Preliminary Study on Applying Semi-Supervised Learning to App Store Analysis

Roger Deocadez
School of Technology
Oxford Brookes University
Oxford OX33 1HX, UK
roger.deocadez@brookes.ac.uk

Rachel Harrison
School of Technology
Oxford Brookes University
Oxford OX33 1HX, UK
rachel.harrison@brookes.ac.uk

Daniel Rodriguez
Dept of Comp Science
University of Alcala
Alcalá de Henares 28871, Spain
daniel.rodriguezg@uah.es

ABSTRACT

Semi-Supervised Learning (SSL) is a data mining technique which comes between supervised and unsupervised techniques, and is useful when a small number of instances in a dataset are labelled but a lot of unlabelled data is also available. This is the case with user reviews in application stores such as the Apple App Store or Google Play, where a vast amount of reviews are available but classifying them into categories such as *bug* related review or *feature request* is expensive or at least labor intensive. SSL techniques are well-suited to this problem as classifying reviews not only takes time and effort, but may also be unnecessary. In this work, we analyse SSL techniques to show their viability and their capabilities in a dataset of reviews collected from the App Store for both transductive (predicting existing instance labels during training) and inductive (predicting labels on unseen future data) performance.

CCS CONCEPTS

•**Information systems** → **Data mining; Web mining;** •**Human-centered computing** → **Mobile computing;** Smartphones; Mobile devices; •**Computing methodologies** → *Supervised learning;*

KEYWORDS

Semi-supervised Learning, Mobile apps, Apps reviews

ACM Reference format:
Roger Deocadez, Rachel Harrison, and Daniel Rodriguez. 2017. Preliminary Study on Applying Semi-Supervised Learning to App Store Analysis. In *Proceedings of EASE'17, Karlskrona, Sweden, June 15-16, 2017,* 4 pages.
DOI: http://dx.doi.org/10.1145/3084226.3084285

1 INTRODUCTION

Over the past few years, there has been some interest in focused theoretical and empirical studies of Semi-Supervised Learning (SSL) algorithms to help address the scarcity of labelled data [10, 16]. The limited amount of labelled data in many situations has frequently been highlighted. The effort needed to annotate data can be considerable, needing both effort and time [5], and sometimes requiring an expert [15]. Fortunately a large quantity of unlabelled data may be available at a relatively small cost [18]. This is the case with application stores where a large number of reviews is available but the classification into categories such as *bug related review* needs to be carried out manually. Semi-supervised learning includes Semi-Supervised Classification (SSC) and semi-supervised clustering. In this paper, we focus on SSC in the problem domain of mobile apps reviews classification.

Numerous studies have been conducted into identifying bugs or issues, features or enhancement requests for mobile app reviews using classical supervised techniques (e.g. [3, 11, 12]). Here we go further, with the aim of exploring semi-supervised learning with the following research questions:

- Are semi-supervised learning techniques a suitable approach for App Store analysis?
- How much data do we need?
- What are the problems faced when applying SSL?

To answer these research questions we conducted an experiment to determine the suitability of SSC techniques as well as the influence of the ratio of labelled data to unlabelled using the KEEL tool following the work by Triguero et al [16].

The remainder of the paper is structured as follows: Section 2 describes the experimental work carried out including the dataset, preprocessing, algorithms and evaluation measures used. Section 3 discusses the results. Section 4 discusses related work and Section 5 raises threats to validity. Finally, Section 6 concludes the paper and discusses future work.

2 EXPERIMENTAL WORK

2.1 Dataset

The dataset used was collected from the App Store during 2015. Apps fall into 10 categories (books, education, games, health, lifestyle, navigation, news, productivity, travel and utilities). Only the top apps (both paid and free apps) were included. Overall 40 apps were selected with a total of 932,388 reviews.

We randomly selected 2,757 reviews as our ground-truth set and manually categorised these reviews into three classes {*bug*, *request*, *other*} resulting in 543, 360 and 1,854 instances respectively. Although we collected all metadata available

EASE'17, Karlskrona, Sweden

DOI: http://dx.doi.org/10.1145/3084226.3084285

Table 1: WEKA `StringToWordVector` Filter parameters

Parameters	*Values*
Inverse Document Frequency (IDF) Transform	True
Term Frequency (TF) Transform	True
Lower case transformation	True
Minimum term frequency	5
Stemmer	Snowball stemmer
Number of words to keep	200

such version, number of stars and price, in this paper we only use actual reviews as plain text together with the label assigned.

2.2 Pre-proccessing

We used WEKA for the text mining pre-processing [19], i.e., to convert the strings with the reviews into vectors of words that classifiers can learn from. Table 1 shows the most important parameter specifications for the "`StringToWordVector`" filter applied in the Weka tool. We trimmed the attributes by removing numbers and other symbolic characters leaving a total of 139 attributes (words representing features).

For the rest of the experimental work, we used the KEEL framework [2]. The next step before applying the machine learning algorithms was to create a Stratified k fold partition for the training and evaluation steps. In particular, we used the *10-Fold Distribution Optimally Balanced Stratified Cross Validation* option provided by the KEEL data management tool.

2.3 Parameters and Classifiers

In this work we selected three well-known SSC algorithms: (i) Self-training [8, 20], (ii) Rasco [17] and (iii) Rel-Rasco [21].

Each of these SSC algorithms was in turn run with well known base algorithms, (i) kNN [1], (ii) C4.5 [14], (iii) naive Bayes (NB) and (iv) Support Vector Machines (SVM) with the Sequential Minimal Optimization (SMO) algorithm [13].

The default configuration parameters for all the methods used in the KEEL toolkit and in this study are based on recommended settings [16].

2.4 Evaluation Metrics

We evaluate the performance of our selected methods and algorithms using accuracy. Accuracy is the number of correct predictions divided by the total number of predictions. In future work, we intend to report on other metrics. Although we do not have extreme imbalance, we should also consider those metrics that are more robust to such problems.

3 RESULTS AND DISCUSSION

Our experimental dataset consists of 2,757 expert labelled mobile app reviews with 19.7% classified as *bugs*, 13.1% classified as *requests* and 67.2% as *others*. We evaluate the classification performances of three SSC methods (Self-Training, Rasco and Rel-Rasco) with four base classifiers (kNN, C4.5, SMO and NB) and four different training ratios (10%, 30%, 50% and 70%).

Table 2: Transductive Accuracy Results with Different Ratios of Labelled Data

Algorithm	10%	30%	50%	70%
Self-Training (kNN)	0.6584	0.7164	0.7332	0.7428
Self-Training (C4.5)	0.7391	0.7549	0.7764	0.7828
Self-Training (SMO)	0.7882	0.8332	0.8404	0.8473
Self-Training (NB)	0.7514	0.7788	0.7869	0.7813
Rasco (kNN)	0.5742	0.6689	0.6801	0.6871
Rasco (C4.5)	0.7198	0.7541	0.7669	0.7684
Rasco (SMO)	0.7788	0.8201	0.8386	0.8421
Rasco (NB)	0.7728	0.7924	0.7935	0.7846
Rel-Rasco (kNN)	0.5672	0.666	0.6976	0.6876
Rel-Rasco (C4.5)	0.7261	0.7572	0.771	0.7711
Rel-Rasco (SMO)	0.7596	0.8121	0.8419	0.8476
Rel-Rasco (NB)	0.7699	0.7855	0.7928	0.7861

Table 3: Inductive Accuracy results with Different Ratios of Labelled Data

Algorithm	10%	30%	50%	70%
Self-Training (kNN)	0.668	0.7258	0.7468	0.749
Self-Training (C4.5)	0.7356	0.7512	0.7817	0.7824
Self-Training (SMO)	0.7867	0.8328	0.839	0.8441
Self-Training (NB)	0.7516	0.7813	0.7861	0.7745
Rasco (kNN)	0.5621	0.6873	0.7225	0.7272
Rasco (C4.5)	0.724	0.7537	0.7741	0.7791
Rasco (SMO)	0.7809	0.8157	0.8332	0.8426
Rasco (NB)	0.7671	0.7911	0.7879	0.7774
Rel-Rasco (kNN)	0.5748	0.6833	0.7258	0.732
Rel-Rasco (C4.5)	0.7277	0.7458	0.7722	0.7759
Rel-Rasco (SMO)	0.7556	0.8139	0.8332	0.8462
Rel-Rasco (NB)	0.7719	0.7875	0.7886	0.7763

The accuracy of the SSC methods with increased ratios of labelled data is shown in Figures 1 and 2. It can be observed there is some increase in performance with the selected algorithms and base learners (kNN, C4.5 and SMO) as the labelled ratio increases from 10% to 30% but in general performance remains quite stable afterwards. Tables 2 and 3 also show the values for the transductive and inductive accuracy values respectively.

The classification performance of the methods Self-Training, Rasco and Rel-Rasco with the selected classifiers (kNN, C4.5, SMO and NB) in terms of labelled ratio factor have significant differences between them (although we have not compared them statistically here). However we can observe that we do not need much data to achieve results that are very similar to the ones obtained with classical supervised techniques. Our findings show there is a very slight increase in performance

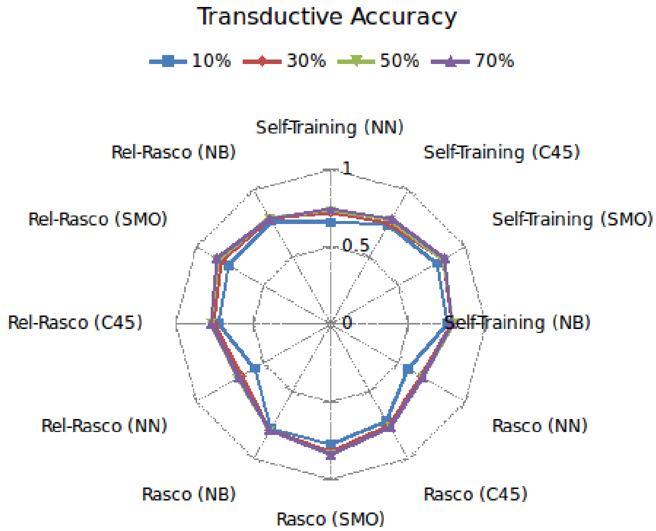

Figure 1: SSC Transductive Accuracy

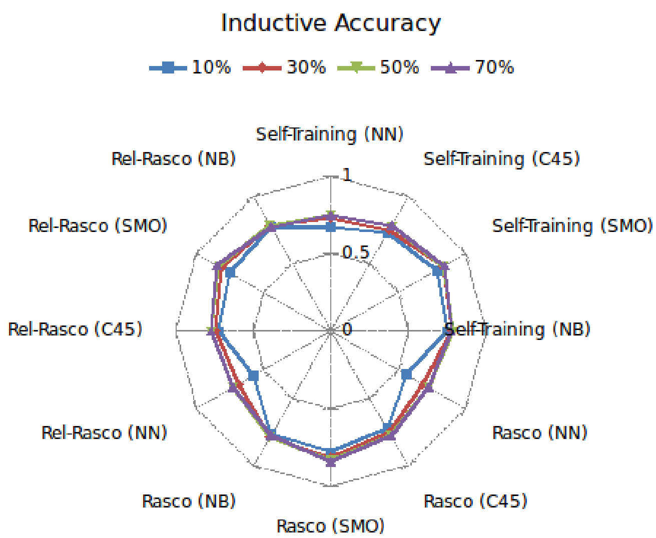

Figure 2: SSC Inductive Accuracy

for the 30% ratio compared to 10% but afterwards the results remain quite stable.

Although we need to do further experimental work, regarding our research questions we can observe that only a little labelled data is needed to achieve good results for accuracy. Furthermore, there are no large differences between transductive and inductive performances and as a result we can classify unseen new reviews with relatively high accuracy using a small amount of data.

4 RELATED WORK

Harman et al [6] suggested that mining for technical, customer and business aspects of the data held in app stores may help a variety of stakeholders. However, the paper does not use or suggest the use of machine learning techniques to automate the analysis.

As mentioned earlier filtering and summarizing reviews is an important part of app store analysis [9]. Maalej and Nabil use supervised machine learning to perform the classification automatically. In order to do this they created a truth set by labelling 4,400 reviews manually using content analysis. Clearly this is a large overhead which can largely be avoided by using semi-supervised learning, as our approach in this paper shows.

In earlier work we used linguistic rules to find feature requests in app store reviews [7]. We used LDA to perform topic modelling and categorise the requests, but the extraction of the feature requests was entirely manual and so would be difficult to scale.

Topic modeling is also used by Carreño and Winbladh in their work on finding requirements from users' comments [3]. The authors use sentiment analysis to categorise the requirements but the work differs from ours in that machine learning is not used to improve the efficiency of the requirements extraction.

Topic modeling is also used by Chen et al [4] in their work on finding the most useful reviews from a large and evolving set of reviews. The AR-Miner tool extracts the reviews, groups them, ranks them and then presents the results. The AR-tool does not use machine learning.

Pagano and Maalej performed review analysis with over a million reviews from the Apple App Store [11]. The techniques employed included manual content analysis and statistical analysis but machine learning techniques were not used. The paper acknowledges that reviews often contain many different topics.

Panichella et al [12] propose the use of review analysis to facilitate software evolution. Their techniques include using NLP, textual analysis and sentiment analysis to automatically classify app reviews. The authors use a number of different machine learning algorithms to help perform the classification of the reviews. The truth set was created manually following some pre-processing. The reported results of using sentiment and intention analysis (with the metrics precision, recall and F-measure) are all very encouraging. The authors used 20% of the truth set for training and the remaining 80% of the truth set as the test set.

5 THREATS TO VALIDITY

Here we consider three types of threats to validity: internal, external and construct.

When considering *internal validity* we must consider whether the treatment caused the outcome or whether it happened by chance or for some other reason. An obvious problem and threat to internal validity occurs because of the act of manually categorizing the truth set. However, we used a simple classification scheme (defect, feature or other) and one of the authors checked the classifications that were performed by a different author before we began our work.

When considering *external validity* we must consider whether the results can be generalised outside the scope of the study. We are relatively confident about the generalisability of our results because of the large number of reviews (932,388) and our relatively large truth set (2,757 reviews). We believe our results are fairly applicable to the Apple App Store because of this quite large random sample. However our results may not generalise to other app stores, particularly those with different quality assurance standards. In addition, we have only considered reviews that are written in English, and so our results cannot be generalised beyond this to other languages.

Turning to *construct validity* we must consider whether the variables used in the study accurately measure the concepts they should. We have used well established toolkits (WEKA and KEEL) and we are confident that the results they returned are correct, because we performed manual checks on a random sample of our results at the start of our work.

6 CONCLUSION AND FUTURE WORK

In this paper, we applied Semi-Supervised Classification (SSC) techniques to study their suitability with reviews from the App Store and our finding shows that SSC technique is benefiting the App Store analysis. Our results showed that although there are differences between the SSC techniques only a small amount of data is needed to achieve similar results to classical supervised techniques and the models learned can properly assign labels to the collected data and can also classify unseen future reviews.

Future work will pursue several paths. From the data mining point of view, we will further analyse other SSL algorithms and semi-supervised clustering techniques, analyse if the label ratio is dependant of the SSL technique, etc. From the mobile application domain point of view, we will make use of metadata not used in this study to better classify reviews to provide useful information to the developers.

ACKNOWLEDGEMENTS

The authors thank Oxford Brookes Universityand the University of Alcala, as well as projects amuSE (TIN2013-46928-C3-2-R) and BadgePeople (TIN2016-76956-C3-3-R).

REFERENCES

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. 1991. Instance-based learning algorithms. *Machine Learning* 6, 1 (1991), 37–66. DOI:http://dx.doi.org/10.1007/BF00153759

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J. C. Fernández, and F. Herrera. 2009. KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing* 13, 3 (2009), 307–318. DOI:http://dx.doi.org/10.1007/s00500-008-0323-y

[3] L.V.G. Carreño and K. Winbladh. 2013. Analysis of user comments: An approach for software requirements evolution. In *35th International Conference on Software Engineering (ICSE)*. 582–591. DOI:http://dx.doi.org/10.1109/ICSE.2013.6606604

[4] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 767–778. DOI:http://dx.doi.org/10.1145/2568225.2568263

[5] Nadia Felix F. da Silva, Luiz F. S. Coletta, and Eduardo R. Hruschka. 2016. A Survey and Comparative Study of Tweet Sentiment Analysis via Semi-Supervised Learning. *Comput. Surveys* 49, 1, Article 15 (June 2016), 26 pages. DOI:http://dx.doi.org/10.1145/2932708

[6] Mark Harman, Yue Jia, and Yuanyuan Zhang. 2012. App Store Mining and Analysis: MSR for App Stores. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. 108–111.

[7] C. Iacob and R. Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *10th Working Conference on Mining Software Repositories (MSR)*. 41–44. DOI:http://dx.doi.org/10.1109/MSR.2013.6624001

[8] Ming Li and Zhi-Hua Zhou. 2005. *SETRED: Self-training with Editing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 611–621. DOI:http://dx.doi.org/10.1007/11430919_71

[9] W. Maalej and H. Nabil. 2015. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *IEEE 23rd International Requirements Engineering Conference (RE)*. 116–125. DOI:http://dx.doi.org/10.1109/RE.2015.7320414

[10] J. Ortigosa-Hernández, I. Inza, and J. A. Lozano. 2016. Semisupervised Multiclass Classification Problems With Scarcity of Labeled Data: A Theoretical Study. *IEEE Transactions on Neural Networks and Learning Systems* 27, 12 (Dec 2016), 2602–2614. DOI:http://dx.doi.org/10.1109/TNNLS.2015.2498525

[11] D. Pagano and W. Maalej. 2013. User feedback in the appstore: An empirical study. In *21st IEEE International Requirements Engineering Conference (RE)*. 125–134. DOI:http://dx.doi.org/10.1109/RE.2013.6636712

[12] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 281–290. DOI:http://dx.doi.org/10.1109/ICSM.2015.7332474

[13] John C. Platt. 1999. Advances in Kernel Methods. MIT Press, Cambridge, MA, USA, Chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, 185–208. http://dl.acm.org/citation.cfm?id=299094.299105

[14] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[15] M. Sigdel, İ. Dinç, S. Dinç, M.S. Sigdel, M. L. Pusey, and R.S. Aygün. 2014. Evaluation of Semi-supervised Learning for Classification of Protein Crystallization Imagery. In *Proceedings of IEEE Southeastcon*. DOI:http://dx.doi.org/10.1109/SECON.2014.6950649

[16] Isaac Triguero, Salvador García, and Francisco Herrera. 2015. Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems* 42, 2 (2015), 245–284. DOI:http://dx.doi.org/10.1007/s10115-013-0706-y

[17] Jiao Wang, Si wei Luo, and Xian hua Zeng. 2008. A random subspace method for co-training. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 195–200. DOI:http://dx.doi.org/10.1109/IJCNN.2008.4633789

[18] Tiejian Wang, Zhiwu Zhang, Xiaoyuan Jing, and Yanli Liu. 2016. Non-negative sparse-based SemiBoost for software defect prediction. *Software Testing, Verification and Reliability* 26, 7 (2016), 498–515. DOI:http://dx.doi.org/10.1002/stvr.1610 stvr.1610.

[19] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. 2016. *Data Mining, Practical Machine Learning Tools and Techniques (4th Edition)*. Morgan Kaufmann.

[20] David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics (ACL '95)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 189–196. DOI:http://dx.doi.org/10.3115/981658.981684

[21] Yusuf Yaslan and Zehra Cataltepe. 2010. Co-training with relevant random subspaces. *Neurocomputing* 73, 10-12 (2010), 1652–1661. DOI:http://dx.doi.org/10.1016/j.neucom.2010.01.018 Subspace Learning / Selected papers from the European Symposium on Time Series Prediction.