Utilizing Crowd Intelligence for Online Detection of Emotional Distress

Master's Thesis Presentation

Siddhant Goel
Advisor: Han Xiao
Supervisor: Prof. Dr. Claudia Eckert

Chair for IT Security
Technische Universität München

March 12, 2013

Outline

Introduction

Backdrop

- Millions of people die every year because of suicide
- Most people are between 15 to 29 years old
- Rise of social media - Twitter, Facebook, Reddit, Wordpress
- Reddit - "/r/happy" [1] and "/r/suicidewatch" [2]
- People are not afraid of posting their inner feelings on the web

[1] http://www.reddit.com/r/happy
[2] http://www.reddit.com/r/suicidewatch

Backdrop

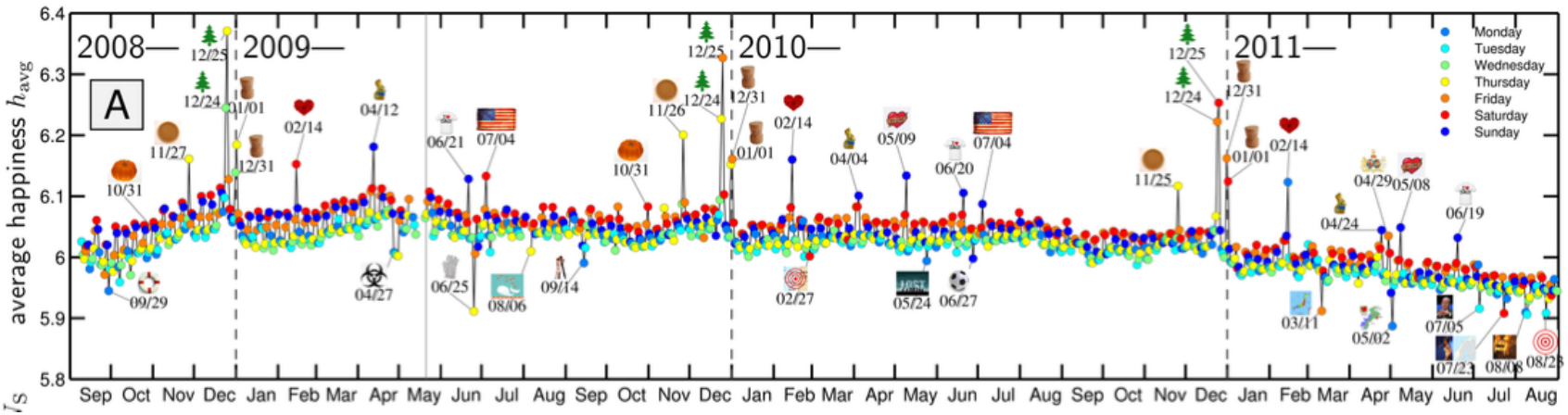


Figure : Happiness on Twitter as a function of time

- 46 billion words collected over 33 months
- Negativity on Twitter has been on the rise
- Words include *death*, *hate*, and even *suicide*

Motivation

Figure : Last tweet of Twitter user "@CapitalSTEEZ_"[3]

- Some accounts have lots of followers, some don't
- Lives can be saved if there is a surveillance system of suicide
- Public sentiment information available on the web + No analysis possible = Disconnect

[3] http://twitter.com/CapitalSteez_

Problem Definition

- Evaluate machine learning algorithms that can be used for identifying depressed emotions in pieces of text
- Build a web based system that can
  - tap into crowd intelligence to incrementally improve the classifiers
  - detect content on the web that indicates that its author is depressed or suicidal

Theoretical Background

Machine Learning

- Algorithms that can learn from data
- Construct a model from a given dataset, and then perform the required task on another dataset
- **Supervised learning** - Train the models on the training data, and predict on the test data
- **Unsupervised learning** - No distinction between training and test data

Text Classification

- Subset of machine learning algorithms (we focus on supervised text classification)
- Given some pieces of text, put future pieces of text into two or more categories
- Dataset $(\mathbf{x_n}, y_n)_{n=1}^{N}$ containing N instances
- Each instance $(\mathbf{x_n}, y_n)$ is of the form $[(x_{n,1}, x_{n,2}, ..., x_{n,D}), y_n]$
- Supervised learning - calculate y_n of test data given information about y_n from training data
- Unsupervised learning - calculate y_n given only information about $\mathbf{x_n}$

Support Vector Machines

- Fairly popular class of algorithms in binary classification
- Given training data in D dimensional space, find a decision boundary (hyperplane) that separates the two classes
- Maximize the distance of the boundary from any data point
- Decision function depends on a (usually small) subset of points called support vectors
- Distance function between two points is calculated using a kernel function

Linear kernel SVM

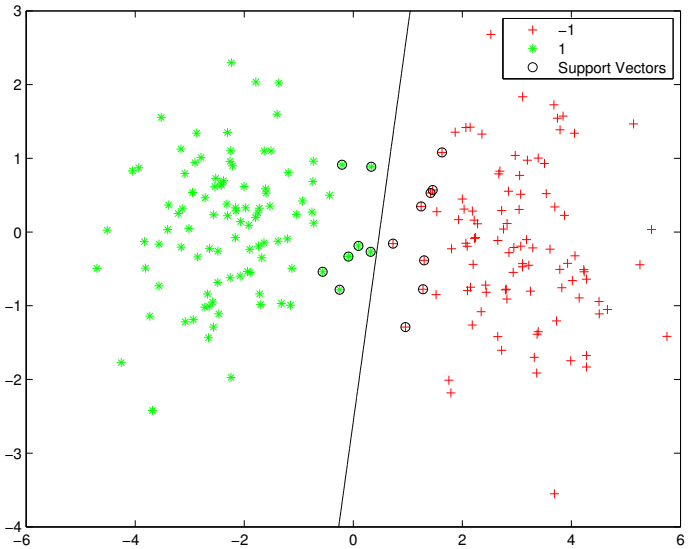


Figure : Classifying two subsets of a dataset using a linear kernel SVM

Kernel functions

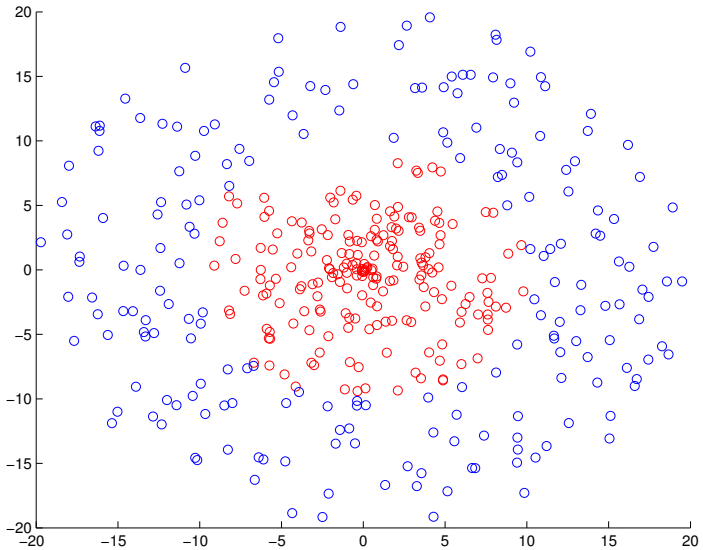


Figure : A dataset that cannot be classified using a linear kernel SVM

Kernel functions

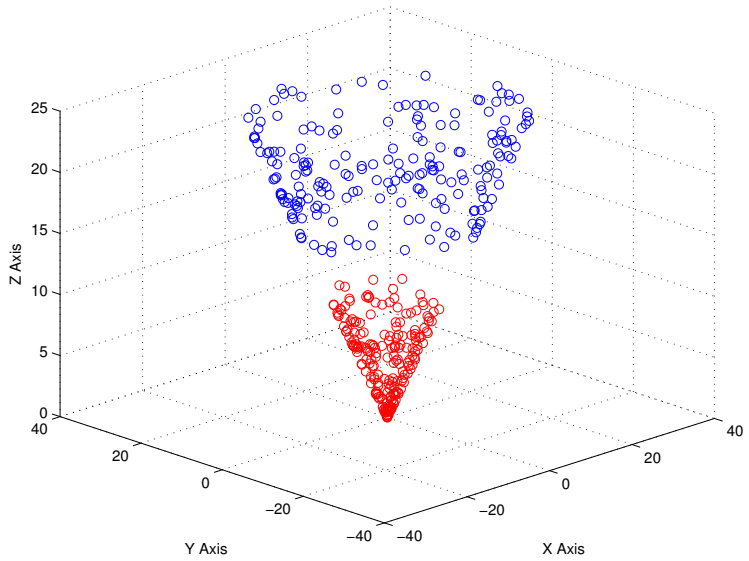


Figure : Add the third dimension as $\sqrt{x_1^2 + x_2^2}$ to transform the dataset into 3D

Ensemble Learning

- Class of machine learning methods that combine models to obtain better predictions
- Performance not guaranteed to be better than constituent classifiers
- Ensemble methods still usually outperform individual classifiers
- Soft requirement - underlying models should be diverse
- Various strategies to combine models - *select best*, *voting*, *boosting*, *stacking*

Bagging

- Combine M classifiers to form a single classifier
- To predict, obtain predictions from all constituent classifiers, and take majority vote
- Requirement - classifiers should change for even small changes in underlying classifiers
- Two main approaches for training individual classifiers
  - Sample split - different samples for different classifiers
  - Feature split - different features for different classifiers

Boosting

- Assign each sample a weight value (same for all samples in the beginning)
- Train M classifiers successively
- Each classifier focuses more on samples that the previous classifier classified incorrectly
- For each classifier, calculate ϵ (measure of error) and α (decreases with ϵ)
- Final prediction $= \text{sign}(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x_n}))$

Stacking

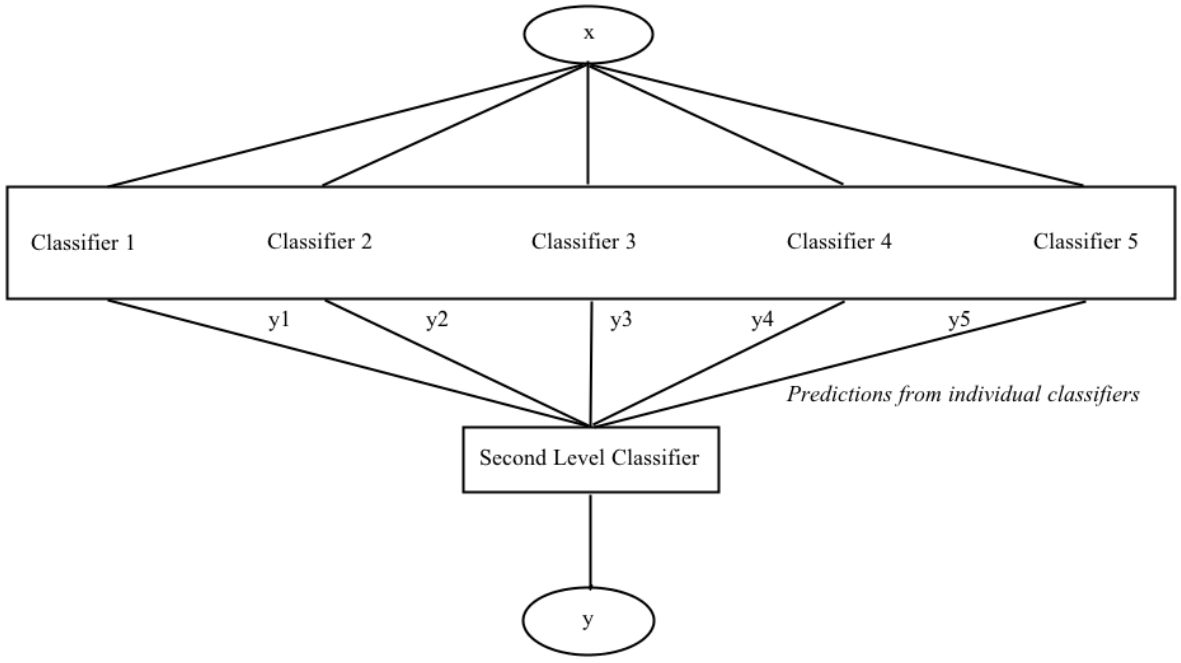


Figure : Prediction using a stacking ensemble

- Outputs from first layer form the input for second layer
- Layer-1 classifiers can be trained using bootstrapping or selecting random features

Experimental Results

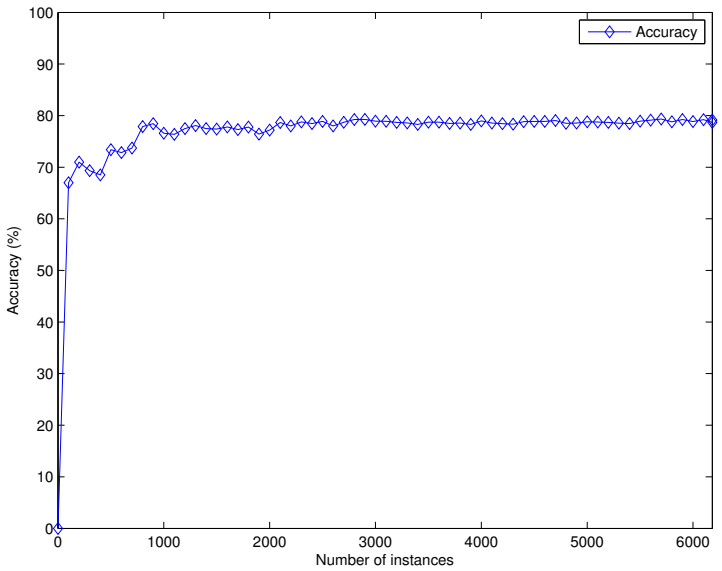
Evaluation of algorithms

Approach

- Implement all the models in MATLAB
- Extract n-grams (size upto 2) out of the Kaggle dataset
- Use tf-idf information as feature values
- Input matrix - 6182 rows and 23175 columns
- **SVM**, **Bagging**, **Boosting**, **Stacking** - obtain accuracy (10-fold cross validation) against number of samples
- **SVM** - Growth of number of support vectors with the number of samples
- **Bagging** - Accuracy against number of underlying models
- Start with 100 samples, and continue adding 100 samples on each iteration until no more samples are left

Evaluation of algorithms

Support Vector Machines



- Linear kernel accuracy - 79.02%
- Polynomial/RBF/Sigmoid kernel accuracy - 34.39%
- Having more features implies no need for a kernel function

Figure : Accuracy v/s Number of instances

Evaluation of algorithms

Support Vector Machines

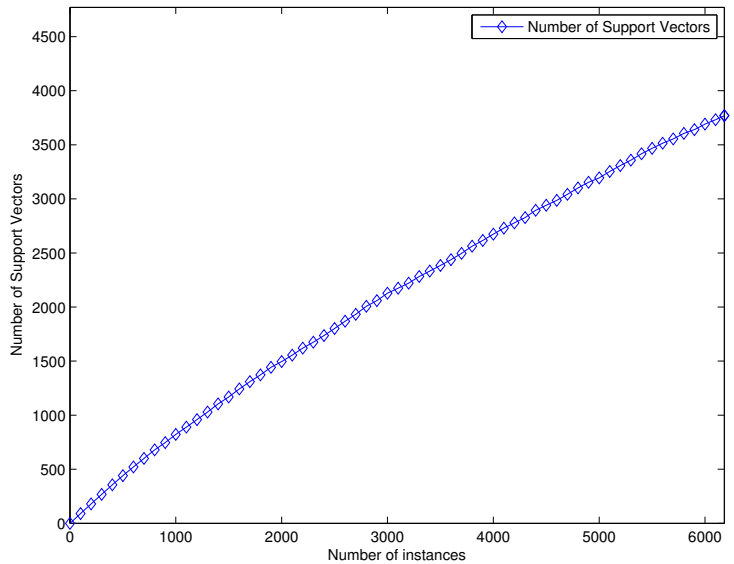
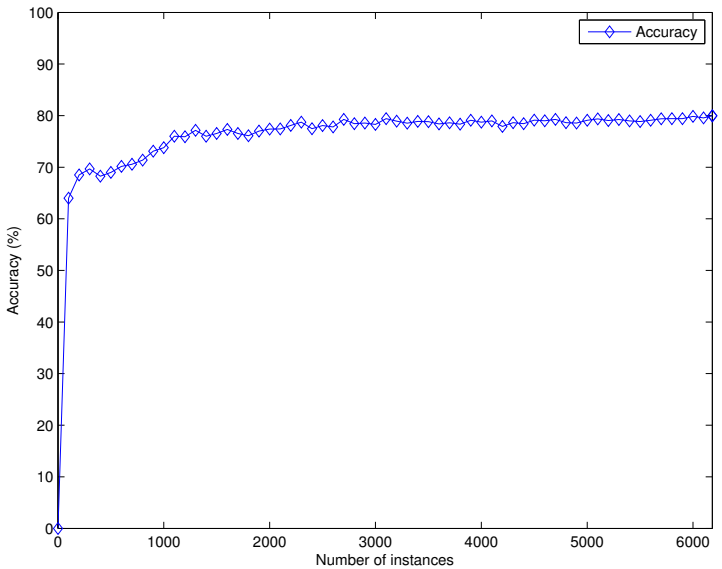


Figure : Number of support vectors v/s Number of instances

- 100 samples - 90% support vectors
- 6182 samples - 60% support vectors
- Less support vectors implies that the classification is easy

Evaluation of algorithms

Bagging



- 9 linear kernel SVMs underneath
- Performance ultimately governed by how SVMs perform
- Average accuracy on all 6182 samples = 79.65%

Figure : Accuracy v/s Number of instances

Evaluation of algorithms

Bagging

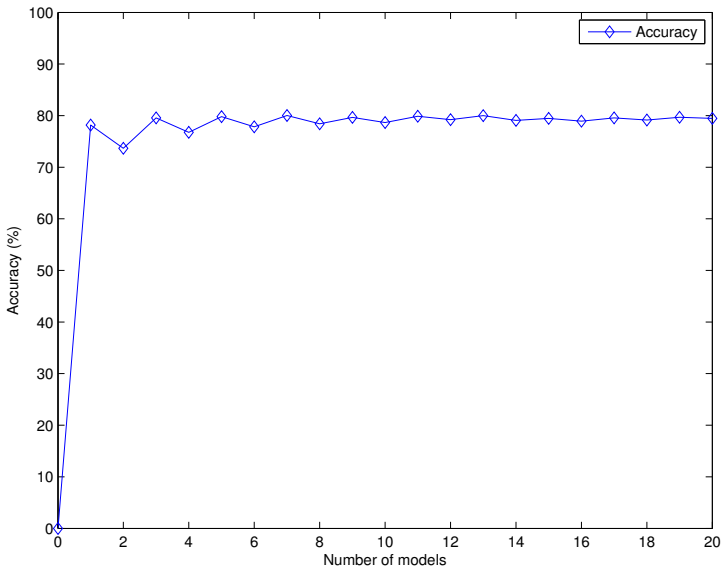
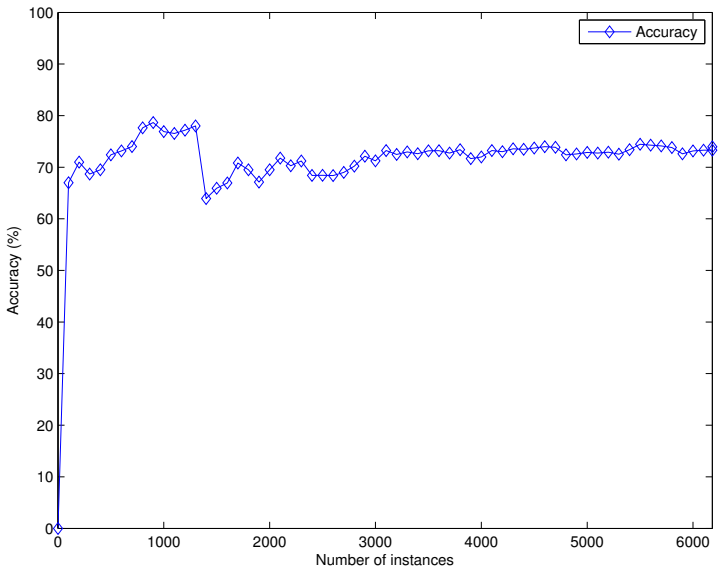


Figure : Accuracy v/s Number of models

- Accuracy "stabilizes" slowly with the number of models
- Each model is assigned a random subset of samples
- Increase in number of models *implies* Subsets overlap *implies* Performance stabilizes

Evaluation of algorithms

Boosting

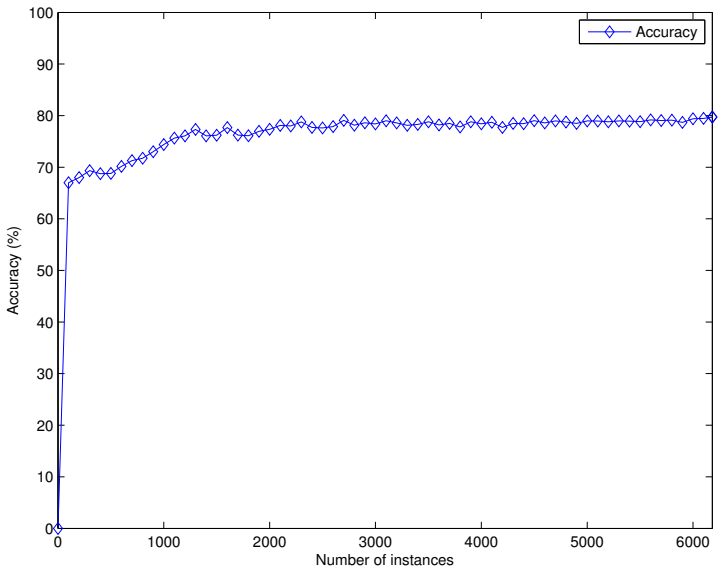


- 9 linear kernel SVMs trained successively on each iteration
- Average accuracy = 72.84%

Figure : Accuracy v/s Number of instances

Evaluation of algorithms

Stacking



- 9 linear kernel SVMs at first level
- Average accuracy = 79.48%

Figure : Accuracy v/s Number of instances

Web based system

Approach

- Implement all the models in Python, and web interface in Django
- Use posts from Reddit to build training data (crowd sourced), and tweets from Twitter as test data

Task	Frequency
Fetch 1000 posts from Reddit	24 hours
Fetch 100 tweets from Twitter	3 hours
Re-assign labels to previous tweets and update statistics	24 hours

Table : Scheduled tasks to download datasets