Preliminary Study on Applying Semi-Supervised Learning in Mobile Application Stores

Roger Deocadez[1] Rachel Harrison[1] Daniel Rodriguez[2]

[1]Oxford Brookes University, UK
[2]University of Alcala, Spain

EASE 2017, Karlskrona, Sweden

Outline

Outline

Semi-Supervised Learning

Semi-Supervised Learning (SSL) lies between *supervised* and *unsupervised* techniques, where a small number of instances in a dataset are labeled but a lot of unlabeled data is also available.

- *Supervised* all data labelled
- Semi-supervised both labelled and unlabelled data
- *Unsupervised* no class attribute (all unlabelled)

This is the case in many situations:

- Natural language processing (Web mining, text mining), Part-of-Speech (POS), Labelling images
- Mobile Apps!

Inductive vs. transductive

- Dataset, $\mathcal{D} = \mathcal{L} \bigcup \mathcal{U}$
- Learner $f : \mathcal{X} \mapsto \mathcal{Y}$
- Labeled data $\mathcal{L}$ = $(X_i, Y_i) = \{(x_{1:l}, y_{1:l})\}$
- Unlabeled data $\mathcal{U}$ = $X_u = \{x_{l+1:n}\}$
  (avalilable during training, usually $l << n$)
- Test data $X_{test} = \{x_{n+1:...}\}$

Inductive vs. transductive

In SSL, threre are two distinct goals:

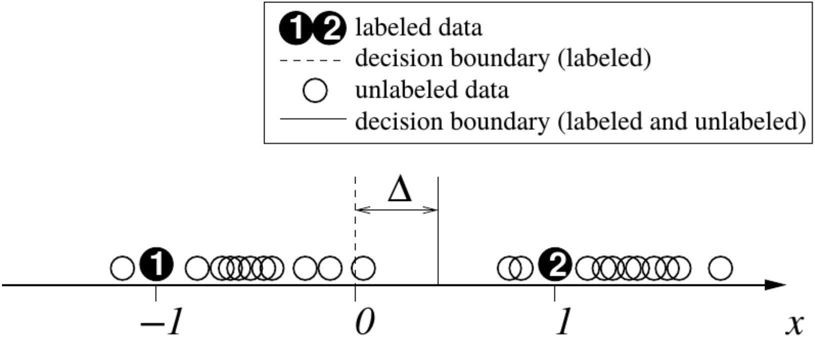
- Inductive. Predict the labels on future test data, i.e., learning models are applied to future test data (not avalilable during training).
  $\{(x_{1:l}, y_{1:l}), x_{l+1:n}, x_{n+1:\ldots}\}$
- Transductive. It is concerned with predicting the labels on the unlabeled instances provided with the training sample.
  $\{(x_{1:l}, y_{1:l}), x_{l+1:n}\}$

SSL Assumptions

Does SSL always work?

- **Smoothness assumption** (continuity), if two points x_1 and x_2 in a high-density region are close, then so should be the corresponding outputs y_1 and y_2
- **Cluster assumption**: If points are in the same cluster, they are likely to be of the same class.
- **Low density separation**: the decision boundary should lie in a low density region

SSL Assumptions



(Source: Zhu, ICML'07 tutorial)

Outline

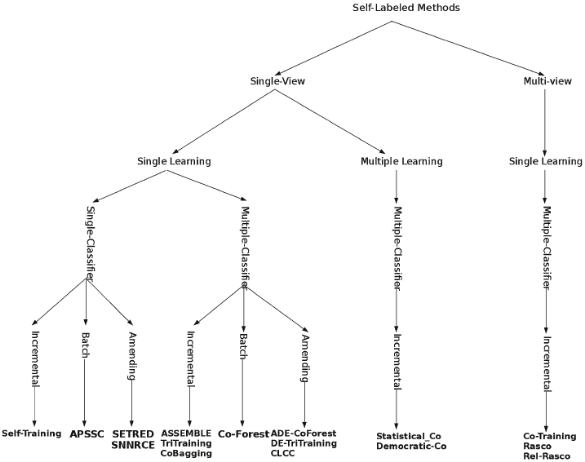
Semi-supervised Learning

Semi-supervised Learning taxonomy

- Semi-supervised classification
  - **Self-learning methods**/(Multi-view methods)
  - Generative models
  - S3VMs - Semi-supervised SVM
  - Graph-based methods
- Semi-supervised clustering
- Semi-supervised regression

In this work, we have tested self-learning approaches in mobile apps.

Self-labeling classification



(Source: Trigero et al. (2014))

Outline

1. Semi-supervised Learning
   - What is Semi-supervised Learning?
   - SSL Classification
2. Experimental Work
   - Problem
   - Dataset
   - Self-labeling Algorithms
   - Results
3. Conclusions and Future Work

Problem: To classify AppStore reviews with SSL

Aim: To analyse SSL techniques to show their viability and their *transductive* (predicting labels during training) and *inductive* (predicting labels on unseen future data) capabilities in a dataset of reviews collected from the AppStore.

Outline

1. Semi-supervised Learning
   - What is Semi-supervised Learning?
   - SSL Classification
2. Experimental Work
   - Problem
   - Dataset
   - Self-labeling Algorithms
   - Results
3. Conclusions and Future Work

Dataset

- Almost a million reviews downloaded from the Apple's AppStore.
- Out of the million reviews from 40 apps, we randomly selected around 3,000 that were manually categorized these into 3 classes *{bug, request, other}* as our ground-truth examples.
- In this work, only those 3,000 review were used and other parameters such as number of starts, category, etc. are available but we have only used the textual description in this work.

Dataset

Class	String
bug	'Waste of time. Loses all your entries.'
bug	'Never works.'
...	...
request	'Needs more email capabilities to make it as good as browser'
request	'It's a good game but needs improvements on resources'
...	...
other	'Excellent app ... Very easy to use'
other	'Love the reminders. :)'

A total of 2,757 reviews were classified into 3 classes *{bug, request, other}* for our ground-truth set resulting in 543, 360 and 1,854 respectively.

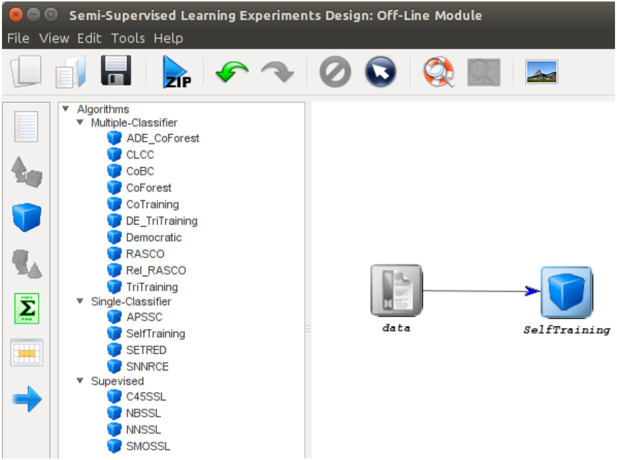
Preprocessing

The textual description was tranformed into a vector of words using WEKA's `StringToWordVector` Filter

Parameters	Values
IDF Transform	True
TFT Transform	True
lowerCaseTokens	True
minTermFreq	5
stemmer	SnowballStemmer
stopwordsHandler	MultiStopwords
wordsToKeep	200

KEEL

Used KEEL's SSL module



Outline

1. Semi-supervised Learning
   - What is Semi-supervised Learning?
   - SSL Classification
2. Experimental Work
   - Problem
   - Dataset
   - Self-labeling Algorithms
   - Results
3. Conclusions and Future Work

Self-labeling Algorithms

- **Self-training** follows a wrapper approach, using a base classifier, unlabeled data are labeled and added to the training set in an iterative way.
- **Co-training**, multi-view approach, each instance is represented by two sets of features (views), $\mathbf{x} = [\mathbf{x}^{(1)}; \mathbf{x}^{(2)}]$, than are trained independently and help each other.
- **RASCO** (RAndom Subspace Method for Co-training) is similar to co-training but with multiple classifiers trained on different attribute splits randomly generated.
- **Rel-Rasco** generates subspaces using the mutual information ranking metric between the features and class.

Self-training

Input: Labeled data $\mathcal{L}$, unlabeled data $\mathcal{U}$, and a supervised learning algorithm $\mathcal{A}$.

1. Learn a classifier f using labeled data $\mathcal{L}$ with f.
2. Label unlabeled data $\mathcal{U}$ with f.
3. Add new labeled data to $\mathcal{L}$ and removed them from $\mathcal{U}$

Repeat 1–3 until it converges or no more unlabeled example left.

Parameters of the SSL Algorithms

Table: Parameters of the Algorithms

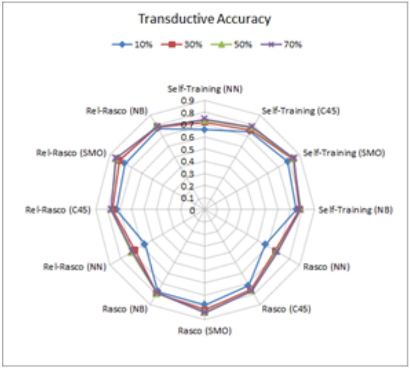
Methods	Parameters
Self-Training	MAX_ITER = 40
Rasco	MAX_ITER = 40, number of views/classifiers = 30
Rel-Rasco	MAX_ITER = 40, number of views/classifiers = 30

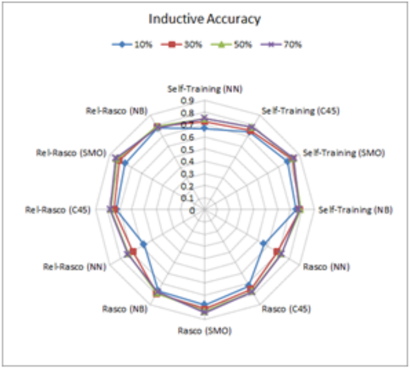
Base Algorithms & Parameters

Algorithm	*Parameters*
kNN	No. of neighbors = 3, Euclidean distance
C4.5	Confidence level $c = 0.25$, Minimum no. of items per leaf $i = 2$, Prune after the tree building
NB	No parameters specified
SMO	$C = 1.0$, Tolerance parameter = 0.001, $\epsilon = 1.0 \times 10^{-12}$, Kernel type = polynomial, Polynomial degree = 1, Fit logistic models = true

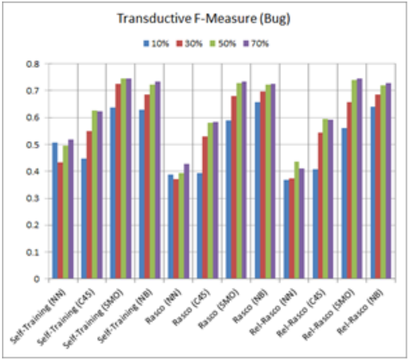
Outline

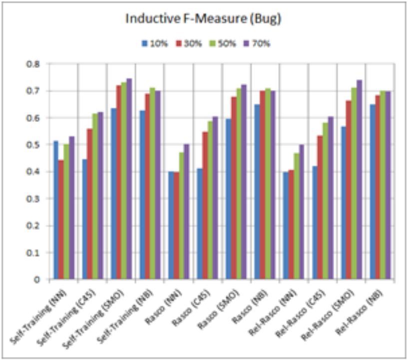
Results


Transductive Accuracy
10%
30%
50%
70%
Self-Training (NN)
Self-Training (C45)
Self-Training (SMO)
Self-Training (NB)
Rasco (NN)
Rasco (C45)
Rasco (SMO)
Rasco (NB)
Rel-Rasco (NN)
Rel-Rasco (C45)
Rel-Rasco (SMO)
Rel-Rasco (NB)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

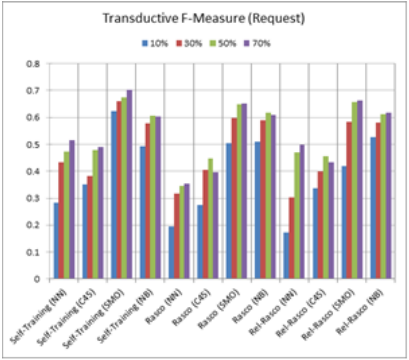

Inductive Accuracy
10%
30%
50%
70%
Self-Training (NN)
Self-Training (C45)
Self-Training (SMO)
Self-Training (NB)
Rasco (NN)
Rasco (C45)
Rasco (SMO)
Rasco (NB)
Rel-Rasco (NN)
Rel-Rasco (C45)
Rel-Rasco (SMO)
Rel-Rasco (NB)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

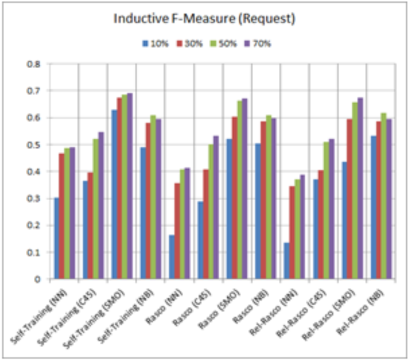
Results


Transductive F-Measure (Bug)
10% 30% 50% 70%
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Self-Training (NN)
Self-Training (C45)
Self-Training (SMO)
Self-Training (NB)
Rasco (NN)
Rasco (C45)
Rasco (SMO)
Rasco (NB)
Rel-Rasco (NN)
Rel-Rasco (C45)
Rel-Rasco (SMO)
Rel-Rasco (NB)


Inductive F-Measure (Bug)
10% 30% 50% 70%
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Self-Training (NN)
Self-Training (C45)
Self-Training (SMO)
Self-Training (NB)
Rasco (NN)
Rasco (C45)
Rasco (SMO)
Rasco (NB)
Rel-Rasco (NN)
Rel-Rasco (C45)
Rel-Rasco (SMO)
Rel-Rasco (NB)

Results


Transductive F-Measure (Request)
10% 30% 50% 70%
0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
Self-Training (NN)
Self-Training (C45)
Self-Training (SMO)
Self-Training (NB)
Rasco (NN)
Rasco (C45)
Rasco (SMO)
Rasco (NB)
Rel-Rasco (NN)
Rel-Rasco (C45)
Rel-Rasco (SMO)
Rel-Rasco (NB)


Inductive F-Measure (Request)
10% 30% 50% 70%
0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
Self-Training (NN)
Self-Training (C45)
Self-Training (SMO)
Self-Training (NB)
Rasco (NN)
Rasco (C45)
Rasco (SMO)
Rasco (NB)
Rel-Rasco (NN)
Rel-Rasco (C45)
Rel-Rasco (SMO)
Rel-Rasco (NB)

Conclusions and Future Work

Conclusions

- Not much data needed to achieve good results (similar to supervised classification)
- Large differences depending on the base classifier
- Apply further algorithms

Future Work

- Imbalance filters + other metrics
- Use the whole information
- Meta-learning - checking the characteristics of the data

Thank you for your attention!