Hindawi
Mobile Information Systems
Volume 2020, Article ID 4521723, 12 pages
https://doi.org/10.1155/2020/4521723

Research Article

A Collaborative Deep and Shallow Semisupervised Learning Framework for Mobile App Classification

MingQi Lv, Chao Huang, TieMing Chen, and Ting Wang

Department of Computer Technology, ZheJiang University of Technology, Hangzhou, China

Correspondence should be addressed to TieMing Chen; tmchen@zjut.edu.cn

Received 14 June 2019; Revised 15 December 2019; Accepted 21 December 2019; Published 14 February 2020

Academic Editor: Claudio Agostino Ardagna

With the rapid growth of mobile Apps, it is necessary to classify the mobile Apps into predefined categories. However, there are two problems that make this task challenging. First, the name of a mobile App is usually short and ambiguous to reflect its real semantic meaning. Second, it is usually difficult to collect adequate labeled samples to train a good classifier when a customized taxonomy of mobile Apps is required. For the first problem, we leverage Web knowledge to enrich the textual information of mobile Apps. For the second problem, the mostly utilized approach is the semisupervised learning, which exploits unlabeled samples in a cotraining scheme. However, how to enhance the diversity between base learners to maximize the power of the cotraining scheme is still an open problem. Aiming at this problem, we exploit totally different machine learning paradigms (i.e., shallow learning and deep learning) to ensure a greater degree of diversity. To this end, this paper proposes Co-DSL, a collaborative deep and shallow semisupervised learning framework, for mobile App classification using only a few labeled samples and a large number of unlabeled samples. The experiment results demonstrate the effectiveness of Co-DSL, which could achieve over 85% classification accuracy by using only two labeled samples from each mobile App category.

1. Introduction

The proliferation of mobile devices in recent years has led to a rapid growth of mobile applications (represented as "App" in the rest of the paper), which play a more and more important role in mobile users' daily lives. For example, in 2017, there are about 64 billion and 28 billion App downloads at Apple's App Store [1] and Android's Google Play [2], respectively. With such a huge number of Apps, it is necessary to classify them into predefined semantic categories to support various intelligent services, such as App search, App recommendation, and user profiling [3–5]. Take user profiling as an example, if user A often uses "*Angry Bird*" and user B usually uses "*Fruit Ninja*," we could know that they both like to play casual games based on an appropriate App classification.

Someone may argue that most App delivery platforms (e.g., App Store and Google Play) have already predefined the taxonomy of Apps and classified each App as one of the categories. However, the App classification provided by App delivery platforms has the following limitations. First, the classification results usually cannot be obtained by third party services. For example, the Android development platform can only access to the App information such as name and version, not including category from delivery platforms. Second, the predefined App taxonomy from delivery platforms is fixed, while users may usually need customized App taxonomy. For example, the App "*The Palace Museum*" is classified as "*Life*" in the App Store, which is sometimes too general for understanding its real usage, e.g., some specific applications might prefer to tag it as "*Tour*."

Thus, it is necessary to design a method that can classify Apps into customized categories based only on the most easily accessed data, i.e., App names. However, it is a challenging task, because the App names are usually too short, and the contained words are too sparse to reflect the relevance between Apps and semantic categories [6]. For example, it is almost impossible to tell the category of App "*YouTube*" by only looking at its name. Aiming at this challenge, most existing works enrich the semantic

information of App names using external knowledge (e.g., Web search engine) by borrowing the idea from short text classification [7–9] and then train App classifiers based on supervised machine learning techniques. However, the existing App classification methods based on supervised machine learning techniques still have the limitation that they require a large number of labeled samples to train the classifiers, especially the deep learning techniques. However, it is an expensive task to collect adequate labeled samples in practice, when a customized App taxonomy is required.

Semisupervised learning is one of the prominent ways to address the problem of limited labeled samples by learning from both labeled and unlabeled samples without human intervention [10]. In semisupervised learning, an effective paradigm is referred to as disagreement-based semisupervised learning [11], where multiple base learners are trained and they learn from each other collaboratively to improve the performance by exploiting the disagreement among them. In disagreement-based semisupervised learning, diversity of base learners is the key requirement [11]. Most existing methods ensure the diversity based on feature split (i.e., using different features to train the base learners). However, this scheme heavily relies on the strategy of feature split. When the features are not properly split, the performance would be greatly degraded [11]. In this paper, we exploit totally different machine learning paradigms (i.e., shallow learning and deep learning) to ensure a greater degree of diversity without feature split. Different from the existing disagreement-based semisupervised learning algorithms, we found that the greatest problem of fusing shallow learning and deep learning in a collaborative framework is that they have highly unbalanced and dynamically varied performance. For example, deep learning learners usually have stronger performance when there are adequate labeled samples. This problem would cause the stronger base learner to easily learn noises from the weaker base learner.

To this end, this paper proposes Co-DSL, a collaborative deep and shallow semisupervised learning method for App classification. First, in order to enrich the textual features of an App, we take advantage of a Web search engine to obtain snippets for the App by using its name as the query keyword. Second, we apply Co-DSL, which uses only a few labeled samples and a large number of unlabeled samples, by exploiting the complementary discriminative power of totally different machine learning paradigms (i.e., deep learning and shallow learning) in a cotraining style. Specifically, the main contributions of this paper are summarized as follows.

(1) We propose a method for automatic App classification. First, it uses Web knowledge to enrich the textual features of mobile Apps. Second, it could train the App classifier using a very limited number of labeled samples. Based on the experiment results, by using only two labeled samples from each App category, the proposed method could achieve over 85% classification accuracy.

(2) We propose a semisupervised learning framework called Co-DSL, which has the following novelties. First, instead of using feature split, we exploit totally different machine learning paradigms (i.e., shallow learning and deep learning) to train the base classifiers. Second, to address the problem of unbalanced performance of base classifiers, we design a biased sampling strategy to choose the unlabeled samples (Section 3.3.1) and a base classifier accuracy estimation strategy to evaluate the base classifier with few labeled samples (Section 3.3.2) in each round of training.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 details the proposed method, including the base classifiers and the semisupervised learning framework. Section 4 reports the experiment results. Section 5 concludes the paper.

2. Related Work

2.1. App Classification. A major challenge of automatic App classification is that there are not many effective and explicit features due to the limited information of Apps available for analysis. In many cases, only the names of the Apps are available. Therefore, the existing works try to enrich the information of Apps by using various strategies.

The most fundamental strategy is the textual enrichment based on Web knowledge. For example, Ma et al. [12] used search engines to obtain several textual snippets to enrich the textual information of the Apps and extracted word frequency vectors as features to represent the Apps. Li et al. [13] also used search engines to enrich the textual information of the Apps and then extracted various features using vector space model and topic model. Garg et al. [14] crawled App descriptions from Google Play and adopted two topic models to create the feature vectors.

In addition to the textual information, some researchers exploit various types of extra data to enrich the information of Apps. The first type is metadata. For example, Berardi et al. [15] gathered metadata (e.g., file size, user rating, and rating count) from the App delivery platforms and selected effective features to build the App classifier. Chen et al. [16] defined multiple kernels based on various metadata (e.g., developer, update, and reviews) to learn a similarity metric of different Apps. Olabenjo [17] used a crawler to extract various attributes (e.g., is free or not and have in App purchases or not) of the Apps from Google Play to create the App classifier. The second type is contextual data. For example, Zhu et al. [6] found that different types of Apps are usually relevant to real-world contexts (e.g., business Apps are likely used in the work place), which could be leveraged for App classification. Radosavljevic et al. [18] considered both the temporal contexts of the sequences of App installation and the description of Apps and integrated these data into an embedding framework. According to the existing works, no matter what types of extra data are considered, the textual information from Web knowledge is always the most effective for distinguishing Apps.

2.2. Text Classification Based on Semisupervised Learning. Text classification is a well-studied problem [19, 20]. However, only a few works consider the problem that the

labeled samples are inadequate. The mostly exploited approach for this problem is semisupervised learning, which utilizes a large number of unlabeled samples in addition to the labeled samples to automatically improve the performance. For example, Nigam et al. [21] proposed a semisupervised learning framework for text classification. It uses an iterative EM (expectation maximization) technique to exploit both the labeled and unlabeled samples. Jiang and Li [22] proposed IWNB, a semisupervised learning framework based on Naïve Bayes, for text classification. It uses the classifier trained on labeled samples to label and weight the unlabeled samples and then retrains the classifier on all samples. Johnson and Zhang [23] proposed a semisupervised learning framework with convolutional neural networks (CNNs) for text classification. It learns the embeddings of text regions from unlabeled samples for integration into a supervised CNN.

In semisupervised learning, an effective paradigm, is referred to as disagreement-based semisupervised learning [11], where multiple base learners are trained, and the disagreement among these base learners is exploited. For example, Nigam [24] found that disagreement-based semisupervised learning algorithms (e.g., Co-Training) outperform other semisupervised learning algorithms when a natural split of features exists. Wan [25] used features of the English documents and features of the translated Chinese documents as the two independent views of the Co-Training algorithm to exploit the unlabeled Chinese documents. Shi et al. [26] constructed an ensemble of classifiers to progressively improve the performance of text classification. In disagreement-based semisupervised learning, diversity of base learners is the key requirement for improving the performance [11]. Different from the existing methods that exploit diversity based on feature split, Co-DSL ensures a greater degree of diversity using two totally different learning paradigms (i.e., shallow learning and deep learning). Since we explore semisupervised learning technique for App classification in this paper, we summarize the existing semisupervised learning techniques in Table 1.

3. Method

3.1. Overview. The proposed method for App classification consists of two main stages. In the first stage, we leverage Web knowledge to enrich the textual information of Apps. To be specific, as shown in Figure 1, given an App a, we firstly submit a's name to a Web search engine (e.g., Google API in our experiments). Then, we obtain the top M search snippets and integrate them into a document D_a (we call it as the App document of a). A search snippet is an abstract of the Web page that is returned as relevant to the submitted App's name. In order to ensure the quality of search snippets, we only keep the search snippets from some professional mobile App websites (e.g., Google Play, App Store, and Wikipedia). Finally, we preprocess D_a by removing all the stop words (e.g., "of" and "the") and normalizing verbs, nouns, and adjectives (e.g., "playing⟶ play," "games ⟶ game," and "better ⟶ good").

In the second stage, we use Co-DSL to train an App classifier. Specifically, as shown in Figure 2, we firstly train two initial base App classifiers using only a few labeled samples based on shallow learning and deep learning techniques, respectively. Then, we retrain and upgrade the two base App classifiers by exploiting a large number of unlabeled samples in a cotraining style. For example, assume that we had 10,000 App samples, and only 100 of them were properly labeled; we firstly train two base App classifiers based on the 100 labeled samples. Then, the base App classifiers could learn from each other in a cotraining style using the 9,900 unlabeled samples based on Co-DSL.

3.2. The Base Classifiers

3.2.1. The Base Classifier of Shallow Learning. The shallow learning technique trains the initial base classifier based on a variety of textual features extracted from the App documents. The textual features are extracted from two aspects, i.e., the vector space model (VSM) features and the topic model features.

The VSM features are extracted as follows. In the first step, we extract the top N keywords for each App category. Specifically, we firstly build a dictionary W that contains all the unique words in all App documents. Second, we calculate the weight of each word in W for each App category. The weight of word w_i for App category c_j is calculated based on the idea of TF-IDF [34] as equation (1), where $n(i, j)$ is the count of w_i in all App documents with App category c_j and C is the set of App categories. Third, we choose N unique words with the highest weights for each App category as keywords:

$$\text{weight}(i, j) = \frac{n(i, j)}{\sum_{w_k \in W} n(k, j)} \log \frac{|C|}{\left|\left\{c_j \mid n(i, j) \neq 0\right\}\right|}. \quad (1)$$

In the second step, we transform each App document D_a into a word vector $\mathbf{wv}(a)$. The ith element in $\mathbf{wv}(a)$ indicates the weight of the ith keyword in D_a ($1 \leq i \leq N \times |C|$), and it is also calculated based on the idea of TF-IDF as equation (2), where $m(i, a)$ is the count of the ith keyword in D_a and D is the set of App documents. Here, $\mathbf{wv}(a)$ is regarded as the VSM feature vector of D_a:

$$\mathbf{wv}(a)[i] = \frac{m(i, a)}{\sum_{1 \leq k \leq N \times |C|} m(k, a)} \log \frac{|D|}{\left|\left\{D_a \mid m(i, a) \neq 0\right\}\right|}. \quad (2)$$

Although the VSM features can capture the relevance between the App document and App category in terms of the word distribution, it does not consider the latent semantic meanings behind the words. For example, "song," "playlist," and "artist" are treated as totally different elements in VSM feature vector, but they implicitly reflect the same latent semantic topic "Music." Previous work also found that the latent semantic topics are useful for short text classification [35]. Thus, we use LDA (Latent Dirichlet Allocation) topic model [36] to learn the latent semantic topics and extract topic model features. The objective of LDA is to determine the word distribution for each topic and the topic distribution for each document by analyzing a large corpus of documents in an unsupervised way. After the LDA model is learnt, an App document D_a could be represented as a distribution of topics

Table 1: The summary of the existing semisupervised learning techniques.

Scheme	Principle	Drawbacks	References
Generative semisupervised learning	It assumes that both labeled and unlabeled samples are generated from the same parametric model; then, it treats the labels of the unlabeled samples as missing values of the model parameters	When the parametric model assumption is incorrect, fitting the model using unlabeled samples would result in performance degradation	[21, 27, 28]
Graph-based semisupervised learning	It constructs a graph whose nodes are samples (including both labeled and unlabeled samples) and edges reflect relations between nodes (e.g., feature similarity); then, the labels are propagated by exploiting the connective characteristics	First, it suffers from poor scalability; second, it is difficult to build the relations between samples when the features are sophisticated	[29–31]
Disagreement-based semisupervised learning	Multiple base learners are initially trained on labeled samples, and then they learn from each other by exploiting the disagreement among them on unlabeled samples	How to guarantee the diversity between base learners is an open problem	[25, 32, 33]

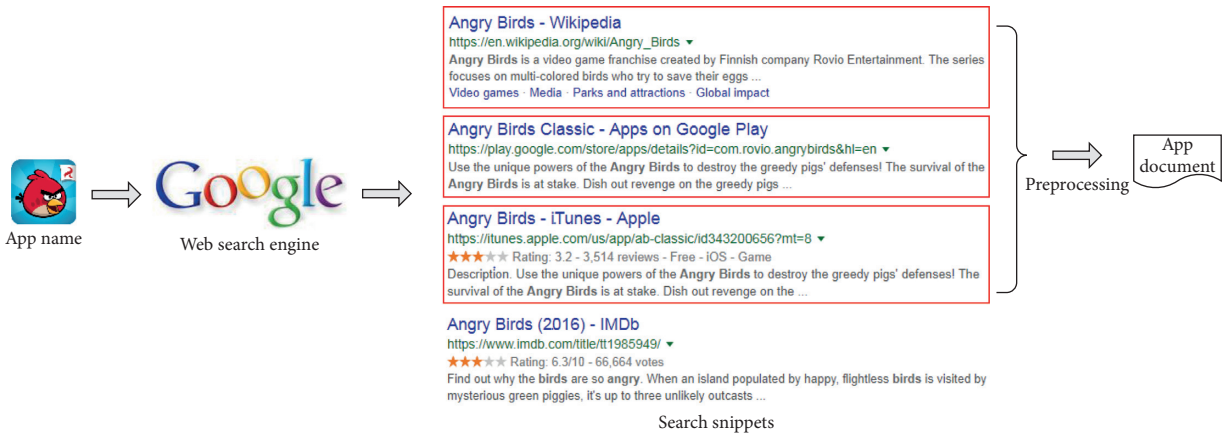


Figure 1: The procedure of enriching Apps based on Web knowledge.

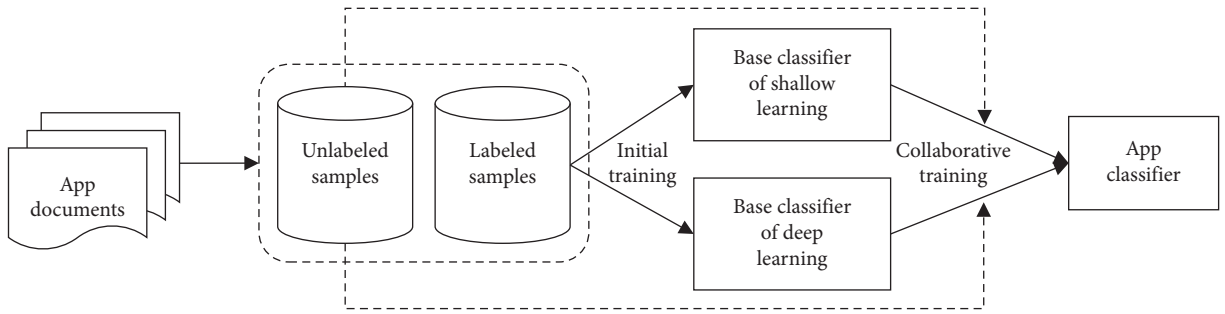


Figure 2: The procedure of Co-DSL.

$\mathbf{tv}(a) = \langle p_a^1, p_a^2, \ldots, p_a^T \rangle$, where p_a^k is the probability that the kth topic is present in D_a. Here, $\mathbf{tv}(a)$ is regarded as the topic model feature vector of D_a.

For each App document D_a, we concatenate $\mathbf{wv}(a)$ and $\mathbf{tv}(a)$ as the final feature vector. Then, we train a base classifier to map the feature vectors to the App categories based on a specific machine learning algorithm (e.g., SVM).

3.2.2. The Base Classifier of Deep Learning. In this paper, we apply convolutional neural network (CNN) proposed in [37] to train the initial base classifier since it shows a good balance between classification accuracy and training efficiency for the text classification tasks. Note that the training efficiency is especially important in our method since the training of base classifiers would be performed multiple times in Co-DSL.

The architecture of the CNN is shown in Figure 3, including an embedding layer, a convolutional layer, a pooling layer, and an output layer. First, the embedding layer initially uses pretrained word vectors [38], and fine-tunes them through training via back propagation. Let the dimension of

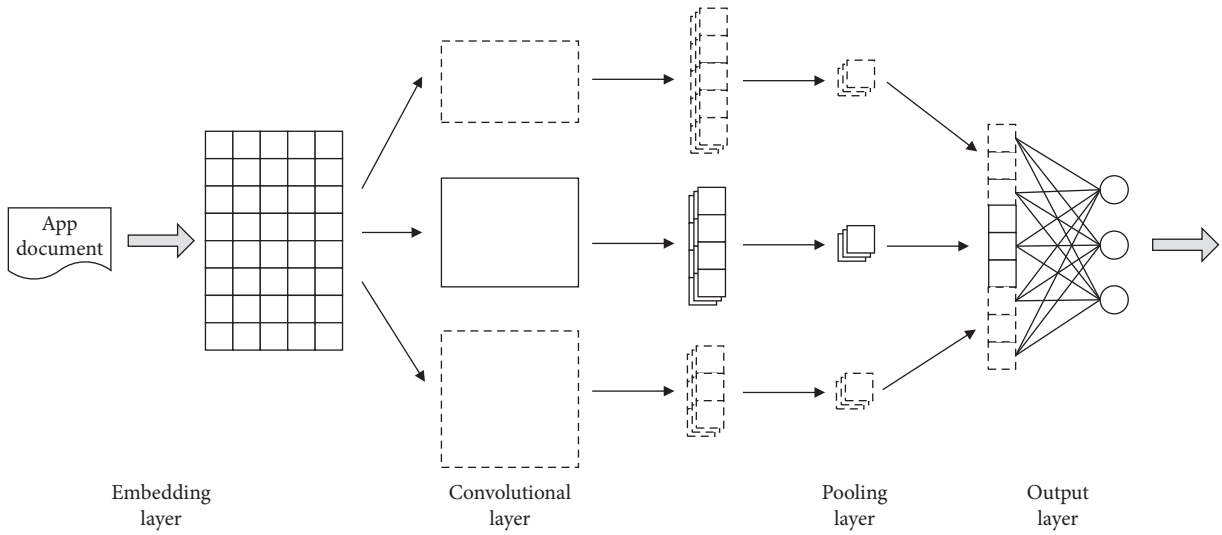


Figure 3: The architecture of the CNN for training the base classifier.

the word vectors be d, the word vector corresponding to the ith word in an App document D_a is notated as $\mathbf{x}_i$, and the App document D_a of length l is then represented as a $l \times d$ matrix. Note that an app document would be padded or truncated if it was shorter or longer than l. Second, the convolutional layer uses multiple filters with different window sizes to extract the feature maps. Specifically, we use three window sizes (i.e., $h \times d$, $h = 3$, 4, and 5) and define 100 filters for each window size. A filter is applied to each possible window of h words in D_a (i.e., $\{\mathbf{x}_{1:h}, \mathbf{x}_{2:h+1}, \ldots, \mathbf{x}_{l-h+1:l}\}$) to produce a feature map $\mathbf{f} = \langle f_1, f_2, \ldots, f_{l-h+1} \rangle$, where $f_i = \text{ReLU}\ (\mathbf{w}\mathbf{x}_{i:i+h-1} + b)$. The convolutional layer would produce 300 feature maps. Third, the pooling layer applies a max-over-time pooling scheme to process the feature maps (i.e., it only takes the maximum value of a feature map). Thus, the pooling result of feature map $\mathbf{f}$ is $\max\{\mathbf{f}\}$. By concatenating the pooling results of all filters, we could obtain a feature vector with a uniform length (i.e., 300) for different App documents. Fourth, the feature vector from the pooling layer is passed to a fully connected layer to output the probability distribution over App categories. Dropout scheme is employed on the fully connected layer to prevent overfitting.

3.3. The Semisupervised Learning Framework. When a new taxonomy of Apps is defined, the training samples have to be relabeled manually. It is an extremely expensive task. Therefore, we could only obtain a small number of labeled samples and a large number of unlabeled samples in practice.

Co-DSL is designed by extending the classical disagreement-based semisupervised learning algorithm Co-Training [32], which trains two base classifiers based on a feature split. Each base classifier is initialized using a few labeled samples on hand. At each round of training, each base classifier chooses one unlabeled sample per class with the highest classification confidence to add to the labeled sample set. Then, each base classifier is retrained from the augmented labeled sample set, and the process is repeated.

3.3.1. The Biased Sampling Strategy. Co-Training treats each base classifier equally, which means that each base classifier chooses the same number of unlabeled samples to add to the labeled sample set in each round of training. However, we found that the two base classifiers in this paper have highly unbalanced discriminative power. This problem would cause the stronger base classifier to easily learn noises from the weaker base classifier. The worst part is the advantage of the base classifiers is variable with the training iteration, e.g., the base classifier of shallow learning has higher performance when the labeled samples are extremely limited, while the base classifier of deep learning gains its advantage when the labeled sample set becomes larger.

Aiming at this problem, we design a biased sampling strategy for choosing unlabeled samples. The idea is that in each round of training, the number of unlabeled samples chosen by each base classifier is proportional to its performance. Thus, a stronger base classifier would choose more samples with higher quality than a weaker base classifier did. Figure 4 shows the pseudo-code of Co-DSL. In each round of training, Co-DSL works in two steps, i.e., retraining and selection. In the retraining step, it applies the shallow and deep learning techniques in Section 3.2 to train two base classifiers based on the current labeled sample set and then evaluates the accuracy of the two base classifiers (lines 3–6). In the selection step, it firstly applies each base classifier to the unlabeled sample set (line 7). Second, each base classifier chooses multiple unlabeled samples per App category with the highest classification confidence to augment the labeled sample set of another base classifier (lines 8–11). Here, the number of chosen unlabeled samples is proportional to the accuracy of the base classifier, i.e., $\min\{n_{\text{SC}}(j), \text{ceil}(\delta \times P_{\text{SC}})\}$ for base classifier SC and $\min\{n_{\text{DC}}(j), \text{ceil}(\delta \times P_{\text{DC}})\}$ for base classifier DC, where δ is a basic number, ceil (x) is a function that returns the smallest integer greater than or equal to x, $n_{\text{SC}}(j)$ is the number of unlabeled samples in U_2 that SC classifies as c_j, and $n_{\text{DC}}(j)$ is the number of unlabeled samples in U_1 that DC classifies as c_j.

Co-DSL

Input: The labeled sample set, L
The unlabeled sample set, U
The shallow learning technique in Section 3.2.1, $Learner_1$
The deep learning technique in Section 3.2.2, $Learner_2$
The basic number of the chosen unlabeled samples in each iteration, δ

Output: The two base classifiers, SC and DC

1. $L_1 \longleftarrow L, L_2 \longleftarrow L, U_1 \longleftarrow U, U_2 \longleftarrow U$
2. **Repeat** for T rounds:
3. $SC \longleftarrow$ use $Learner_1$ to train a base classifier based on L_1
4. $DC \longleftarrow$ use $Learner_2$ to train a base classifier based on L_2
5. $P_{SC} \longleftarrow$ evaluate the accuracy of SC
6. $P_{DC} \longleftarrow$ evaluate the accuracy of DC
7. Apply SC to each sample in U_2 and DC to each sample in U_1
8. **for** each App category c_j **do**
9. Pick min $\{n_{SC}(j), \text{ceil}(\delta \times P_{SC})\}$ samples that SC mostly confidently classifies as c_j, and add it to L_2
10. Pick min $\{n_{DC}(j), \text{ceil}(\delta \times P_{DC})\}$ samples that DC mostly confidently classifies as c_j, and add it to L_1
11. **end for**
12. **if** U_1 and U_2 are both empty **then**
13. Break

Figure 4: The pseudo-code of Co-DSL.

The ceil $(*)$ function is used to ensure that at least one unlabeled sample would be picked for each App category in each iteration.

After the two base classifiers are trained, we fuse them to output the final classification result based on the Stacking framework [39]. Given the set of App categories $C = \{c_1, c_2, \ldots, c_M\}$ and the set of training samples $X = \{x_1, x_2, \ldots, x_N\}$, $P_{\text{SC}}(x_i, c_j)$ and $P_{\text{DC}}(x_i, c_j)$ denote the confidence of classifying x_i as c_j by SC and DC, respectively. Then, a meta-classifier MC is trained based on the outputs of the base classifiers, i.e., MC: $P \longrightarrow C$, where $P = \{<P_{\text{SC}}(x_1, c_1), \ldots, P_{\text{SC}}(x_1, c_M), P_{\text{DC}}(x_1, c_1), \ldots, P_{\text{DC}}(x_1, c_M)>, \ldots, <P_{\text{SC}}(x_N, c_1), \ldots, P_{\text{SC}}(x_N, c_M), P_{\text{DC}}(x_N, c_1), \ldots, P_{\text{DC}}(x_N, c_M>\}$.

3.3.2. The Accuracy Estimation Strategy. The biased sampling strategy is designed based on the evaluation of the base classifiers (lines 5-6), which is not an easy task here. A classifier is normally evaluated by using a testing set segmented from the original labeled sample set (e.g., by performing cross-validation), but the original labeled samples are too few to support the testing set segmentation in our problem. Aiming at this problem, we design a base classifier accuracy estimation strategy as follows.

First, we construct an undirected graph SG over all the original unlabeled samples. Each vertex in the graph $v_i = (\mathbf{z}_i, y_i)$ corresponds to a sample, where $\mathbf{z}_i$ is the feature vector of the sample and y_i is the predicted label of the sample. In this paper, $\mathbf{z}_i$ is represented as the VSM feature vector in Section 3.2, and y_i is predicted by the base classifier to be evaluated at each round of training. There is an edge connecting two vertices v_i and v_j if v_i is close to v_j in the feature space. In this paper, we employ the k-nearest neighbour criterion to generate the edges (i.e., v_i is close to v_j if v_i is among the k-nearest neighbours of v_j or v_j is among the k-nearest neighbours of v_i). Here, we use the cosine similarity of $\mathbf{z}_i$ and $\mathbf{z}_j$ to measure the similarity of v_i and v_j, which is also associated as the weight w_{ij} if there is an edge between v_i and v_j.

Second, we exploit the structure of SG to estimate the accuracy of SC and DC at each round of training. Our evaluation strategy is designed based on a basic assumption that a correctly labeled sample should possess the same label to most of its neighbours (i.e., those having an edge with it in SG), which coincides with the manifold assumption that samples with high similarity in the input space would also have high similarity in the output space [40]. At each round of training, we apply the retrained SC to predict the label of each sample in SG, and an edge in SG is defined as a cut edge if the connected two vertices have different predicted labels. Then, we estimate the accuracy of SC based on equation (3), where $\text{SG}.V$ represents all the vertices in SG, $N(v_i)$ denotes the set of samples that are connected with v_i in SG, and I_{ij} is a function that returns 1 if the edge between v_i and v_j is not a cut edge and returns 0 if the edge is a cut edge. We use the same procedure to estimate the accuracy of DC at each round of training:

$$\text{eAcc} = \sum_{v_i \in \text{SG}.V} \frac{\sum_{v_j \in N(v_i)} w_{ij} I_{ij}}{\sum_{v_j \in N(v_i)} w_{ij}}. \quad (3)$$

4. Experiment

4.1. Experiment Setup and Datasets. We collected 4364 Apps from AppBrain. We used part of the App categories in AppBrain to define the App taxonomy in the experiments. The App taxonomy contains 11 App categories, and the details are summarized as follows. 8.5% Apps belong to "Social," 7.3% Apps belong to "Navigation," 8.6% Apps

belong to "Music," 10.3% Apps belong to "News," 11.4% Apps belong to "Photograph," 10.8% Apps belong to "Shopping," 9.7% Apps belong to "Communication," 9.3% Apps belong to "Education," 9.0% Apps belong to "Finance," 5.8% Apps belong to "Food," and 9.3% Apps belong to "Health." It can be seen that the distribution is relatively even.

The default values of the parameters are set as follows. For the base classifier of shallow learning, we set the number of keywords for each App category $N = 1000$ and the number of topics $T = 11$. For the base classifier of deep learning, we set the dimension of the word vectors $d = 200$ and the length of App documents $l = 3000$. For the semisupervised learning framework, we set the number of nearest neighbours in the k-nearest neighbour criterion $k = 5$, and the basic number of the chosen unlabeled samples in each iteration δ is tuned in the experiment.

4.2. Experiment 1: The Evaluation of the Base Classifiers. In the first experiment, we evaluate the performance of different base classifiers in a supervised learning manner. We adopt 5-fold cross-validation strategy for evaluation. Specifically, the base classifiers are trained on 80% of the labeled samples and tested on the remaining 20% labeled samples. Then, the procedure is repeated five times, and the average performance is reported. Three base classifiers of shallow learning (i.e., SVM, Naive Bayes, and C4.5) and two base classifiers of deep learning (i.e., MLP and CNN) are evaluated. The base classifiers of shallow learning uses features discussed in Section 3.2.1, CNN is the base classifier discussed in Section 3.2.2, and MLP is a deep neural network that replaces the convolutional layer and pooling layer of CNN with a multilayer perceptron layer. The experiment results are shown in Figure 5(a). It can be clearly found that SVM has the best performance among the base classifiers of shallow learning, and CNN has the best performance among the base classifiers of deep learning. Thus, we use SVM as the base classifier of shallow learning and CNN as the base classifier of deep learning in the following experiments.

In the second experiment, we evaluate the performance of the base classifiers by varying the number of the labeled samples. As shown in Figure 5(b), SC denotes the base classifier of shallow learning, and DC denotes the base classifier of deep learning. First, the performance of both SC and DC improves with increasing the number of labeled samples. Second, when the labeled samples are extremely scarce (e.g., only one or two labeled samples for each App category), DC has a far worse performance than that of SC. Third, when there are relatively adequate labeled samples, DC starts to outperform SC. It shows that deep learning technique relies more on the number of labeled samples, and it gains advantage when more labeled samples are available.

4.3. Experiment 2: The Evaluation of the Semisupervised Learning Framework. In the first experiment, we evaluate the effect of δ (the basic number of the chosen unlabeled samples in each iteration in Section 3.3.1). The results are shown in Figure 6. First, the performance of both base classifiers generally increases along with the iterations of the semi-supervised learning procedure, while DC shows a far more significant increase. It might be because that DC has a lower initial accuracy and deep learning techniques benefit more from the number of labeled samples. Second, a larger δ leads to a faster improvement of performance, especially for DC. However, a larger δ tends to pick more unlabeled samples when the base classifiers are relatively weak, and thus the chosen samples might contain more noise. As a result, it is more difficult to reach the global optimum. In this experiment, the global optimum is reached when $\delta = 3$ for SC and $\delta = 5$ for DC. To this end, we set $\delta = 5$ in the rest of the experiments.

In the second experiment, we evaluate the effectiveness of the base classifier accuracy estimation strategy. We compare the real accuracy and the estimated accuracy by varying the number of the labeled samples. As shown in Figure 7, although the proposed strategy is only a heuristic way to estimate the accuracy of the base classifiers, the experiment results validate its usefulness in capturing the variation trend of the accuracy of the base classifiers. The mean absolute errors between the real accuracy and the estimated accuracy of SC and DC are 0.045 and 0.090, respectively.

In the third experiment, we evaluate the necessity of the biased sampling strategy, where "original sampling strategy" means that Co-DSL in Figure 4 picks $\delta = 5$ unlabeled samples for each App category in each iteration (lines 9 and 10) without considering the unbalanced performance of the two base classifiers. The results are shown in Figure 8. First, "biased sampling strategy" outperforms "original sampling strategy" in almost all the iterations for both SC and DC. It demonstrates the capability of "biased sampling strategy" for choosing unlabeled samples with higher quality. Second, as more iterations are performed, "biased sampling strategy" gains more advantage over "original sampling strategy." It shows that "biased sampling strategy" could effectively reduce the accumulated noises.

4.4. Experiment 3: The Comparison with Baselines. To evaluate the effectiveness and show its competitive performance of Co-DSL, we compare it with the following eight baseline methods, including five supervised learning baseline methods and three semisupervised learning baseline methods. The evaluation procedure is conducted as follows. The App document samples are divided into a training set and a testing set. We randomly choose two samples from each App category to form the training set, and the rest samples are all put in the testing set. The supervised learning baseline methods are trained using the training set and tested on the testing set. The semisupervised learning baseline methods are trained using both the training set (treated as labeled samples) and the testing set (treated as unlabeled samples), and tested on the testing set. The evaluation procedure is repeated ten times, and the average performance is reported.

The supervised learning baseline methods are as follows.

(1) **SC-NB**: it refers to the classifier of shallow learning in Section 3.2.1, i.e., training a Naive Bayes classifier based on VSM features and topic model features.

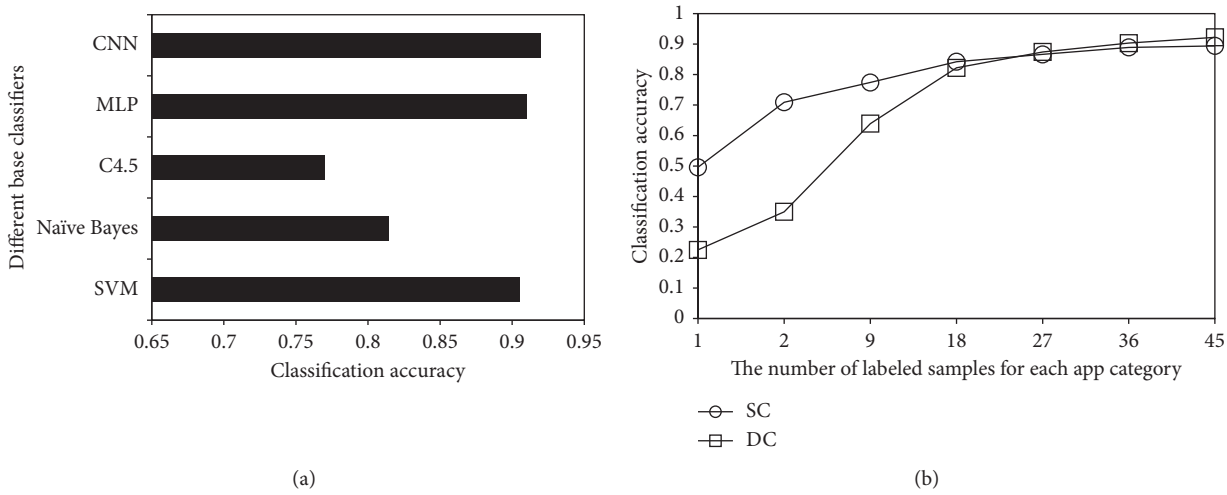


Figure 5: The evaluation of base classifiers: (a) the performance of different base classifiers; (b) the effect of the number of labeled samples.

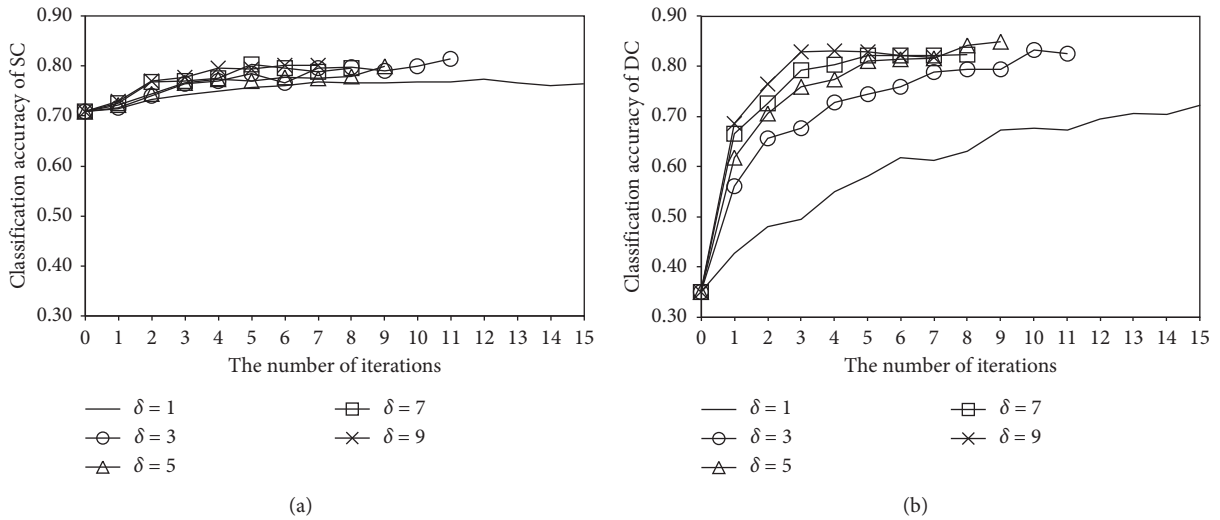


Figure 6: The effect of the parameter δ: (a) the effect on SC; (b) the effect on DC.

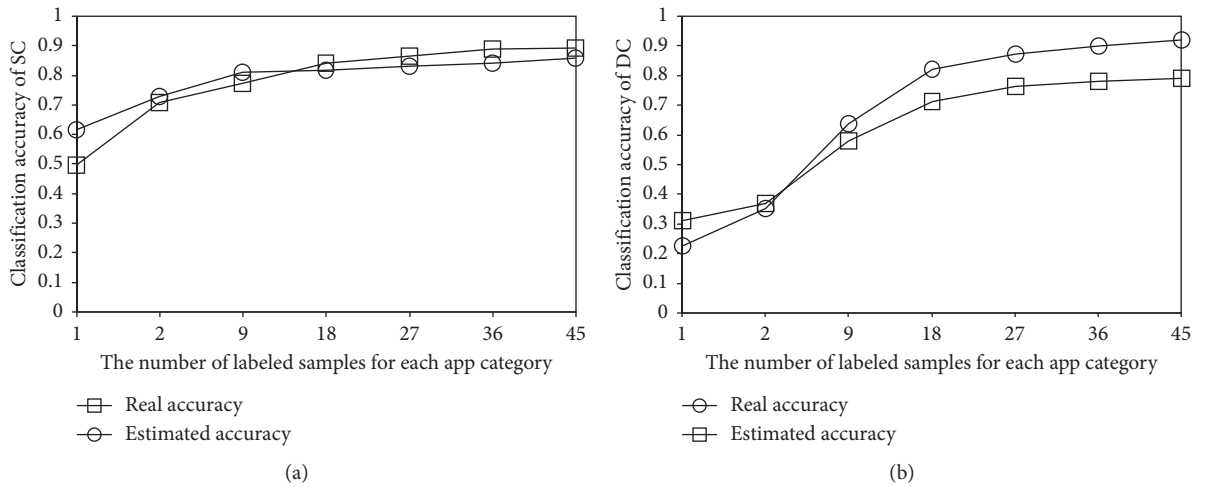


Figure 7: The performance of the base classifier accuracy estimation strategy: (a) the performance for SC; (b) the performance for DC.

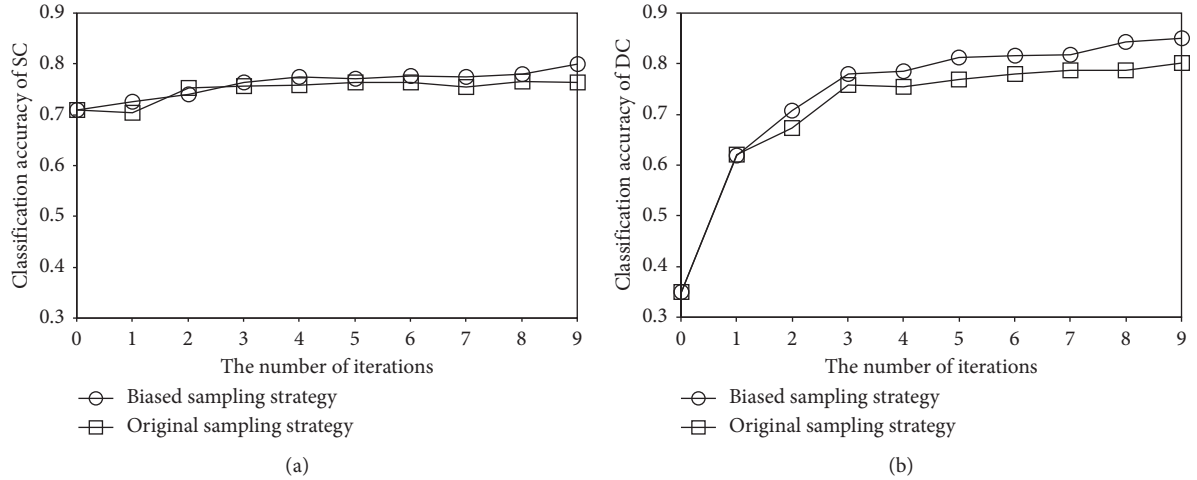
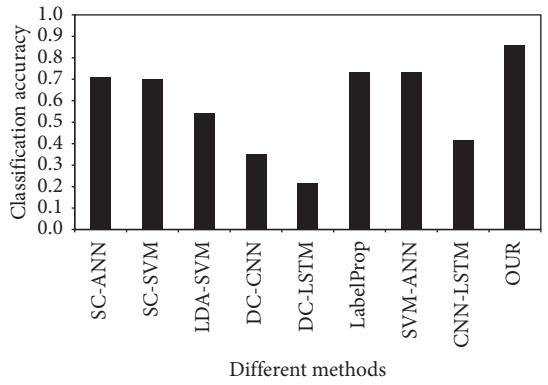


Figure 8: The effect of the biased sampling strategy: (a) the effect on SC; (b) the effect on DC.



(a)

	0	1	2	3	4	5	6	7	8	9	10
0	75%	0.0%	0.0%	0.0%	4.2%	4.1%	0.0%	16.7%	0.0%	0.0%	0.0%
1	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
2	0.0%	12.5%	87.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
3	4.2%	0.0%	0.0%	91.7%	0.0%	0.0%	0.0%	0.0%	0.0%	4.1%	0.0%
4	4.2%	0.0%	0.0%	0.0%	87.5%	0.0%	0.0%	0.0%	8.3%	0.0%	0.0%
5	0.0%	0.0%	8.3%	0.0%	0.0%	87.5%	4.2%	0.0%	0.0%	0.0%	0.0%
6	4.2%	0.0%	0.0%	0.0%	4.2%	0.0%	87.5%	0.0%	0.0%	4.1%	0.0%
7	0.0%	0.0%	0.0%	20.8%	4.2%	0.0%	0.0%	75.0%	0.0%	0.0%	0.0%
8	4.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	95.8%	0.0%	0.0%
9	12.5%	4.2%	0.0%	0.0%	0.0%	0.0%	8.3%	4.2%	8.3%	62.5%	0.0%
10	0.0%	0.0%	4.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	95.8%

(b)

Figure 9: The comparison experiments: (a) the comparison of different methods; (b) the confusion matrix of Co-DSL ("0" denotes "Social," "1" denotes "Navigation," "2" denotes "Music," "3" denotes "News," "4" denotes "Photograph," "5" denotes "Shopping," "6" denotes "Communication," "7" denotes "Education," "8" denotes "Finance," "9" denotes "Food," and "10" denotes "Health").

(2) **SC-SVM**: it also refers to the classifier of shallow learning in Section 3.2.1, and support vector machine (SVM) is used to train the classifier.

(3) **LDA-SVM**: it trains a classifier of shallow learning based on only the topic model features in Section 3.2.1 using SVM.

(4) **DC-CNN**: it refers to the classifier of deep learning in Section 3.2.2, i.e., training a classifier using CNN.

(5) **DC-LSTM**: it trains a classifier of deep learning using long short-term memory (LSTM).

The semisupervised learning baseline methods are as follows:

(6) **LabelProp**: it refers to a semisupervised learning framework called Label Propagation [29], which propagates labels to the unlabeled samples by leveraging the structure of a graph, constructed by

the similarity of the feature vectors between different samples. The feature vector of a sample is formed in Section 3.2.1.

(7) **SVM-NB**: it trains two base classifiers, both using shallow learning technique (i.e., using SVM and NB, respectively, based on the features in Section 3.2.1), and then integrates the two base classifiers based on the semisupervised learning framework in Section 3.3.

(8) **CNN-LSTM**: it trains two base classifiers, both using the deep learning technique (i.e., one is trained using CNN, and the other is trained using LSTM) and then integrates the two base classifiers based on the semisupervised learning framework in Section 3.3.

The results are shown in Figure 9(a). The following tendencies could be discerned from the results. First, the baselines using only deep learning techniques (i.e., DC-CNN, DC-LSTM, and CNN-LSTM) have far worse performance than those using only shallow learning techniques (i.e., SC-NB, SC-SVM, LDA-SVM, and SVM-NB), in both supervised learning and semisupervised learning cases. It is an interesting phenomenon. It shows that deep learning techniques are usually weak at learning knowledge from extremely small data (only two labeled samples for each App category are available). This conclusion could also be verified by the result that DC-LSTM has the worst performance since LSTM is the most complex model here, and it requires more labeled samples to train. Second, SVM-NB and LabelProp outperform SC-NB and SC-SVM, while CNN-LSTM outperforms DC-CNN and DC-LSTM. It indicates that the disagreement-based semisupervised learning techniques are effective at exploiting unlabeled samples to improve the classification accuracy. Third, SC-NB and SC-SVM outperform LDA-SVM. It indicates that the VSM features are more effective at representing App samples than topic model features do. Fourth, Co-DSL has better performance than SVM-NB and CNN-LSTM. It shows that combining shallow learning and deep learning techniques could achieve better performance than combining different shallow learning techniques or combining different deep learning techniques. Since the diversity between base classifiers is the key for the success of disagreement-based semisupervised learning, the result shows that different machine learning paradigms (i.e., shallow learning and deep learning) could provide diversity to a greater extent.

At last, we plot the classification confusion matrix of Co-DSL. The value in the ith row and the jth column is the proportion of App samples with the ith category that are classified as the jth category. As shown in Figure 9(b), it has relatively low classification accuracy on "Social" (it tends to be misclassified as "Education" or "Food"). It might be because that the definition of "Social" is relatively vague.

5. Conclusions

In this paper, we studied the problem of automatic App classification, which is a challenging task. First, name is the most easily accessed information of an App, but App names are usually too short to represent the semantic meanings. Second, it is difficult to collect adequate labeled samples to train a good classifier when a customized taxonomy of Apps is needed. Aiming at these challenges, we propose a collaborative deep and shallow semisupervised learning framework. It firstly leverages Web knowledge to enrich the textual features of the Apps. Then, it trains two base classifiers based on totally different machine learning paradigms (i.e., shallow learning and deep learning) to maximize the model diversity by using only a few labeled samples. Finally, it fuses the two base classifiers with unbalanced performance by exploiting a large number of unlabeled samples. The experiment results show that the proposed method outperforms the existing deep learning methods and semisupervised learning methods.

In the future, we will extend our work from the following directions. First, the method in this paper is designed based on natural language process (NLP) technique, which is computationally expensive. Therefore, we will try to make more efficient usage of the Web knowledge by preprocessing it based on the knowledge graph technique. Second, it is necessary to design a more efficient framework that could run on resource limited mobile devices.

Data Availability

The data used to support the findings of this study are available from the authors upon reasonable request. And the authors will publish their dataset in GitHub in the future.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Joint Funds of the National Natural Science Foundation of China (no. U1936215), the Zhejiang Provincial Natural Science Foundation of China (no. LY18F020033), and the National Natural Science Foundation of China (nos. 61772026 and 61602412).

References

[1] App Store. https://www.apple.com/ios/app-store/.

[2] Google Play. https://play.google.com/store/.

[3] D. Jiang, J. Vosecky, K. Leung, and W. Ng, "Panorama: a semantic-aware application search framework," in *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pp. 371–382, Genoa, Italy, March 2013.

[4] D. Cao, X. He, L. Nie et al., "Cross-platform app recommendation by jointly modeling ratings and texts," *ACM Transactions on Information Systems*, vol. 35, no. 4, 2017.

[5] H. Zhu, E. Chen, K. Yu, H. Cao, H. Xiong, and J. Tian, "Mining personal context-aware preferences for mobile users," in *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, pp. 1212–1217, Brussels, Belgium, December 2012.

[6] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian, "Mobile app classification with enriched contextual information," *IEEE*

Transactions on Mobile Computing, vol. 13, no. 7, pp. 1550–1563, 2013.

[7] J. Li, Y. Cai, Z. Cai, H. Leung, and K. Yang, "Wikipedia based short text classification method," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 275–286, Springer, Berlin, Germany, 2017.

[8] M. Wang, L. Lin, J. Wang, P. Yu, J. Liu, and F. Xie, "Improving short text classification using public search engines," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 157–166, Springer, Berlin, Germany, 2013.

[9] A. Z. Broder, M. Fontoura, E. Gabrilovich, A. Joshi, V. Josifovski, and T. Zhang, "Robust classification of rare queries using web knowledge," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07*, pp. 231–238, Amsterdam, The Netherlands, July 2007.

[10] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*, MIT Press, Cambridge, MA, USA, 2006.

[11] Z.-H. Zhou and M. Li, "Semi-supervised learning by disagreement," *Knowledge and Information Systems*, vol. 24, no. 3, pp. 415–439, 2010.

[12] H. Ma, H. Cao, Q. Yang, E. Chen, and J. Tian, "A habit mining approach for discovering similar mobile users," in *Proceedings of the 12—21st Annual Conference on World Wide Web*, pp. 231–240 WWW', Lyon, France, April 2012.

[13] X. Li, Y. H. Lian, and H. Yu, "Classification of mobile APPs with combined information," in *Proceedings of the 2016 IEEE International Conference on Cloud Computing and Big Data Analysis, ICCCBDA 2016*, pp. 193–198, Chengdu, China, July 2016.

[14] M. Garg, A. Monga, P. Bhatt, and A. Arora, "Android app behaviour classification using topic modeling techniques and outlier detection using app permissions," in *Proceedings of the 2016 4th International Conference on Parallel, Distributed and Grid Computing, PDGC 2016*, pp. 500–506, Waknaghat, India, December 2016.

[15] G. Berardi, A. Esuli, T. Fagni, and F. Sebastiani, "Multi-store metadata-based supervised mobile app classification," in *Proceedings of the ACM Symposium on Applied Computing*, pp. 585–588, Salamanca, Spain, April 2015.

[16] N. Chen, S. C. H. Hoiy, S. Li, and X. Xiao, "SimApp: a framework for detecting similar mobile applications by online kernel learning," in *Proceedings of the WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pp. 305–314, Shanghai, China, January 2015.

[17] B. Olabenjo, "Applying naive bayes classification to google play apps categorization," 2016, https://arxiv.org/abs/1608.08574.

[18] V. Radosavljevic, P. Chen, M. Grbovic et al., "Smartphone app categorization for interest targeting in advertising marketplace," in *Proceedings of the International Conference Companion on World Wide Web*, pp. 93-94, Montreal, Canada, April 2016.

[19] L. Jiang, C. Li, S. Wang, and L. Zhang, "Deep feature weighting for naive bayes and its application to text classification," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 26–39, 2016.

[20] L. Jiang, S. Wang, C. Li, and L. Zhang, "Structure extended multinomial naive Bayes," *Information Sciences*, vol. 329, pp. 346–356, 2016.

[21] K. Nigam, A. K. Mccallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[22] L. Jiang and C. Li, "Learning instance weighted naive Bayes from labeled and unlabeled data," *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 257–268, 2012.

[23] R. Johnson and T. Zhang, "Semi-supervised convolutional neural networks for text categorization via region embedding," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 919–927, Montreal, Canada, December 2015.

[24] K. Nigam, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the International Conference on Information & Knowledge Management*, McLean, VA, USA, November 2000.

[25] X. Wan, "Co-training for cross-lingual sentiment classification," in *Proceedings of the ACL-IJCNLP 2009—Joint Conference.of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 235–243, Stroudsburg, PA, USA, 2009.

[26] L. Shi, X. Ma, L. Xi, Q. Duan, and J. Zhao, "Rough set and ensemble learning based semi-supervised algorithm for text classification," *Expert Systems with Applications*, vol. 38, no. 5, pp. 6300–6306, 2011.

[27] D. J. Miller and H. S. Uyar, "A mixture of experts classifier with learning based on both labelled and unlabelled data," in *Advances in Neural Information Processing Systems*, pp. 571–577, MIT Press, Cambridge, MA, USA, 1997.

[28] V. Sindhwani and S. Keerthi, "Large scale semi-supervised linear SVMs," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 477–484, New York, NY, USA, August 2006.

[29] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," vol. 3175, no. 2004, Technical report, pp. 237–244, Carnegie Mellon University, Pittsburgh, PA, USA, 2002.

[30] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the of the 18th International Conference on Machine Learning*, pp. 19–26, Montreal, Canada, June 2001.

[31] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the International Conference on Machine Learning*, vol. 2, pp. 912–919, Washington, DC, USA, August 2003.

[32] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the Annual ACM Conference on Computational Learning Theory*, pp. 92–100, New Brunswick, NJ, USA, July 1998.

[33] Z.-H. Zhou and M. Li, "Tri-training: exploiting unlabeled data using three classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[34] H. Wu, R. Luk, K. Wong, and K. Kwok, "Interpreting TF-IDF term weights as making relevance decisions," *ACM Transactions on Information Systems*, vol. 26, no. 3, p. 13, 2008.

[35] X.-H. Phan, C.-T. Nguyen, D.-T. Le, L.-M. Nguyen, S. Horiguchi, and Q.-T. Ha, "A hidden topic-based framework toward building applications with short web documents," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 961–976, 2011.

[36] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.
[37] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014.
[38] https://code.google.com/p/word2vec/.
[39] M. Lv, L. Chen, T. Chen, and G. Chen, "Bi-view semi-supervised learning based semantic human activity recognition using accelerometers," *IEEE Transactions on Mobile Computing*, vol. 17, no. 9, pp. 1991–2001, 2018.
[40] V. Sindhwani and D. S. Rosenberg, "An RKHS for multi-view learning and manifold co-regularization," in *Proceedings of the International Conference on Machine Learning*, pp. 976–983, Helsinki, Finland, July 2008.