Extracting User-Reported Mobile Application Defects from Online Reviews

Yue Wang
University of Delaware
Newark, Delaware, USA
Email: wangyue@udel.edu

Hongning Wang
University of Virginia
Charlottesville, Virginia, USA
Email: hw5x@cs.virginia.edu

Hui Fang
University of Delaware
Newark, Delaware, USA
Email: hfang@udel.edu

***Abstract*—User-generated mobile application reviews have become a gold mine for timely identifying functional defects in this type of software artifacts. In this work, we develop a hidden structural SVM model for extracting detailed defect descriptions from user reviews at the sentence level. Structured features and constraints are introduced to reduce the demand of exhaustive manual annotation at the sentence level and enable the use of partially annotated review data for model training. Extensive empirical evaluations on a large collection of mobile application reviews collected from Apple App Store demonstrate the effectiveness of our proposed solution in recognizing the user-reported implementation defects from review content, especially when only partial annotation is available.**

I. Introduction

Recent years have witnessed an increasing popularity of mobile applications (or mobile apps in short). The same as any other types of software artifacts, mobile apps also suffer from implementation defects, which cause detrimental effects on both developers and users. Traditional software testing based solutions are time consuming [1]. Fortunately, users often describe the problems they encounter in the reviews [2]. These user-generated reviews thus become a gold mine for timely identifying mobile app functional defects.

Extracting informative descriptions of mobile app defects from user reviews is challenging. Typically, a review may cover various aspects of an app; but not necessarily all of its content is about the functional defects. In this work, we define *defects* as abnormal behaviors of an app which produce incorrect or unexpected results. A sample app review is shown in Figure 1. This review mentions two specific defects encountered after an update of the app, and a new function suggestion that is not related to any defect. Because of the unstructured nature of user reviews, these three parts are mixed in this review. A review-level classification solution might recognize this review mentions functional defects, but it is insufficient for end-users to digest the detailed defects. For example, an app developer still needs to read through the categorized reviews, figure out specific sentences describing the defects, and stitch relevant information from all the selected reviews. As a result, extracting user-reported defects at the sentence-level is necessary.

Sentences in a user review are not independent. As shown in Figure 1, consecutive sentences are more likely to describe the same defect. This also indicates the review content is self-consistent: it is very unlikely for a reviewer to describe a specific aspect as both problematic and normal. Building

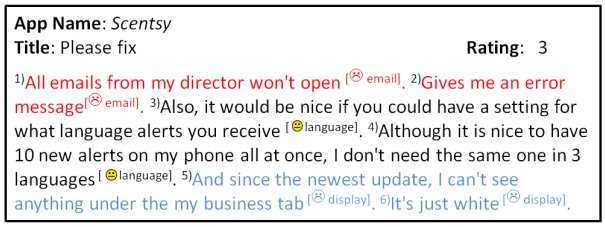

Fig. 1. A sample mobile app review from Apple App Store. Aspects and sentence-level defect labels are manually annotated in the superscripts with different colors.

a sentence-level classifier [3] ignore such dependency and therefore suffer from suboptimal performance. For example, if one only looks at the sixth sentence in the sample review, it is very hard to judge whether it is reporting a defect. But with its preceding sentences, the conclusion becomes clear that the white business tab is unexpected. In addition, sentences mentioning app defects also contribute to the overall sentiment polarity of the review, which is often reflected in the given overall ratings. As a result, only when integrating all these aforementioned dependencies, an effective solution could be derived to extract useful sentence-level descriptions of mobile app defects from user reviews.

In this work, we develop a structured learning solution based on the hidden structural support vector machines [4] to exploit the dependency among sentences for identifying app defects in user reviews. The proposed method has two unique advantages. First, it exploits the consistency and transitional structures among sentences in the same review. Predictions on sentences thus become dependent, which addresses the challenges and serious bottleneck of inferring the labels in short sentences, which usually take a major proportion in mobile app reviews. Second, because of this explicit dependency modeling, the proposed model can be estimated from partially annotated reviews, where not all sentences need to be annotated. This is of particular importance in practice, as obtaining sentence-level annotations in reviews is time-consuming and expensive.

Our empirical evaluations on a large collection of mobile app reviews collected from Apple App Store demonstrated the effectiveness of the proposed solution in recognizing the user-reported implementation defects from review content, especially when only partially annotated training data is available. We compared the identified defects against the release history of several popular apps, and found that in most cases our

method correctly identified defects right after new releases. This confirms the practical value of the proposed solution.

II. Related work

This paper focuses on extracting sentences which are related to functional defects of mobile apps from user-generated reviews. Several directions of previous studies directly relate to our work.

Mobile app review mining: Mobile app reviews have been explored in various applications, including app recommendation [5] and mobile app retrieval [6]. Some recent works examine the utility of such user-generated content in software engineering area. For example, Chen et.al. ranked reviews based on the criterion of whether a review mentions app defects or makes new function suggestions [7]. Maalej and Nabil classified reviews as defect reports, feature requests, user experiences, and ratings [8]. However, these studies only focused on review-level analysis. Given a popular app can easily receive thousands of reviews after each release, the end-users still need to read through those selected reviews to figure out the detailed defects or suggestions.

There are studies identifying user-reported mobile app defects at the sentence level. Moghaddam proposed a distant learning method to extract user-reported defects and improvement suggestions from app reviews [9]. The reported experiments showed that a simple linear classifier with bag-of-word features worked the best. But because this empirical study was only performed on reviews of a single app, the conclusion is potentially biased. Panichella et. al. classified sentences in mobile app reviews into more categories: feature request, problem discovery, information seeking, and information giving [3]. They introduced a rich set of features including linguistic features, text features, and sentiment features. However, they considered sentences as independent, where features were constructed for each sentence individually. As we discussed in the introduction, this design unfortunately loses important structural information in a user review and leads to suboptimal extraction performance, especially on those short sentences. Furthermore, in those existing solutions, detailed and exhaustive sentence-level annotation is inevitable to estimate the sentence classifiers, which increases the overhead of applying such solutions in practice.

Software defect identification: Software defect report has been extensively studied as a useful resource to help developers improve their product in software engineering. Most of the efforts have been focused on automatically classifying defect report [10], determining defect severity level [11], and predicting the effort to fix a defect [12]. A few existing work studied identifying defect reports from user-generated text content. Antoniol et. al. [13] introduced a text mining based method to identify different types of issues in a defect tracking system. Gegick et. al. [14] studied how to automatically identify security defect report, which is considered as a special and important type of defect report. But little has been done in user review data, which is featured with open purposes and free text format. Besides, all of these studies targeted in the development/testing stage of software. This is different from our problem setup, as we aim to identify defects from user reviews after the mobile apps have been released.

III. Method

Formally, the input of our problem is a user-generated review $R = \{s_1, s_2, \ldots, s_n\}$, consisting of n sentences, among which one or more sentences provide details about functional defects of the app. In this work, we define defect as the abnormal behavior of an app which produces an incorrect or unexpected result, or behaves in unintended ways. Therefore, the learning problem is to infer the defect label h_i on sentence s_i, which indicates whether the sentence contains information about any functional defect of the app. To facilitate our discussion, we define a review-level binary variable y to indicate whether the review contains any sentence mentioning about defects, i.e., $y = 1$ if at least one $h_i = 1$.

Based on the above definition and notations, we describe the proposed hidden structured learning solution, the developed features, and the constraint-driven parameter estimation method in the following subsections.

A. A structured learning solution

A straightforward solution to extract sentence-level defect descriptions is to manually annotate sentences in a set of training reviews, and build a supervised classifier over those sentences. However, such a classifier would ignore the dependency among sentences by classifying each sentence individually. In addition, the construction of a fully labeled training set is labor-intensive, as annotators have to make inference across all sentences in a review to determine the detailed sentence-level labels. To address these challenges, we propose a structured learning solution, which explicitly models the dependency among sentences in a review, such that annotations can be propagated among labeled and unlabeled sentences, and model training can be naturally performed on partially annotated data sets.

Specifically, we formalize our solution within the hidden structural SVM framework [4]: we consider the sentence-level defect labels as hidden variables, which contribute to review-level defect labels. The problem can then be formulated as a joint inference problem over both the sentence-level and review-level labels in the following way:

$$(\hat{y}, \hat{H}) = \underset{(y,H)\in\mathscr{Y}\times\mathscr{H}}{\arg\max}\ \omega^\top \Phi(R, H, y) \tag{1}$$

where $\mathscr{Y}$ and $\mathscr{H}$ are the search spaces for review label y and sentence labels $H = \{h_1, h_2, \ldots, h_n\}$. $\Phi(R, H, y)$ is a structured feature vector depicting the matching quality of those labels with respect to the review content, and ω is the corresponding model parameter indicating the importance of features when predicting the defects. As a result, we refer to our structured learning solution as *hSVM* in this paper.

The inference problem defined in Eq (1) does not consider the structural dependency between sentence-level defect labels and review-level labels. As we discussed in our problem definition, a review to be considered as defect report should contain at least one sentence indicating a specific defect. To

enforce this, we introduce the following constraint into the inference problem defined in Eq (1),

$$\sum_{i=1}^{n} \hat{h}_i \geq \hat{y} \tag{2}$$

However, because of this constraint, the inference problem becomes computationally intractable (as the sentence-level predicted labels become dependent on each other, no matter what kind of structured features will be there in Eq (1)). We formulate this constrained inference problem as an integer linear programming (ILP) problem [15], and solve it by an off-the-shelf optimization package.

The key to the success of our proposed solution is thus the design of structured features $\Phi(R, H, y)$ and the estimation of corresponding feature weight ω. When designing the features, we need to include not only sentence-level but also review-level information that can capture the consistency and transition structures among sentences. When estimating the model parameters, to overcome the difficulty of constructing a fully annotated training set, we focus on how to estimate the model from partially annotated data, i.e., reviews with only review-level annotations or those with only partially labeled sentence-level annotations.

B. Structured feature design

In order to capture the dependency among sentences, our solution leverages both sentence-level and review-level features. In the sentence-level features, we consider the textual content of sentences, the topics of them, as well as the similarity between sentences and related app descriptions. In the review-level features, we leverage information from the observed sentiment ratings, the first-order transitions and pairwise consistency between sentences and their defect labels. We summarize all the features in Table I and describe them in more details by grouping them into different categories.

1) Sentence-level features: Sentence-level features focus on exploiting information conveyed in the content of a sentence. Since a sentence can be considered as a short review, we can directly utilize features developed for in review-level defect detection works [8]. In addition, we train a review-level classifier based on multiple content-based features, and use the classifier's output on a sentence as one sentence feature, i.e., $\Phi_{s\_content}(s_i, h_i, y)$. Moreover, we notice that users tend to use a shared set of words to describe defects, such as "freeze," "idle," and "crash". To capture this, we manually compile a list of indicator words by looking through the frequent words in our annotated defect reporting sentences. We use the number of indicator words matched in a sentence as another sentence feature, which is denoted as $\Phi_{indicator}(s_i, h_i, y)$.

We observe that a user tends to mention one or more specific aspects of the app if it is defective. Thus, whether a sentence mentions a specific aspect of an app becomes an important indicator for functional defect reporting. Because the topical aspects of mobile apps are not directly observable in the review content, we train a PLSA topic model [16] to recognize them. Specifically, the PLSA model is estimated based on the concatenated reviews from all apps under the same category. Each sentence in a target review is assigned to the most similar aspect. We denote this feature as $\Phi_{aspect}(s_i, h_i, y)$.

When building features for a sentence, we also include the similarity between the given sentence to the other reviews of the same app and version as a sentence-level feature. The design behind this feature is that if the sentence is describing certain defect, it is very likely that the other reviewers would also report the same defect, such that we should expect higher concentration of similar mentions. We denote this feature as $\Phi_{high\_sim}(s_i, h_i, y)$. Based on a similar reason, we also utilize the average rating of the reviews similar to the target sentence as another sentence-level features (ratings are weighted by the corresponding content similarities). A smaller value of this feature indicates the other similar reviews give a lower overall rating to this app; and as a functional defect usually leads to lower overall ratings, this sentence is more likely to describe a defect. We denote this feature as $\Phi_{avg\_rating\_sent}(s_i, h_i, y)$.

2) Review-level features: We develop a set of review-level features, including the overall rating of this app and the average rating of this version of this app. Besides, we also include the highest similarity value between the target review and the other reviews of the same app and same version as one of the features. Due to space limit, we skip the detailed discussions about such review-level features, but concentrate on the review-level structured features.

We first incorporate the first-order transition feature $\Phi_{trans}(s_i, s_{i+1}, h_i, h_{i+1}, y)$ to capture the changes of a reviewer's focus when describing the app. As shown in Figure 1, users tend to discuss the same defect in consecutive sentences. This transition feature is designed to capture such local transition pattern in users' defect descriptions. And to differentiate the potential different patterns in defect reporting reviews and non-defect reporting reviews, we also include the predicted review-level label in this feature.

We also develop pairwise features to check the consistency between the predicted sentence-level defect labels. To be specific, if a pair of sentences in the same review shares the same indicator word, the predicted sentence-level defect labels are supposed to be the same for these two sentences. Similarly, if two or more sentences describe the same aspect, the labels for these sentences should also be the same. We denote these two features as $\Phi_{indicator\_cons}(s_i, s_j, h_i, h_j, y)$, and $\Phi_{aspect\_cons}(s_i, s_j, h_i, h_j, y)$, which are also related to the predicted review-level defect label y.

C. Parameter Estimation

With a properly annotated training collection, the model parameters in hSVM could be effectively estimated. In this work, we derived our model learning method from the latent structural SVMs framework [17] as:

$$\begin{aligned} &\min_{\omega,\xi} \frac{1}{2}\|\omega\|^2 + C\sum_{m=1}^{M}\xi_m \\ s.t.\ \ &\forall m, \max_{H\in\mathscr{H}} \omega^\top \Phi(R_m, H, y_m) \geq \\ &\max_{(\hat{y},\hat{H})\in\mathscr{Y}\times\mathscr{H}} \omega^\top \Phi(R_m, \hat{H}, \hat{y}) + \Delta(y_m, \hat{y}, H_m, \hat{H}) - \xi_m \\ &\xi_m \geq 0 \end{aligned} \tag{3}$$

Type	Feature	Description
Sentence level features	$\Phi_{indicator}(s_i, h_i, y)$	The number of matched indicator words in sentence s_i
	$\Phi_{constrast}(s_i, h_i, y)$	Whether sentence s_i contains any contrast conjunction
	$\Phi_{sentiment}(s_i, h_i, y)$	The sentiment orientation of sentence s_i
	$\Phi_{length}(s_i, h_i, y)$	The length of sentence s_i
	$\Phi_{negation}(s_i, h_i, y)$	Whether sentence s_i contains any negation
	$\Phi_{stopword}(s_i, h_i, y)$	The proportion of stopwords in sentence s_i
	$\Phi_{title}(s_i, h_i, y)$	Whether sentence s_i is the title of the review
	$\Phi_{last}(s_i, h_i, y)$	Whether sentence s_i is the last sentence
	$\Phi_{aspect}(s_i, h_i, y)$	Whether sentence s_i mentions a particular aspect
	$\Phi_{s\_content}(s_i, h_i, y)$	The output of content-based review-level classifier for sentence s_i
	$\Phi_{high\_sim}(s_i, h_i, y)$	The highest similarity between sentence s_i and other reviews of the same app and version
	$\Phi_{avg\_rating\_sent}(s_i, h_i, y)$	The average overall rating from reviews similar to sentence s_i
	$\Phi_{std\_rating\_sent}(s_i, h_i, y)$	The standard deviation of overall ratings from reviews similar to sentence s_i
	$\Phi_{length\_disc}(s_i, h_i, y)$	The discretized length of sentence s_i
Review level features	$\Phi_{rating}(R, y)$	The overall rating of the app which review R is about
	$\Phi_{r\_content}(R, y)$	The output of the content-based classifier for review R
	$\Phi_{aspect\_c}(R, y)$	The aspect coverage in review R
	$\Phi_{sim\_doc}(R, y)$	The highest similarity between review R and other reviews for the app and version
	$\Phi_{avg\_rating\_doc}(R, y)$	The average overall rating of the version which review R is about
	$\Phi_{trans}(s_i, s_{i+1}, h_i, h_{i+1}, y)$	Transition between defect labels of consecutive sentences
	$\Phi_{aspect\_cons}(s_i, s_j, h_i, h_j, y)$	Whether a pair of sentences mentioning the same aspect have the same defect label
	$\Phi_{indicator\_cons}(s_i, s_j, h_i, h_j, y)$	Whether a pair of sentences containing the same indicator word share the same defect label

TABLE I

STRUCTURED FEATURES FOR HSVM. SENTENCE-LEVEL FEATURES CHARACTERIZE THE ASSOCIATION BETWEEN A SINGLE SENTENCE AND DEFECT LABEL, AND REVIEW-LEVEL FEATURES DEPICTS DEFECT LABELS IN A PAIR OF SENTENCES AND IN THE WHOLE REVIEW.

where $\Delta(y_m, \hat{y}, H_m, \hat{H})$ captures the structural loss caused by the predicted labels $(\hat{y}, \hat{H})$ against the ground-truth labels (y_m, H_m), $\{\xi_m\}_{m=1}^{M}$ is a set of slack variables to allow errors in the training data, and C controls the trade-off between empirical training loss and model complexity.

The design of $\Delta(y_m, \hat{y}, H_m, \hat{H})$ is a key component in our model estimation, as it represents the quality of predicted labels with regard to the ground-truth. If the training data set is fully annotated, any form of structured loss can be utilized, such as hamming distance between to label vectors [18]. But in our problem, as exhaustive sentence-level annotation is labor-intensive and not feasible at scale, we relax the loss function on sentence-level labels, i.e., no penalty is incurred in a sentence if it is unlabeled. But to capitalize on the reviews that only have review-level annotations, we also utilize the constraint defined in Eq (2) to punish the learned model if the inferred sentence-level labels contradict with the known review-level labels. We should note as our training procedure only requires a partially annotated collection, when sentence-level annotations are available, the original structural loss will be computed. In our experiment, we utilized standard hamming distance to materialize $\Delta(y_m, \hat{y}, H_m, \hat{H})$, but other types of structured loss are also applicable in our model (with potentially different inference complexity). In both cases, the required inference problems in Eq (3), i.e., $\max_{H \in \mathscr{H}} \omega^\top \Phi(R_m, H, y_m)$ and $\max_{(\hat{y}, \hat{H}) \in \mathscr{Y} \times \mathscr{H}} \omega^\top \Phi(R_m, \hat{H}, \hat{y}) + \Delta(y_m, \hat{y}, H_m, \hat{H})$, can be effectively solved by the same ILP solver. More specifications can be found in [15].

IV. EXPERIMENT DESIGN

We crawled more than 753K apps from Apple App Store in the period from November 2013 to December 2013, via its search API. For each app, we collected information including its name, official description, developer, category and version history. We used the RSS feed provided in the App Store to fetch the reviews for a given app. 9 graduate students with computer science background were recruited to perform the annotation task, i.e., labeling whether a review or a sentence mentions any specific mobile app defect. When the annotators judged the reviews/sentences, the description of the app was provided. They were required to read the description and understand the functionality of the app before annotations.

A. Training the content-based classifier

To construct the training data set for this classifier, we first randomly selected 1,200 reviews, and each one of them was manually annotated at the review level to indicate whether it mentions a particular defect. The annotators were instructed to label a review as a defect report if at least one of its sentences mentioned a functional defect of the app. Among all the reviews, 355 were labeled as positive, i.e., defect reports, and 845 as negative.

With the annotated reviews, we trained a binary Random Forest classifier. The classifier used the following features: unigram and bigram textual features, sentiment score of the review based on SentiWordNet, review length, and overall rating of the review. ChiSquare statistics and information gain were used to select the textual features. The trained classifier performed well with a precision of 1.00 and recall of 0.89 in a 5-fold cross validation on this annotated data set.

B. Sentence-level annotations

Since we focus on sentence-level defect detection, it is necessary to obtain sentence-level annotations for evaluation. Because the number of sentences to be annotated is very large in our collected data set, it is impossible to exhaustively label all of them. Instead, we considered only the apps that contain more than 40 reviews. We then applied the content-based classifier to the reviews from these apps. For each app, we sorted the reviews based on how likely a review mentions at least one defect of the app. We then selected the top 5 reviews with the highest prediction score, and then 5 random reviews from the each quarter of the ranked list (except the top 5). This

stratified sampling method is to ensure that the selected 25 reviews are not biased toward positive or negative examples.

	Fully labeled	Partially labeled
# of Reviews (defect / non-defect)	274 / 572	400 / 818
# of Sentences (defect / non-defect)	557 / 3917	84 / 441
Unlabeled sentences	0	8968

TABLE II
BASIC STATISTICS OF ANNOTATED DATA SETS FOR MOBILE APP DEFECT DETECTION.

We randomly selected 15 apps for each annotator, and every review was assigned to 3 different annotators. The annotators were asked to provide annotation for each sentence separately together with a review-level defect label. We showed the whole review with sentences in their original order to the annotators, rather than the individual sentences. The annotators were asked to annotate each sentence as "defect report", "not defect report", or "unknown". We took majority vote from the three annotators as the final label of each sentence. We denoted the reviews, in which every sentence had a determined label either of "defect report" or "not defect report", as **fully labeled reviews**. Some sentences might not have a consented label due to the "unknown" option for annotation. We put these reviews together with the 1200 reviews that only contain review-level annotations into one group, and denote them as **partially labeled reviews**. As a result, the reviews in the partially labeled data set contain at least one unlabeled sentence. The statistics of reviews and sentences of our annotated data set is demonstrated in Table II.

V. EXPERIMENT RESULTS

We conduct experiments to evaluate the effectiveness of the proposed hSVM model on detecting sentences that report mobile app defects from user-generated reviews. Standard classification performance metrics, including precision, recall and F1 score, are used in our evaluations. When reporting the performances, we use the superscripts of R, L, J and S to indicate the improvement over RF, LR, J48 and SVM is statistically significant based on t-test, respectively.

A. Using review-level classifier for sentence-level prediction

The first set of experiments are designed to examine whether a classifier trained from the review-level annotations is effective for the sentence-level detection.

	Prec	Recall	F1
RF-R	0.651	0.544	0.593
LR-R	0.584	0.572	0.578
SVM-R	0.633	0.536	0.580
MIL-R	0.647^{L}	0.572	0.607
hSVM-R	$\mathbf{0.784}^{R,L,S}$	$\mathbf{0.641}^{R,L,S}$	$\mathbf{0.705}^{R,L,S}$

TABLE III
PERFORMANCE OF REVIEW-LEVEL CLASSIFIERS ON SENTENCE-LEVEL PREDICTION.

In addition to the review-level classifier described in Section IV-A, we also trained logistic regression and SVM classifiers for comparison. These three methods are denoted as *RF-R*, *LR-R* and *SVM-R* respectively, where R indicates the classifiers are trained only using the review-level annotations. We also included a state-of-the-art multi-instance learning method developed in [19], which ensembled the aforementioned three review-level classifiers. And we refer to it as *MIL-R*. We also trained our hSVM model with only review-level annotations to understand how well our method could perform when only review-level annotations are available. All these methods are tested on the fully labeled reviews. We conducted 5-fold cross validation and reported the results in Table III. It is clear that directly applying a classifier trained for the review-level detection cannot provide satisfactory performance for extracting sentence-level defect descriptions. The best precision of the baseline methods is only 0.651, which would introduce too many false positives and increase unnecessary burden for an end-user to digest the detection results. Our method outperformed the baselines, with a precision of 0.784 and a recall of 0.641, which indicates the necessity of exploring dependency among sentences in this particular problem.

B. Performance comparison on fully annotated data

The second set of experiments is to compare the proposed method against sentence-level classification baselines. All the methods were trained with sentence-level annotations, and the experiments were conducted over the fully-annotated reviews, i.e., all the sentences in these reviews have been annotated.

The first baseline is a state-of-the-art sentence-level defect detection method proposed in [3], denoted as *J48-S*. It used a combined feature set including NLP features, text features and sentiment features, with a J48 decision tree classifier. The other baselines are similar to the methods described in Section IV-A except they are trained with the sentence-level annotations. These methods are denoted as *RF-S*, *LR-S* and *SVM-S* accordingly.

	Prec	Recall	F1
J48-S	0.716	0.863	0.783
LR-S	0.799	0.811	0.805
SVM-S	0.785	0.832	0.808
RF-S	$0.882^{J,L,S}$	$\mathbf{0.947}^{J,L,S}$	$\mathbf{0.913}^{J,L,S}$
hSVM	$\mathbf{0.919}^{J,L,S}$	$0.898^{L,S}$	$0.908^{J,L,S}$

TABLE IV
PERFORMANCE COMPARISON AMONG SENTENCE-LEVEL DEFECT DETECTION METHODS.

We performed 5-fold cross-validation to reduce the randomness in training/testing split, and the average performance is reported in Table IV. It is clear that our proposed *hSVM* model outperformed most of the baseline methods. The improved performance of *hSVM* stems from its capability in utilizing not only content-based features but also structured features that capture the consistency and transition patterns among sentences. We also notice that *RF-S* achieved better recall than *hSVM* model (and therefore its F1 score is better). This is mostly due to the non-linearity of the random forest model, which helps the model capture more complex relations between sentence-level features and labels. However, as we will show in the following experiments, *RF-S* relies heavily on the availability of annotations and its performance degenerates significantly when only partially labeled data is available.

As we emphasized in the introduction, the sentences in mobile app reviews are usually very short (the average sentence length in our collection is 13.72, with a median of 10).

Therefore, it is worth our effort to compare different models' performance in recognizing defects among short sentences. We reported the classification performance on the top 25% shortest sentences in the fully annotated data set in Table V.

	Prec	Recall	F1
J48-S	0.672	0.826	0.741
LR-S	0.761^{J}	0.792	0.776
SVM-S	0.763^{J}	0.814	0.781
RF-S	$0.854^{J,L,S}$	$0.893^{J,L,S}$	$0.873^{J,L,S}$
hSVM	$\mathbf{0.912}^{J,L,S,R}$	$\mathbf{0.931}^{J,L,S}$	$\mathbf{0.921}^{J,L,S,R}$

TABLE V
PERFORMANCE COMPARISON AMONG SENTENCE-LEVEL DEFECT DETECTION METHODS ON THE TOP 25% SHORTEST SENTENCES.

The results show that our hSVM model outperformed all the baselines on those short sentences. As we discussed in our running example, short sentences usually suffer from insufficient feature representation, such that independent sentence-level classifiers can hardly make any reasonable prediction on them. Our structured learning solution utilizes the consistency and transition features to exploit the dependency among sentences. The label of a short sentence is not solely inferred by the sentence itself, but also the sentences surrounding it and in the same review.

C. Performance comparison on partially annotated data

Our model is able to utilize partially annotated data for parameter estimation. This would lead to considerably reduction of annotation effort in practice. This experiment verifies the effectiveness of our model training in this scenario. We only used 20% of the fully labeled reviews as our training set, and left the remaining 80% for testing purpose. Then we gradually added the partially annotated reviews into the training set. For the sentence-level baselines, when the sentence-level annotation is totally missing in a review, we will simply treat review-level labels as the sentence-level labels. Otherwise, if some sentences in a review are labeled, we will ignore the other unlabeled ones in that review for model training. To avoid any bias resulted from the data set separation, we repeated the experiments 5 times by randomly shuffling the fully labeled reviews for training and testing. The average F1 and the standard deviation of classification performance across all sentence-level classifiers are reported in Figure 2.

From the results, we can clearly notice that hSVM model could quickly converge to its optimal performance even with limited training instances. This proves that our model could utilize the structural dependency between sentence-level and review-level labels to better estimate the model parameters. In addition, the variance of our method was much smaller than that in most of the baselines, which indicates that our learned model is less likely to overfit to a particular training data set.

It is reported by our annotators that providing the review-level labels and partial sentence-level labels is much easier than the exhaustive sentence-level labels. Therefore, it is worthwhile to further test different algorithms' performance with only the partially annotated data is available for training. To be more specific, we trained our model only with the partially annotated reviews (including both review-level labels and partial sentence-level labels) in the same way as in the last

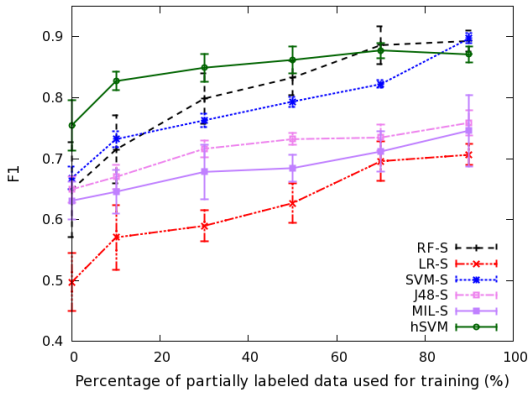

Fig. 2. Classification performance with gradually added partially labeled reviews for model training. The error bar indicates the standard deviations.

experiment, and applied the trained model on the fully labeled data set for testing. The fully labeled data set has been split into 5 equal subsets only for the testing purpose. The average precision, recall and F1 score are reported in Table VI.

	Prec	Recall	F1
J48-S	0.641	0.775^{L}	0.701^{L}
LR-S	0.649	0.643	0.652
SVM-S	0.749^{L}	0.804^{L}	0.774^{L}
RF-S	$0.824^{J,L,S}$	0.789^{L}	$0.806^{J,L}$
hSVM	$\mathbf{0.851}^{J,L,S}$	$\mathbf{0.846}^{J,L,S,R}$	$\mathbf{0.827}^{J,L,S}$

TABLE VI
PERFORMANCE COMPARISON TRAINED ON PARTIALLY ANNOTATED DATA SET AND TESTED ON FULLY LABELED DATA SET.

We can clearly notice that our proposed method outperformed all the baselines. The major reason for this performance improvement is that the baselines cannot fully utilize the unlabeled sentences in model training, and in testing phase the sentences are independently classified. On the contrary, our method utilized the connection of between the sentences, i.e., the first-order transition feature and consistency feature, to propagate the label information for model training and regularize the predictions in testing instances.

Since the partially annotated data contains two types of annotations (i.e., the ones with only review-level annotations, and those also with partial sentence-level annotations), we also studied the effectiveness of different types of partially annotated data for model training. Specifically, we first randomly split the fully annotated data into 5 folds. Three folds were used as the initial training set, and one fold was kept for testing. For the remaining fold, we randomly removed some sentence-level annotations, and gradually added these reviews to the training set. To compare with this setting, we added the same set of reviews with only review-level annotation to create another training set (by removing their entire sentence-level labels). We performed the experiments 5 times by switching the folds for training and testing, and each time we randomly created the partially annotated reviews to be added into the training set. The mean and standard deviation of F1 score are reported in Figure 3. The methods labeled with "-sentence" indicate the models that used both review-level and sentence-level annotations as the additional training instances, while the

ones with "-review" indicate those only used the review-level annotations as their additional training instances.

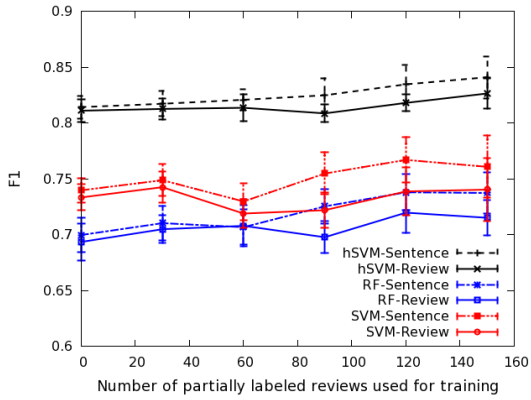

Fig. 3. Classification performance improvement when adding different types of partially labeled reviews for training. The error bar indicates the standard deviations.

As we can expect that in all models the partially annotated reviews with sentence-level annotations were more helpful than those with only review-level annotations in boosting models' classification performance. In particular, only our model achieved consistent improvement from both types of partially annotated reviews when increasing amount of such training data became available, while the baselines' performance varied quite a lot. One major reason is that the performance of baselines heavily depends on what types of sentences are fully labeled and their annotation quality for model training. In other words, they are more sensitive to the partially annotated data set, and it therefore imposes more constraints on the manual annotation process. Because our model is more robust to such partially annotated data, it further reduces the manual annotation cost. This is of particular importance for applying our solution in practice.

D. Effectiveness of structured features

In order to better understand the contribution of different features in our structured learning solution, we conducted experiments to evaluate our model under different configurations of features with repeated the five-fold cross validations. Table VII summaries the results.

	F1	Prec	Recall
SLF only	0.904	0.912	0.897
Textual features only	0.451	0.621	0.354
SLF+Textual	0.874	0.915	0.837
SLF+Textual+Consistence	0.890	0.924	0.858
SLF+Textual+Transition	0.893	0.916	0.872
All features	0.908	0.919	0.898

TABLE VII
EFFECTIVENESS OF DIFFERENT FEATURE CONFIGURATIONS IN HSVM, WHERE SLF STANDS FOR SENTENCE-LEVEL FEATURES.

Comparing the first two rows in Table VII, it demonstrates that our model with our developed sentence-level features could perform much better than that with the traditional textual features. It suggests that not only the actual words used in the review, but also the other associated signals, such as sentence length and sentiment orientations, should also be considered when estimating the model. When adding the consistency feature and transition feature into the model (5th and 6th rows in the table), they further improved the performance. This shows that the dependency between the sentences helps better predict sentence-level defect labels.

Comparing the first row and the last row in Table VII, we could observe that only using the sentence-level features already achieved satisfactory performance. Thus, it is necessary to further analyze which sentence-level feature plays the most important role in sentence-level defect prediction. In order to get this level of understanding, we iteratively removed one feature from the sentence-level feature set and test the performance with the remaining features. We used the fully labeled dataset with 5-fold cross validation in this experiment. The difference of the F1 performance against the full feature set is reported in Table VIII. The negative value means the F1 score decrease when the corresponding feature is removed from the feature set. Due to the space limit, we only list the top five features that are most helpful (i.e., with the most F1 decrease when the feature is removed).

Feature	F1(difference)
$\Phi_{indicator}(s_i, h_i, y)$	-0.113
$\Phi_{aspect}(s_i, h_i, y)$	-0.097
$\Phi_{sentiment}(s_i, h_i, y)$	-0.071
$\Phi_{constrast}(s_i, h_i, y)$	-0.063
$\Phi_{negation}(s_i, h_i, y)$	-0.062

TABLE VIII
PERFORMANCE CHANGES WHEN ONE FEATURE IS REMOVED FROM THE FULL FEATURE SET OF HSVM.

It is not surprising to observe that the performance decreased the most when the indicator words based feature is removed from the feature set. This shows that our manually compiled indicator words capture the most discriminative pattern in users' defect descriptions. The performance would also decrease when the aspect feature was removed. This confirms our assumption that users tend to be more specific on particular aspect when reporting the defects in their reviews. In addition, sentiment orientation is also a good indicator for identifying the defect reporting sentences.

E. Qualitative studies

Identifying the defect reporting sentences and connect them to the functions is a key step for solving the defects for the developers. In Figure 4, we demonstrate how the developers could benefit from our system. Figure 4 shows the update history, word cloud visualization of aggregated review content, and detected defect reporting sentences of one randomly selected app "*Microsoft Tag*". In the word cloud, we also highlighted the words that frequently appeared in the recognized defect reporting sentences. The number of total reviews for each version is labeled next to the word cloud. In particular, based on the official description of this app, three new functions were introduced in the version 5.5, i.e., bar code scanning, QR code scanning, and save the information from the scanned bar code and QR code. Based on the identified defect-reporting sentences in user reviews, we can notice that these three new features caused unexpected

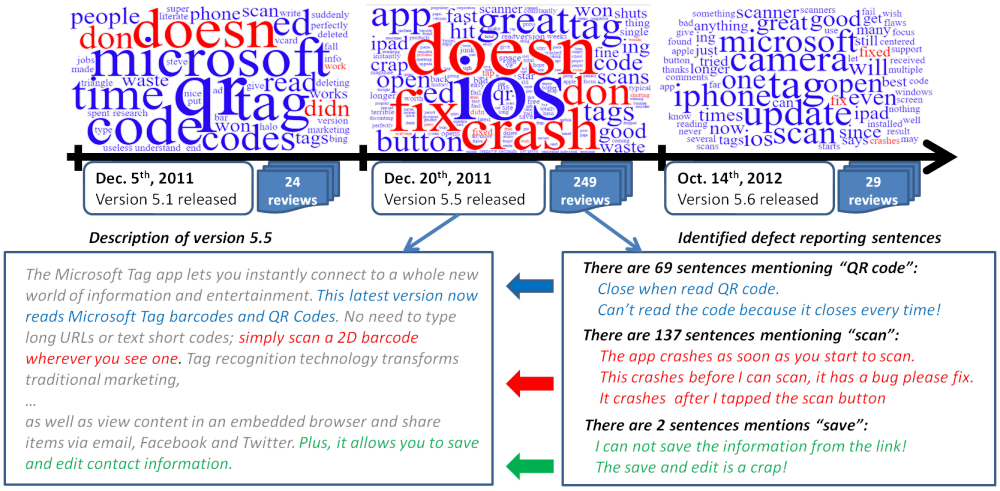

Fig. 4. A visualization of identified sentence-level defect reports from hSVM. The word clouds summarize the reviews of the this app in each specific version. The description of version 5.5 and the corresponding defect reporting sentences identified by hSVM model are shown at the bottom of the figure.

crash of the app. Our solution identified such defects from the reviews, and based on it we can effectively summarize the identified sentences for the developers to digest. In this study, we first extracted the specific function descriptions from the official app description, then we clustered the defect reporting sentences with respect to these functions. In the bottom-right part of Figure 4, we highlighted the identified sentence-level defect reports with respect to each specific new feature of this app. This clearly provides developers a more concrete and comprehensive understanding of the functional defect of the app, with different use case scenarios and error descriptions.

In addition, we can find that the recognized defect report about this particular issue immediately emerged right after the release of its version 5.5 and our method can accurately recognized these defects. This clearly demonstrates our proposed solution could recognize the defects from user reviews in timely fashion. It took the developers 10 months to identify and fix the defects in version 5.5, our method could shorten this maintenance cycle by accelerating the defect identification procedure for the developers.

VI. Conclusion and Future Work

We proposed a structured learning solution to address the problem of identifying sentence-level description of mobile app defects in user reviews. Structured features and constraints are developed to capture the dependency among sentences in a review. This reduces our model's dependency on fully annotated training data and improves its extraction accuracy on short sentences, which make a major portion of user-generated review content.

Our work opens several interesting future directions. First, it is meaningful to investigate how to incorporate unlabeled data into model training, which can further reduce the manual effort on providing the training instances. Second, summarizing the extracted sentences and mapping them to the detailed features of an app is desirable the end-users. As shown in the Figure 1, assigning the defect reporting sentences to the corresponding aspect would be more helpful. Last but not least, an app's version updating history can be used as remote signals for model training.

Acknowledgment

This research was supported by the U.S. National Science Foundation under IIS-1423002.

References

[1] D. E. Harter, M. S. Krishnan, and S. A. Slaughter, "Effects of process maturity on quality, cycle time, and effort in software product development," *Management Science*, 2000.
[2] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *KDD 13*, 2013.
[3] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, ""how can i improve my app? classifying user reviews for software maintenance and evolution"," in *ICSME '15*, 2015.
[4] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *ICML '09*, 2009.
[5] H. Zhu, H. Xiong, Y. Ge, and E. Chen, "Mobile app recommendations with security and privacy awareness," in *KDD '14*, 2014.
[6] D. H. Park, M. Liu, C. Zhai, and H. Wang, "Leveraging user reviews to improve accuracy for mobile app retrieval," in *SIGIR '15*, 2015.
[7] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "Ar-miner: Mining informative reviews for developers from mobile app marketplace," in *ICSE 2014*, 2014.
[8] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *IEEE RE 2015*, 2015.
[9] S. Moghaddam, "Beyond sentiment analysis: Mining defects and improvements from customer feedback," in *ECIR 2015*, 2015.
[10] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *ICSE '06*, 2006.
[11] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in *ICSE 08*, 2008.
[12] S. Kim and E. J. Whitehead, Jr., "How long did it take to fix bugs?" in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, 2006.
[13] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: A text-based approach to classify change requests," in *CASCON '08*, 2008.
[14] M. Gegick, P. Rotella, and T. Xie, "Identifying security bug reports via text mining: An industrial case study," in *MSR 2010*, 2010.
[15] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Map estimation via agreement on trees: message-passing and linear programming," *IEEE Transactions on Information Theory*, 2005.
[16] T. Hofmann, "Probabilistic latent semantic indexing," in *SIGIR 99*, 1999.
[17] M.-W. Chang, V. Srikumar, D. Goldwasser, and D. Roth, "Structured output learning with indirect supervision," in *Proceedings of ICML*, 2010.
[18] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of machine learning research*, vol. 6, no. Sep, pp. 1453–1484, 2005.
[19] Z.-H. Zhou and M.-L. Zhang, *Ensembles of Multi-instance Learners*, 2003.