section II-C).

B. Feature extraction

The objective of the feature extraction step is to turn raw, unstructured natural language texts about apps into structured data elements and to create a feature vector for each app. Feature extraction is done in the following three sub-steps:

(i) Word tokenization, (ii) Stemming and lemmatization, (iii) Keyword extraction.

Through (i) and (ii), we translate each word in the text into \{word term, occurrence frequency\} tuples. We omit detailed description of the tokenization and lemmatization methods as we follow standard methods described in [8].

For (iii), for each app \(a_i\), we vectorize the term-frequency tuples into a frequency vector (i.e., feature vector) of length \(1 \times d\), where \(d\) is the number of all terms that appear in the entire corpus (i.e., in all app descriptions). By concatenating all the row vectors as columns, we build a matrix \(X\) of dimension \(n \times d\), where \(n\) is total app count and \(d\) is length of each feature vector.

Further, to help alleviate the ill-effect of stop words (e.g., articles and prepositions) and other uninformative words that appear frequently in the textual description of apps, we convert the values of the matrix from term frequency into well-known, Term-Frequency Inverse Document Frequency (TF.IDF) [9]. Each entry \(X_{ij}\) corresponds to the TF.IDF value of the \(j\)th term \(t_j\) of \(i\)th app \(a_i\). The \(TF.IDF\) value is computed based on the equation: \(X_{ij} = tf(a_i, t_j) \log idf(t_j, A)\), where \(tf(a_i, t_j)\) denotes the frequency of term \(t_j\) in the text description for app \(a_i\) and \(idf(t_j, A) = N/(1 + |\{a \in A : t_j \in a\}|)\) is a measure of importance of term \(t_j\) with respect to the entire app description corpus \(A\). \(|\{\cdot\}|\) denotes the cardinality of a set.

C. Classifier construction

Using both the app-keyword matrix and ground truth categorization, we develop a semi-supervised Non-negative Matrix Factorization (NMF) framework to classify the apps. Baseline NMF is a generic learning paradigm that has been successfully applied to solve a wide variety of unsupervised and supervised learning problems. The idea behind matrix factorization is to separate out latent categorical information from large input matrices by decomposing them into products of their underlying latent factors. The enforcement of non-negativity to matrix factorization is added because in document categorization problems, it is natural to represent affinity between a document and a class as a non-negative value. In Section III-C1, we explain the standard NMF. Then in Sections III-C2 and III-C3, we explain how we extended the baseline NMF to perform semi-supervision with side-information.

1) Baseline NMF: NMF was originally developed for unsupervised clustering on an input data matrix \(X \in \mathbb{R}_+^{n \times d}\) by solving the following objective function:

\[
\min_{W,L} \|X - WL^T\|_F^2 \quad \text{s.t.} \quad W \geq 0, L \geq 0
\]

where \(\|\cdot\|_F^2\) denotes Frobenius norm, \(L^T \in \mathbb{R}^{k \times d}\) is a matrix of basis vectors defined in the lower-dimensional manifold \((k < d)\), and \(W \in \mathbb{R}^{n \times k}\) is low-rank approximation that encodes cluster membership of the \(n\) apps into \(k\) clusters.

The non-negativity constraint in NMF ensures that the cluster membership matrix \(W\) is easier to interpret (in terms of degree of closeness of an app to a class). By imposing
appropriate constraints on Eq. (1), the NMF approach can be shown to be mathematically equivalent to the optimization criterion used by various algorithms including k-means and spectral clustering [10]. However, the unsupervised NMF approach does not consider the ground truth labels acquired through crowdsourcing. To overcome this problem, we present a semi-supervised NMF framework using side information from a custom categorization (e.g., Google Ad tree).

2) *Side information from custom categorization:* To help guide the app categorization with custom ground truth categories, the NMF framework can be modified to utilize side information about the classes in the input categories. Specifically, the side information can be represented as a $k \times k$ class similarity matrix $P$, where $k$ is the total number of available categories. Each entry $P_{ij}$ denotes the similarity between classes $i$ and $j$, which is obtained by checking their sibling relationship (i.e., whether $i$ and $j$ share a common parent in Google Ad Tree). Formally,

$$P_{ij} = \begin{cases} p = \frac{1}{\# \text{of levels}} & \text{if } i \text{ and } j \text{ are siblings,} \\ 0 & \text{otherwise.} \end{cases}$$ (2)

Using $P$ as a regularization constraint, our NMF framework can derive a new similarity matrix that best approximates the inter-class relationships found from the input data.

3) *NMF with semi-supervision:* The semi-supervised NMF formulation developed in this study takes into account various factors, such as the class labels associated with sampled training set apps, feature vectors of unlabeled apps, as well as the side information encoding affinity relationship between input categories. All of this information are integrated into a unified learning framework that automatically generates (i) the predicted labels for previously uncategorized apps, (ii) the important features characterizing different classes, and (iii) a modified inter-class similarity matrix that better fits to specific characteristics of mobile apps. Specifically, the semi-supervised NMF framework developed in this paper is designed to minimize the following objective function:

$$\min_{Y_u, L, B} \quad \|Y_l - X_l W l\|_F^2 + \|Y_u - X_u W u\|_F^2 + \beta \|L\|_F^2 + \gamma \|B - P\|_F^2,$$ (3)

$$\text{s.t.} \quad W = LB, \quad B \geq 0, \quad L \geq 0, \quad Y_u \geq 0$$

where the subscripts $l$ and $u$ denote the labeled and unlabeled data sets; $n_l$ and $n_u$ denote the number of labeled and unlabeled examples, respectively. $Y_l \in \mathbb{R}^{n_l \times k}$ and $Y_u \in \mathbb{R}^{n_u \times k}$ are the class indicator matrices, where $k$ is the number of classes. $X_l \in \mathbb{R}^{n_l \times d}$ and $X_u \in \mathbb{R}^{n_u \times d}$ are the feature matrices associated with the labeled and unlabeled data, respectively.

The objective function consists of four terms. The first two terms, $\|Y_l - X_l W l\|_F^2 + \|Y_u - X_u W u\|_F^2$, measure the error of the fitted classification model. Note that although a linear model is employed in this study, the formulation can be easily extended to a non-linear setting using well-known kernel trick approach [11]. For semi-supervised learning, the second term allows us to incorporate unlabeled data into the NMF formulation. The parameter matrix $W \in \mathbb{R}^{d \times k}$ is further decomposed into a product of two matrices, $L$ and $B$. $L \in \mathbb{R}^{n_l \times k}$ identifies the important features of each class and $B \in \mathbb{R}^{n_u \times k}$ is the class similarity matrix estimated from the input similarity matrix $P$. The third and fourth terms in the objective function are regularizer terms to control the model complexity. The third term $\beta \|L\|_F^2$ ensures that the approximated feature matrix $L$ is sparse and concise. The fourth term $\gamma \|B - P\|_F^2$ penalizes the model if the matrix $B$ deviates significantly from the input similarity matrix $P$ obtained from our side information.

By simultaneously learning $Y_u$, $L$, and $B$, this unified framework seeks to simultaneously generate outputs of the three objectives mentioned earlier: (i) predicted classifications for the previously uncategorized app $i$ in $\max(Y_u[i,:])$, (ii) the important features characterizing different classes in $L$, and (iii) a modified inter-class similarity matrix specific to mobile apps, encoded in the matrix $B$.

We employ an alternating coordinate descent algorithm [10] to solve the objective function. First, the objective function in Equation 3 can be simplified into the following form:

$$J = \text{Tr}[-Y_l^T X_l L B - B^T L^T X_l^T Y_l + B^T L^T X_u^T Y_u L B$$

$$- Y_u^T X_u L B - B^T L^T X_u^T Y_u + B^T L^T X_u^T X_u L B]$$

$$+ \beta \text{Tr}[L^T L] + \gamma \text{Tr}[B^T B - B^T P - P^T B] + \text{Tr}[Y_u^T Y_u] + \text{constant.}$$ (4)

By taking the partial derivative of each variable with respect to $L$, $B$, and $Y_u$, we obtain the following:

$$\frac{\partial J}{\partial L} = -2[X_l^T Y_l + X_u^T Y_u] B^T$$

$$+ 2[X_l^T X_l + X_u^T X_u] L B B^T + 2\beta L$$

$$\frac{\partial J}{\partial B} = -2L^T [X_l^T Y_l + X_u^T Y_u]$$

$$+ 2L^T [X_l^T X_l + X_u^T X_u] L B + 2\gamma (B - P)$$

$$\frac{\partial J}{\partial Y_u} = 2Y_u - 2X_u L B$$

Although the overall objective function is non-convex, it is convex with respect to minimizing each of the unknowns $Y_u$, $L$, and $B$. The standard gradient descent algorithm can be applied to iteratively update the matrices from their initial values as follows:

$$L \leftarrow L + \eta_l \frac{\partial J}{\partial L} \text{ and } B \leftarrow B + \eta_B \frac{\partial J}{\partial B}.$$

Note that $Y_u = X_u L B$ is non-negative if $X_u$, $L$, and $B$ are non-negative. To ensure non-negativity of $L$ and $B$, we transform the additive updates used in the original gradient descent to a multiplicative update by choosing appropriate learning rates, $\eta_B$ and $\eta_L$. The resulting multiplicative update formula are

$$L_{ij} \leftarrow L_{ij} \times \frac{|X_l^T Y_l B^T + X_u^T Y_u B^T|_{ij}}{|X_l^T X_l L B B^T + X_u^T X_u L B B^T + \beta L|_{ij}}$$ (5)

$$B_{ij} \leftarrow B_{ij} \times \frac{(L^T X_l^T Y_l + L^T X_u^T Y_u + \gamma P)_{ij}}{|L^T (X_l^T X_l + X_u^T X_u) L B + \beta B|_{ij}}.$$ (6)
A summary of our proposed method is given in Algorithm 1. The algorithm takes in app-term matrices of labeled data $X_l$, unlabeled data $X_u$, class indicator $Y_l$, and custom class similarity $P$. As long as $B$ and $L$ are initialized to be non-negative matrices, the resulting $B$ and $L$ will be non-negative upon convergence. Once $B$ is found, we apply the group average hierarchical clustering algorithm to infer the hierarchical relationship of the classes.

**IV. EXPERIMENTAL EVALUATION**

This section presents the experimental results of our proposed semi-supervised NMF framework. The framework is evaluated based on the following two criteria—(i) accuracy performance of its app categorization and (ii) effectiveness of its class hierarchy customization.

We begin by evaluating the accuracy and coverage of categorizing apps into finer-grained Google Ad Category Tree (Section IV-A). Here, we evaluate the performance of the semi-supervised NMF on both hard classification and soft multi-class classification. We then apply hierarchical clustering to the estimated class similarity matrix to obtain a modified class hierarchy and analyze the effectiveness of the approach by comparing the obtained hierarchy against the Google Ad Category Tree (Section IV-B). Finally, we present interesting insights into the structure of mobile app markets through an in-depth analysis of the modified hierarchy (Section IV-C).

**A. Performance evaluation of app categorization**

In order to have a basis for comparison, the LibLinear software [12] implementation of L2-regularized logistic regression [13] is run on the same input data. Experimental results presented in this section are obtained through 10-fold cross validation on the 1,065 apps with ground truth labels. Each app is characterized by a TF-IDF feature vector of length 7,745.

**Parameter setting.** Through a series of sensitivity analyses, we found that $\beta$ parameter that controls the sparsity of the feature-to-class matrix $L$ achieves its highest accuracy when it is between $10^4$ and $10^5$. The large value of $\beta$ signifies that the $L$ is sufficiently sparse to avoid model overfitting. For $\gamma$ that penalizes deviation of the estimated class similarity $B$ from the input similarity $P$, we find the overall accuracy only changes by 5% when it ranges from $10^{-3}$ to $10^0$. This suggests that we are able to give freedom to new categorization structure by deviating it from the input categorization structure without sacrificing accuracy. We set the default values of the $\beta$ and $\gamma$ parameters as $5 \times 10^4$ and 0.2, respectively.

**Metrics.** To measure the accuracy of the framework on the set of classified apps that satisfy the minimum threshold $\tau$, we define Accuracy as the fraction of correctly classified apps among all the classified apps, i.e., $|\text{correctly classified apps} \cap \text{classified apps}|/|\text{classified apps}|$, and define coverage as $|\text{classified apps}|/|\text{all apps tested}|$.

Our semi-supervised NMF framework outputs a class membership matrix $Y_u$ for the unlabeled apps. We further apply normalization to $Y_u$ by normalizing each row vector with its row sum. Each entry $Y_u(i,j)$ then provides a membership score of unlabeled app $i$ in class $j$ between 0 and 1. For hard classification (i.e., categorizing an app into only one label), we assign the app to the top score class only if its membership score exceeds a minimum threshold $\tau$. The threshold $\tau$ allows the classifier to refrain from making a prediction unless it is absolutely confident of its prediction. For soft classification, we select top-$r$ classes for app $i$ among the values of $Y_u(i,*)$ above $\tau$.

**Hard classification accuracy.** Employing a score threshold $\tau$ on the normalized class membership matrix $Y_u$, we plot the trade-off curve between coverage and accuracy in Figure 4 and observe that our (hard) semi-supervised NMF algorithm consistently outperforms the accuracy of logistic regression by about 2%. For example, with $\tau = 0$, our semi-supervised NMF achieves accuracy of 83.2% with a coverage of 100%. For the same coverage, we obtain 81.4% accuracy using logistic regression. With 70% coverage, accuracy of our scheme raises to 93.8% while accuracy of logistic regression is at 91.0%.

**Soft categorization accuracy.** In the case of soft categorization, our framework outputs the top-$r$ class labels for each app. An app is said to be correctly classified if its ground truth label matches one of the top-$r$ predicted categories. We present our soft categorization results for $r = 2$ and $r = 3$. Figure 4
shows that our NMF with $r = 2$ exhibits 90% accuracy, which is a 7% improvement over hard classification. NMF with $r = 3$ yields a 93% accuracy, achieving 10% improvement. In addition, the soft categorization with $r = 3$ consistently outperforms hard classification with logistic regression by 12%. Inspection on individual classification results discovered that the soft classification approach enables several apps to be correctly assigned to multiple categories. For example, a fashion shopping app with ID com.holosfind.showroom was assigned to both news::shopping search and beauty & fitness::fashion & style. As another example, an instant messaging app with social networking capabilities with ID jp.naver.line.android was classified into both online communities::social networks and internet & telecom::email & messaging.

**Area Under Curve (AUC) for each class.** Inspired by the above results on individual class, we further measure classification performance for each class by employing a well-known metric, Area Under Curve (AUC) of Receiver Operating Characteristic (ROC) [14].

Using right y-axis of Figure 5, we plot a histogram of 49 Google-ad categories that represents the number of apps in each category (in black). Using left y-axis of Figure 5, we plot two curves that represents AUC of our NMF for each class (in red), and AUC of logistic regression (in blue). For fairness, we use hard classification for both algorithms. For large classes with more than 50 apps, both NMF and logistic regression yields similarly good AUC performance. However, for smaller classes with less than 50 apps, we observe that semi-supervised NMF clearly outperforms logistic regression. We speculate that the introduction of side information helps NMF to perform well for imbalanced categories. Logistic regression, on the other hand, cannot perform well as it cannot build reliable class membership model for classes with small amount of input data.

**B. Evaluation of Modified Class Hierarchy**

Apart from improved accuracy in app classification, our framework provides a unique advantage over existing categorization algorithms by proposing a modified category structure. In this section, we analyze the macro-scale structure of Google Play market through our fine-grained, mobile app optimized categorization. Specifically, Eq. 3 induces an estimate of the pairwise class similarity matrix $B$ from the data. By applying the group average hierarchical clustering method on $B$, we construct a modified class hierarchy as shown in Figure 6. The modified hierarchy contains a number of interesting observations.

**Adequacy of Google Ad tree on app categorization.** Before we discuss structural changes made by our framework, we mention an observation both held in the input and the modified hierarchy. As denoted in braces under x-axis in Figure 6, the majority of second level classes (9 out of 10 classes) retains their original sub-categorical information provided by Google Ad tree (except for Jobs & Education: Job Listings, which does not share a common parent as Jobs & Education: Education). The fact that the sub-classes under first level classes such as Games, News, Reference, and Travel are preserved shows that, by and large, Google Ad tree has an adequate taxonomy for textual descriptions on Google Play market. The two exceptional subclasses (Jobs & Education: Job Listings and Jobs & Education: Education) exhibit a large distance because their topics are very different.

**Associations among first-level classes.** Although we consider the input Google ad tree comprised of only two levels, the induced hierarchy exhibits more complex structure. Specifically, groupings of originally first-level classes such as {Online communities, Beauty & fitness, and Travel} and {News, Reference, and Arts & entertainment} suggests that Google Ad tree did not fully represent affinity among its first level nodes.

Figure 6 shows a closer examination on a pair of example classes, News and arts & entertainment. As it can be seen from their class labels, the key terms of the second-level classes can be summarized as “media.” Upon inspection on common key terms across classes, we observe that Arts & entertainment class share over twice as many terms (5 terms) with News as the rest of classes (average of 2 terms).

Through our analysis on pair-wise class comparisons as exemplified above, we notice that there are largely three groups of classes among the previous first-level classes as denoted in red group labels in Figure 6.

---

**Fig. 5.** AUC comparison for classes with different number of apps.

**Fig. 6.** Overview of modified category hierarchy obtained by hierarchical clustering on $B$. 

---
1) **Social networks & photographs group.** We find a group comprised of Online communities, Beauty & fitness, and Travel. The classes share common terms such as “social networks”, “blogs”, “photos”, and “fashion”.

2) **Media group.** In the middle of the graph, we find another group comprised of News, Reference, and Arts & entertainment. The classes share the following popular terms: “news papers”, “books”, and “(TV) programs”.

3) **Game & education group.** On the right side of the graph, we find the last group comprised of Games, Education, and Internet & telecom. They share the following terms: “fun”, “play”, “learn”, and “online”.

Interestingly, we corroborate our finding on these three larger groups of Social networks, Media, and Game with a report from business research [15] stating that the three groups are the top three user activities on smart devices, collectively spanning 80% of all activities. While our research on app market should not directly reflect popularity of apps to users (or frequency of app use), the high-level similarity signifies that there is a non-negligible relationship between the two.

**Separation among second-level classes.** The induced similarity can separate second-level classes that were originally on a single branch. In Figure 7, we observe a sub-class, Education, which is originally under Jobs & education being separated from the rest of the siblings and gets merged with subclasses of another class, Games::*.

To further investigate the association between these classes, we plot Figure 8 that displays a histogram of terms shared between Jobs & education::education and ten level-1 classes. As shown in the figure, we observe that Games category shares more than twice more terms (10 terms) than the rest of categories (3 terms on average). Upon inspection of shared terms between Jobs & education::education and level-2 subclasses of Games::*, we observe that Games::Puzzles and Games::Educational games share large number of common words (8 and 7 terms, respectively) while the rest of the subclasses share an average of 2 terms with Jobs & education::Education. From these results, we can confirm that the keywords associated with Jobs & education::Education subclass (e.g., training, teaching, lessons) are semantically closer to Games::Puzzles and Games::Educational games than with its previous-siblings, Job listings and Internships.

**C. Macro-scale Analysis of Google Play Market**

For our macro-scale experiment, we classify 166,937 unlabeled Google Play apps by applying feature weight $W$ we trained earlier. Since there is no ground truth label available for these apps, instead of calculating the accuracy of the framework, we examine the class distribution of the apps generated by our framework and compare it against their original market categorization assigned by the developers. Figure 9 shows a comparison between the 30 Google Play categories (24 non-game categories and 6 subcategories of Games) on the x-axis and the 49 Google-Ad categories on the y-axis. Each cell in the heat map represents the number of apps that belong to the Google Play category that were assigned to the corresponding Google Ad category by our framework. There are several interesting insights provided by this analysis.

First, there are quite a substantial number of Google Play categories that were mapped to a single Google Ad category. Upon further examination, many of the uniquely mapped labels are consistent with our general expectation. For instance, almost all the shopping apps from Google Play categories are mapped into News::shopping search from Google-ad; almost all the sports apps from Google Play go into News::sports news from Googld-ad, etc. This supports our initial assumption that the Google Ad preference tree provides good guidance to the classification of the mobile apps.

Second, we observe some coarse-grained Google Play categories being mapped into multiple fine-grained categories in Google Ad tree. For example, apps from Communication category were separated into three sub-categories of internet & telecom::email & messaging, internet & telecom::teleconferencing, and online & communities::social networks, all located closely in our hierarchy. In the case of apps from shopping category, they categorized into two distant categories of news::shopping search and beauty & fitness::fashion & style. Both of these examples demonstrate the advantage of our framework in providing detailed categorization beyond original market categories.

**V. Related Work**

**App classification.** Previous work on mobile app classification categorizes the apps along various dimensions. [16] categorized the apps into a three-dimensional taxonomy with respect to the app development process. [17] proposed a hierarchical taxonomy of the apps from a business perspective while [18] defines a taxonomy of apps from the perspective of user-app interactions. None of these works consider an
automated approach for classifying mobile apps, thus limiting the scalability of these approaches.

To overcome this limitation, a number of categorization methods employed machine learning techniques including maximum entropy classifier [19], bayesian networks and k-nearest neighbor [20]. The approaches, however, were neither designed to incorporate class hierarchy nor do they reorganize input categorization such that the modified structure can better reflect affinity among input data.

**NMF for classification.** Mobile apps classification can be regarded as a special class of document classification. In recent years, the basic NMF framework has received wide attention from the text mining community ([21], [22]) thanks to its effective handling of large-scale corpora through dimensionality reduction. Unlike the approach proposed in this paper, the NMF formulation used in [21][22] focused on flat (non-hierarchical) categorization of documents. However, many real-world classification problems involve a large number of classes that are not entirely independent of each other.

These problems can be naturally casted as a hierarchical learning problem, where the goal is not only to classify the data points into their appropriate classes, but also to induce a hierarchical relationship among the classes. In [23], a fast hierarchical document clustering algorithm based on NMF is constructed to generate a nested hierarchy of document clusters. The method is unsupervised and only considers binary splits at each iteration. Thus, the method is inapplicable to the mobile app categorization problem.

In contrast, [2] proposed a novel recursive NMF approach that leverages supervision on ground truth labels. The proposed scheme automatically learns a labeling tree and conducts model for testing on large multi-class classification. However, unlike the framework presented in this paper, the recursive NMF scheme developed in [2] does not accommodate external class hierarchies obtained from other sources. As shown in our experimental results, the fine-grained categories obtained from the Google Ad Tree helps guiding detailed classification even for classes with few training data.

### References


