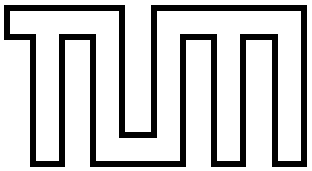


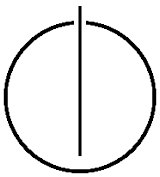
FAKULTÄT FÜR INFORMATIK

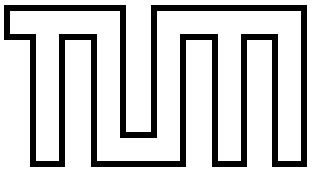
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

# Utilizing Crowd Intelligence for Online Detection of Emotional Distress

Siddhant Goel





FAKULTÄT FÜR INFORMATIK

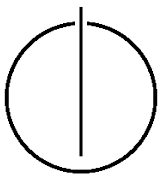
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

# Utilizing Crowd Intelligence for Online Detection of Emotional Distress

## Insert thesis title in German here


| | |
|---|---|
| Author: | Siddhant Goel |
| Supervisor: | Prof. Dr. Claudia Eckert |
| Advisor: | Han Xiao, M.Sc. |
| Date: | March 15, 2013 |

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I assure the single handed composition of this master's thesis only supported by declared resources.

München, den March 15, 2013 Siddhant Goel

# Acknowledgments

If someone contributed to the thesis... might be good to thank them here.

# Abstract

An abstracts abstracts the thesis!

# Contents

# Outline of the Thesis

## Part I: Introduction

CHAPTER 1: INTRODUCTION This chapter presents an overview of the thesis and it purpose. Furthermore, it will discuss the sense of life in a very general approach.

CHAPTER 2: RELATED WORK Related Work

## Part II: Theoretical Background

CHAPTER 1: CLASSIFICATION METHODS

CHAPTER 2: TEXT REPRESENTATION

CHAPTER 3: ENSEMBLE LEARNING

## Part III: Experiments

CHAPTER 1: EXPERIMENTS

CHAPTER 2: RESULTS

## Part IV: Conclusion

CHAPTER 1: CONCLUSION

# Part I.

# Introduction

Chapter 1

# Introduction

The Internet today is a vast source of information curated by real people. In the recent times, the popularity of websites like Twitter, Reddit, and Wordpress has shown that every day, more and more people are getting comfortable with expressing their inner feelings on the web, irrespective of whether they feel good or bad. Negative feelings expressed this way provide an important indicator of what might be going wrong in their lives, or what may ultimately lead up to something more tragic or even terminal. In some cases, factors leading up to suicide can be identified early on by looking at the physical and verbal behavior of the person under concern. The work presented in this thesis addresses this problem.

## 1.1. Machine learning and text classification

Machine learning is a branch of computer science that deals with building and analyzing systems that are able to learn from data. Algorithms based on such analysis involve constructing a model from a given dataset, and then using this model to perform required tasks. Machine learning techniques can broadly be divided into two categories - supervised learning, and unsupervised learning.

- Supervised Learning
  Methods falling in this category operate in two phases
  - In the first step, the availability of training data is assumed, which is used to build a model so that it takes into account the structure of the given dataset
  - In the second step, this model is used to make predictions on the testing data (the real world data). This is the data that the model has not seen yet, and is required to make predictions on.
- Unsupervised Learning
  Methods falling under this category operate in a single phase. It starts with a model with zero knowledge about the structure of the given dataset. As data is fed into the model, it continuously learns the structure of the given dataset and calculates the predictions based on this knowledge. The main difference between this family of

algorithms and supervised learning algorithms is the presence/absence of training labels.

The primary focus in this thesis remains on supervised learning methods.

Text classification is a subset of machine learning algorithms used to assign specific categories to pieces of data. More precisely, given some sample information (such as what kind of data points are to be assigned to which category), these algorithms allow for assigning categories to data points in the future. Under the scope of this thesis, the data is always assumed to be English-text. Since machine learning algorithms only deal with real numbers, a necessary first step is the conversion of natural language text into real numbers. The work presented in this thesis makes extensive use of supervised text classification algorithms.

## 1.2. Problem Definition

A large portion of the nearly one million people who die every year from suicide includes young people. In recent times, these people have started to express their inner feelings on the web, on websites like Twitter or Wordpress. Suicide is a medical condition that has the potential to be detected early on, by observing the physical and verbal behavior of the person under concern. The work presented in this thesis exploits these two facts, and aims to build a system that can monitor the public feed of Internet websites such as Twitter (on which people post about what they feel) and detect the content that may have been posted by a person facing emotional distress.

Pointing out the exact phrases which lead one to believe that a person may be under some degree of emotional distress is a problem best handled by psychologists. Even though prediction with machine learning algorithms is less sophisticated than human classification, it is known (and also presented in this thesis) that such techniques can identify and categorize the sentiment of a piece of text to a reasonable accuracy. A person may be under emotional distress if he/she posts content which is indicative of the following three major sentiments - suicide, depression, and loneliness. During the work performed in this thesis, the following categories of phrases were found indicative of whether or not a person needs help.

- Direct
  Phrases such as *"thoughts of suicide make me happy"*, *"I have a rope around my neck"* or *"my suicide note"* indicate in a very straightforward manner that the author is not only depressed, but is on the path of taking his/her own life. Such phrases are very direct indicators of the emotional condition of the person.

- Indirect
  Phrases such as *"I don't know anything anymore"* or *"Need someone to talk to"* or *"Please help"* indicate that the person who is posting such content does not necessarily want to kill themself yet, but they are not in a good emotional health either. Such words indicate that this person is under depression, or is feeling lonely. Even though they may not be suicidal yet, such a tragic event might be the next step in their lives.

Both the category of phrases indicate that the person under consideration is not in a very good emotional health, and may need help.

The final outcome of the work presented in this thesis is twofold -

- an evaluation of various text classification algorithms that could be a good fit for the problem at hand (identification of text which may or may not have been posted by a depressed person)
- a system that proactively looks for content on the Internet (on places like Twitter), and identifies people who have been posting emotionally distressed content; this system is then going to serve as a first step towards extending a helping hand to people who need it

The classification done in this work is distinguished by normal text classification by the sole factor of crowd intelligence, a detailed explanation of which can be found in Chapter 6.

Chapter 2

# Related Work

The work presented in this thesis falls under the combined domain of document classification and sentiment analysis, both of which have been very well researched. A summary of some of the main approaches to document categorization based on their content was presented in [24]. This work also discussed representing documents in such problems (approaches similar to building n-grams), building classifiers that can perform this categorization, as well as how to evaluate such classifiers. Some work that used n-grams to build a document classifier was presented in [4], wherein an accuracy as high as 99.8% was achieved (in one particular test involving classifying documents of length more than 300 and selecting the top 400 n-grams). However, documents of length less than 300 bytes and selecting the top 100 n-grams yielded an accuracy of 92.9%.

Much of the research done in the field of sentiment analysis has been summarized in [18], which also covers techniques that can be used to build systems that can enable information retrieval in opinionated data. The work done by [19] concludes that sentiment analysis is a much more challenging problem than simple text classification, and classifiers that perform well in text classification usually perform worse when it comes to sentiment analysis. For classification of movie review data from IMDB, they were able to obtain a cross validation accuracy of 82.7% using a linear-kernel SVM. Much of the research in this field focuses on full scale text documents, but not so much on microblogging platforms. Although one such work is [17], presenting a linguistic analysis (for the purposes of sentiment mining) of data collected from Twitter.

Support vector machines have been found to be highly successful in text classification problems mainly because of their ability to handle a large number of sparse features. This has been addressed theoretically in [14], including thorough explanations of document representation as well as efficient algorithms for training SVMs. An early analysis of the use of support vector machines in text categorization problems can be found in [13] and [6]. For the problem that [6] addressed (classifying emails as spam or not), SVMs provided the best performance when using binary features, and also took significantly less training time as compared to another model presenting comparable performance. The problem of categorizing the *Reuters* [5] dataset was also addressed in [16], wherein documents were represented using the tf-idf representation. From a practical perspective, [11] presents a software implentation of support vector machines as well as a guide to effectively using

them. Their implementation is (indirectly) used in the system presented in Section 7.2

A lot of the research in ensemble learning focuses on creating ensembles out of decision trees (summarized in [23]). The reasons behind the fact that a combination (ensemble) of models can outperform a single classifier are covered in [27]. Bagging, first proposed in [3], is well suited for addressing variance problems. Bagged ensembles of support vector machines for analyzing gene expression data are built and used in the work done by [28], where it is observed that such ensembles achieve better or equal, if not worse, classification accuracy with respect to a single SVM. As noted in [1], the performance of boosting is not uniformly better for *all* datasets. AdaBoost, short for Adaptive Boosting was first introduced in [8], and was found to be performing better than using individual classifiers by [2] when tested on the *Reuters* [5] corpus. Stacking [29] is another ensemble learning method that has been studied extensively. [7] conclude that a stacking ensemble performs equally as good as the best individual classifier in the ensemble.

Even though not a lot of work has been done in applying ensemble learning to perform classification on data extracted from microblogging websites, text classification on Twitter has seen a lot of activity in the recent times. Works including [25], [9], and [12] thoroughly investigate the problem of identifying the sentiment of a tweet based on its content. While [25] focuses on putting tweet in one out of five categories (news, opinions, deals, events, and private messages), [9] investigates the issues faced in assigning a binary sentiment (positive or negative) to a tweet in regards to a particular query term, achieving an accuracy of above 80% when including emoticons data. In spite of the fact that a tremendous amount of work has been done on identifying sentiments on Twitter, not a lot of visible work was found that addressed specifically the problem of finding out tweets displaying depressive emotions.

# Part II.

# Methodology

Chapter 3

# Classification Methods

Classification is one of the fundamental problems in machine learning. Given a dataset $\mathbf{X}$, it is required to separate the samples contained within the dataset into two (or more than two, depending on the input) classes. Formally, given a dataset that contains $N$ instances $(\mathbf{X}_n, \mathbf{Y}_n)|_{n=1}^{N}$, where each instance $(\mathbf{x}_n, \mathbf{y}_n)$ is of the form $[(\mathbf{x}_{n,1}, \mathbf{x}_{n,2}, ...\mathbf{x}_{n,D}), \mathbf{y}_n]$, (each $\mathbf{x}_{n,d}$ being the value of the feature $d \in [1, D]$, and $\mathbf{y}_n$ is the label of the sample which can take a limited number of possible values) the aim is to calculate the value of $\mathbf{y}_n$ given the feature information. This separation can usually be done using a supervised learning method, in which case the training data (on which the model is built) is given and it is required to predict the labels of the test data, or using unsupervised methods, where the model is required to identify the categories of the samples without any information on $\mathbf{y}$.

The performance of a particular classifier varies with the type of the data to be classified. Not all classifiers are good for all classes of problems. Some classifiers suit a particular problem more than some others; choosing a classifier for a problem still remains a decision which may or may not be completely scientific, even though there have been a number of tests been done to correlate classifier performance with data type [citation needed].

## 3.1. Support Vector Machines

Support Vector Machines (SVMs) form a fairly popular class of machine learning algorithms used mainly for binary classification and regression analysis. Given the training data, the goal of an SVM is to find a decision boundary (a hyperplane in a high or infinite dimensional space) that separates the two classes of data while maximizing the distance of the boundary from any data point. The resulting decision function is fully specified by a (usually small) subset of the data, and the points in this subset are referred to as support vectors.

All classifiers resort to a distance function, in some form or the other, that can provide a similarity measurement between two points. In the simplest form of an SVM, the distance function is simply the dot product between two points, and such SVMs are referred to as *Linear Support Vector Machines*. In the case that a simple linear SVM is not able to find a sufficiently accurate decision boundary that can separate the data points into two classes (simply because the input data is not linearly separable), the so-called *kernel trick* is used,

which involves transforming the data from a low dimensional space to a much higher dimensional feature space (in which the input data may be separable) using an appropriate function $\phi(\mathbf{x}) : \mathbf{X} \in \mathbb{R}^L \rightarrow \mathbb{R}^H (L \ll H)$, and then using the kernel function $\mathbf{K}(\phi(\mathbf{x}), \phi(\mathbf{y}))$ to actually perform the separation. The trick involved is that even though the transformation $\phi(x)$ may be expensive, computing the final similarity value $\mathbf{K}(\phi(\mathbf{x}), \phi(\mathbf{y}))$ is not.

The most popular kernel functions include

- Linear (the simple SVM) - $\mathbf{K}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})$
- Polynomial - $\mathbf{K}(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x} \cdot \mathbf{y} + \mathbf{c})^{\mathbf{d}}$
- Radial Basis - $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma |\mathbf{x} - \mathbf{y}|^2)$
- Sigmoid - $\mathbf{K}(\mathbf{x}, \mathbf{y}) = \tanh(\gamma \mathbf{x} \cdot \mathbf{y} + \mathbf{c})$

## 3.2. Multiple Kernel Learning

Multiple Kernel Learning makes an improvement to standalone support vector machines by using multiple kernels instead of a single one. In contrast to standard SVMs which use a single kernel function to calculate the similarity score between two data points, MKL algorithms combine scores from multiple kernels to obtain one single score, which is then used as the final similarity score. As summarized by [10], multiple kernel learning algorithms can be put into 12 major categories, based on some of their key properties (learning method, functional form, target function, training method, base learner, and the computational complexity).

The two main uses of MKL algorithms are also discussed by [10]. The first one uses the fact that since different kernels may be used as different notions of similarity, the accuracy obtained from each kernel may be maximized, and then the learning algorithm could be left to decide whether one kernel (which works better than the rest) or a combination of kernels is suitable for the task at hand. The second use is a more traditional use, where different kernels corresponding to different notions of similarity are combined, either in a linear or in a non-linear fashion to obtain a single kernel. As with all machine learning algorithms, multiple kernel learning does not suit all possible use cases, but in a lot of cases, combining multiple kernels may result in increased accuracy.

Chapter 4

# Text Representation

## 4.1. Introduction

Machine learning algorithms are designed to operate on real values. While classifying documents, a prerequisite is to convert the text in natural language to a format that such an algorithm can understand and work on. One such format is the vector space mode. Such processes include obtaining text from documents, building a vocabulary, text preprocessing (such as tokenization, stemming, stop words removal), and converting terms to actual usable feature values. All such issues are explored in this chapter, except the issues that arise due to usage of non-English languages (since this work focuses only on text in English).

## 4.2. Preprocessing

Preprocessing documents typically involves all the steps that are needed to be performed before proceeding on to extracting meaningful information from text. Text documents in their original forms normally contain a lot of redundant information that usually does not affect how a classifier behaves. Preprocessing a document removes all the noise, and brings a document in a state where it can be used for scoring. The following are the various filters that are applied.

- Tokenization
  Tokenization is the process of splitting a sequence of words into distinct pieces of alphanumeric characters, called tokens. Extra characters which are not needed, such as punctuations and white spaces are removed. For instance, tokenizing the text *"The quick, brown, fox jumps over the lazy dog;"* results in the following list of tokens -

  | the | quick | brown | fox | jumps | over | the | lazy | dog |
  |---|---|---|---|---|---|---|---|---|

  In most cases, tokens are split using whitespcaces, line breaks, or punctuation characters. Splitting on white spaces may also result in some loss of information. For

instance, if the string *San Francisco* appears in the text, then the tokens extracted will be *San* and *Francisco*, whereas the correct tokenization should treat both the terms in one token. Such problems are solved using n-grams, which are explained later in this section.

- Stop Words Removal
  Some of the words in the English language such as *and*, *the*, and *a*, besides some others appear in almost every text. These words add very little meaning to the text on the whole, and their frequency of occurrence is very high. Consequently, since these documents appear a lot in almost every document, it may lead to high, but inaccurate similarity scores between two documents; the best option is to remove such words from the text.

- Stemming
  Stemming is the process of reducing words to their root form, usually by stripping some characters from the word endings. A strict requirement for stemming is that related words must stem to the same final word. For instance, the words *car*, *cars*, *car's* and *car's*, must all stem to the same word *car*. This helps in removing unnecessary information from the text which would otherwise inflate the vocabulary with low-signal information.

## 4.3. Representation and Vector Space Classification

After a document has been preprocessed, it needs to be represented in terms of the useful content that it has, such that the relative importance of each word has been taken into account. Since a document is just a collection of tokens, a document $\mathbf{x_n}$ is represented as a vector $\mathbf{x}_n = (\mathbf{x}_{n,1}, \mathbf{x}_{n,2}, ..., \mathbf{x}_{n,D})$, where each dimension $\mathbf{x}_{n,d}$ corresponds to a token, and the value depends on its occurrence, either only in the document, or in the document as well as in the entire corpus. This approach is called the bag-of-words model.

For instance, consider two simple text documents,

| We are headquartered in Munich, Germany |
|---|
| We also have an office in Berlin, Germany |

Based on these two documents, the token dictionary is built like the following -

| "We": 1, "are": 2, "headquartered": 3, "in": 4, "Munich": 5, "Germany": 6, "also": 7, "have": 8, "an": 9, "office": 10, "Berlin": 11 |
|---|

and the documents are represented as the following vectors -

| 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0<br>1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1 |
|---|

where each value represents the number of times the corresponding token appeared in the document. The order of words does not matter; the features simply indicate whether or not the word appears in the document.

In this approach, the values represent term frequencies. A slight improvement is to divide the term weights with the total number of terms in the document, which assigns

the feature values while also taking into account the relative importance of the term in the document. But an even better improvement is to use the *tf-idf* value, which (for a term in the document) is defined as

$$\mathbf{tfidf}_{t,d} = \mathbf{tf}_{t,d} * \mathbf{idf}_t$$

where $\mathrm{tf}_{t,d}$ is the term frequency of the term $\mathbf{t}$ in document $\mathbf{d}$, and $\mathrm{idf}_t$ is the inverse document frequency of the term $\mathbf{t}$ across the entire collection of documents, usually defined as $\log(N/\mathrm{df_t})$, where $\mathrm{N}$ is the total number of documents, and $df_t$ is the number of times this term appears in the entire collection of documents. The main idea here is to reduce the term frequency weight of a term by a factor that increases proportional to its frequency in the corpus. This tf-idf value is the highest when the term occurs many times within a small number of documents, and the lowest when the term occurs in almost all documents. Hence it provides a good representation of a document in the vector space format. This representation is now used to calculate the similarity between two documents. The standard way for this is to compute the dot product between the two vector representations

$$\mathbf{sim}(\mathbf{x}, \mathbf{y}) = \frac{\vec{\mathbf{x}} \cdot \vec{\mathbf{y}}}{|\vec{\mathbf{x}}||\vec{\mathbf{y}}|}$$

Representing documents as vectors and calculating similarities based on their dot products leads to a view of a corpus as a *term-document* matrix, the rows for which represent the documents, and the columns for which represent the terms present in the corpus.

Chapter 5

# Ensemble Learning

Ensemble Learning is a class of supervised learning algorithms that use multiple models instead of a single one to obtain better predictions. A common problem with classifiers is that not all classifiers are suited for all kinds of problems. Ensembles combine multiple classifiers to construct a (hopefully) better classifier. The performance of an ensemble is not guaranteed to be better than the individual classifiers, but depending on the problem, an ensemble usually outperforms its constituent classifiers by a fair margin. Using ensemble learning requires performing more computation, but ensemble learning provides better results when the classifiers are not similar to each other. In this chapter, we explore the main ensemble learning techniques that we employed in our work.

## 5.1. Bagging

Bagging, also referred to as Bootstrap Aggregation is one of the most basic forms of ensemble methods. Given a training dataset $D$ of $n$ samples, each sample having $f$ different features, the aim of bagging is to combine a certain number of classifiers, each trained using a (mostly different) subset of the main data, such that every classifier learns about a different structure of the input data, and when the outputs from all such classifiers is combined, the final output is expected to be better than the individual predictions. Bagging is known to improve the performance when the classifier is slightly better than a random classifier. On the other hand, in cases where the constituent models are already strong, the performance is known to often degrade.

To train different classifiers on a different subset of the data, the training data is either split on the number of samples, or on the number of features. In the first case, we pick $n'(< n)$ samples (with replacement) and each time train a new classifier using the new dataset. Since the data points are picked with replacement, some of the data points may be common to some classifiers. In the second case, we train every new classifier using $m'(< m)$ features (with replacement). Whenever the ensemble is required to make a prediction on a new sample, a majority vote from all the constituent classifiers is taken, which becomes the final output of the ensemble. Bagging is known to reduce variance and avoid the overfitting problem.

## 5.2. Boosting

Boosting aims to build an ensemble by iteratively training weak classifiers on a distribution, and then finally putting them all together to build a strong classifier. In the training phase, the classifiers are trained one by one. Each training instance is assigned a numeric weight, which is the same for all samples in the beginning. After the first classifier has been trained, the weights for the points which were misclassified are increased, and the resulting input data (along with the modified weight values) is fed to the second classifier. This process continues until the last classifier has been trained. Each classifier has an incentive to focus more on classifying correctly those points which the previous classifier classified incorrectly.

At all times, each classifier's performance is maintained in a vector (usually named $\alpha$). Whenever a new point has to be classified, the individual predictions are obtained from the constituent classifiers, and a weighted sum (from the $\alpha$ values) is taken across all the classifiers to obtain the final prediction. One of the very popular Boosting techniques is AdaBoost. The main aim of Boosting is to build a strong classifier from a set of weak classifiers. To that effect, even classifiers only slightly better than random are considered useful, because in the final predictions, they will still contribute positively to the aggregate prediction by behaving like their inverses because of having negative coefficients in the final linear combination of classifiers.

## 5.3. Stacking

Stacking combines classifiers not by analyzing their performances on the training data, but by treating the individual predictions (on the training data) as a second level of training data, and then using this piece of data to build a final model that is ultimately used for making the final prediction. Different from the previous ensemble learning methods, stacking operates by feeding the initial training data to the set of base classifiers chosen. After these base classifiers have been well calibrated, the predictions on the training data are obtained from the base classifiers, forming a new dataset. This dataset is now fed into a second level classifier, which is responsible for making the final prediction. Often, the base classifiers (at the first level) are trained using different features of the input data to increase classifier diversity.

For making a prediction on a new point, predictions are made using all the base classifiers, forming an equivalent data point in which the number of features is the same as the number of base classifiers. This point is then run through the second level classifier to obtain the final prediction. In contrast to other ensemble learning methods, no voting or aggregation takes place in prediction. This class of methods has proven to be very successful, one recent example being that of the winning team in the Netflix prize competition.

# Part III.

# Experimental Results

Chapter 6

# Experiments

## 6.1. Dataset

This work focuses on detecting emotional distress amongst the general public from publicly available tweets/blogs. To predict the label for a particular tweet, machine learning models first need to be trained on similar data. This means that availability of a collection of tweets, each having a positive/negative label assigned to it was a prerequisite.

Non availability of such data in the beginning of this work led to the following approach - conduct benchmarks and evaluations on a similar problem, and apply the results to build the final system that can perform the said task. Experiments were initially performed on a dataset made available by a machine learning competition website [15], where the aim is to predict whether a certain piece of text (a comment from a conversation on the internet) can come across as insulting to a user or not. The dataset was a list of 6182 comments, each with a binary label.

Preprocessing this text consisted of - cleaning the text (allowing only alphanumeric characters and removing unnecessary punctuation marks), building n-grams of size 2, and using tf-idf information as feature values. Performing these steps resulted in an input matrix consisting of 6182 rows and 23175 columns. Each row corresponds to a distinct comment and each column corresponds to a distinct feature (n-gram, in this case).

After the evaluations, the next step in the work required the availability of an unlabelled dataset which could be labelled using crowd intelligence, and which could then finally serve as a basis for the final system to identify emotional distress. This data was fetched from the website Reddit [20], which is an online community for people to interact with one another. This website hosts a number of subwebsites called subreddits, two of which were of particular interest - *"/r/happy"* [21] and *"/r/suicidewatch"* [22]. The first subreddit is where people submit content if they are feeling happy and want to share their happy moments with others. The second subreddit is where people post when they feel depression, loneliness, feelings of helplessness, and feelings of ending their own lives. On this website, one piece of content is called a "story", having a title and a full text. Since the training data is supposed to resemble a tweet in length, only the shorter length "title" of the story was selected. Consolidation of dataset from these websites involved downloading and labelling a total of 2000 stories, 1000 each from *"/r/happy"* and *"/r/suicidewatch"*.

This builds a strong foundation for the system, after which stories are left to be fetched and labelled on demand by the users of the system.

Finally, the main data for identifying emotional distress is fetched from Twitter. Starting from January 17, Twitter's public streaming API [26] was used to fetch 100 tweets every 3 hours, for a total of 800 tweets everyday. Interruptions were faced in this process from February 5 until February 15. This gives an overall view of the general sentiment of the public, on which the analysis was performed .

## 6.2. Approach and Setup

The work done in this thesis was done in two phases. In the first phase, various machine learning techniques are evaluated, which include support vector machines, and ensemble learning methods. In the second phase, a system is built that monitors emotional distress on the internet.

In the evaluation phase, the comments dataset from Kaggle [15] was used. N-grams of length 2 were extracted and used as features in the vector space representation of comments. To assign values to these features, the tf-idf information as collected from across all comments was used. Once this representation was obtained, the dataset was then split into training and test data as per hold-out cross validation. A 70-30 split on the data was performed, i.e. to evaluate the performance of a particular model, it was trained over 70% of the comments, and was tested over the remaining 30%. This procedure was repeated for standalone support vector machines (using linear, polynomial, gaussian, and RBF kernels) and the ensemble learning methods which include bagging, boosting, and stacking.

In the second phase, a web based system is built that is able to

- identify emotional distress amongst the general public on Twitter
- allow anyone to contribute to the training data (crowd intelligence)

To incorporate crowd intelligence, the appropriate dataset (from Reddit) was downloaded and presented to the users of the system to label. This process results in consistent growth of the training data, which in turn improves the machine learning models continuously as and when they are retrained. These models are then used to make a more informed analysis about the level of distress on Twitter.

Chapter 7

# Results

The final output of this work is twofold

- an evaluation of support vector machines and ensemble learning methods in the domain of text classification
- a web based interface that implements the evaluated techniques and presents a system which can monitor the public feed from the Internet to find out people who are suffering emotional distress and may need help

## 7.1. Evaluation

The evaluation of all algorithms used is done on the comments dataset [15]. To report online learning scores, the learning continues in iterations. In each iteration, 100 random samples are picked on which the models are trained and the accuracy is measured. This is continued until no more samples are left. All accuracies reported henceforth are calculated using hold-out cross-validation, training the model on 70% of the data and testing on the remaining 30%.

### 7.1.1. SVM

The behavior of SVM based classification depends heavily on the kernel being used. Using a linear kernel yields a cross validation accuracy of 79.06%, while using either of polynomial/RBF/sigmoid kernels gives an accuracy of only 34.39%. This can be attributed to the intuitive notion of how kernel functions operate. A kernel function calculates the similarity between vectors when the vectors have been transformed from their original low dimensional space into a high dimensional space. In the current scenario, the text corpus contains 23175 features, which is already a very high number. Any transformation from this space into an even higher space results in a decrease in the final accuracy calculated, as can be noted in the results.

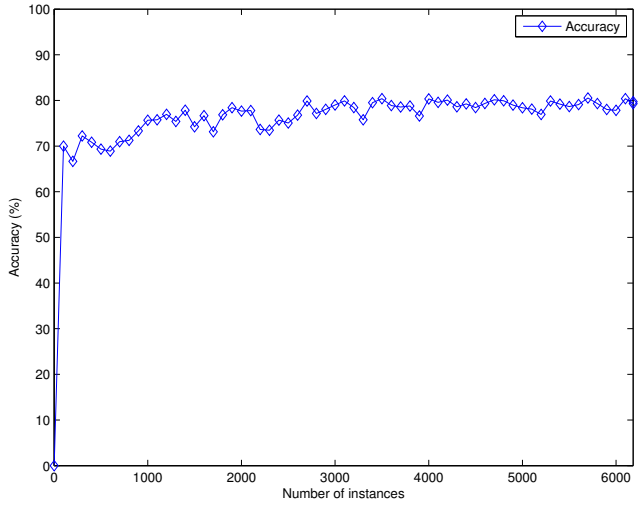



Figure 7.1.: Accuracy of a linear kernel measured against the number of instances

#### Accuracy

When using a linear kernel, the accuracy of an SVM shows an increase as the number of input instances increases. The score ranges from 70.00% when measuring on 100 samples, to 79.71% when measuring on all 6182 samples, as shown in Figure 7.1.

#### Number of support vectors

Support vectors is the subset of data points that the main classifier uses to establish the margin between the two class of points. Measuring the number of support vectors against the total number of instances gives an idea about the performance of the classifier - the more the number of support vectors (relative to the number of instances), the harder it is to perform a classification. Figure 7.2 shows this variation for the problem at hand. The number ranges from 70 for 100 samples to 3113 for 6182 samples, the increase being almost linear in the number of instances. It can be noted that the percentage decreases from 70% to around 50%, which shows that the more the information available to the classifier, the easier the task at hand becomes.

### 7.1.2. Ensemble Learning

Ensemble learning algorithms are nothing but a combination of individual classifiers. In this work, these individual classifiers happen to be support vector machines, which means that irrespective of which class of ensemble methods are used, the behavior is going to depend on the kernel being used. Since the maximum accuracy in the earlier section was found with the linear kernel, the same is used in the following experiments.

Plot number of instances against -

- accuracy

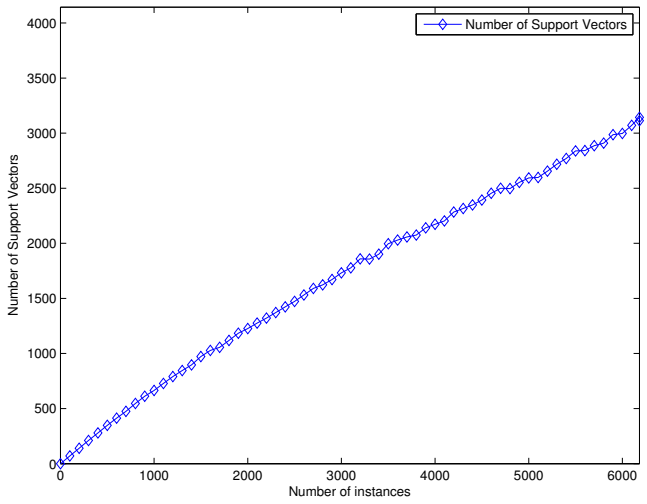



Figure 7.2.: Variation of the number of support vectors in a linear kernel SVM, measured against the number of instances

Plot accuracy against -

- number of models in the ensemble
- sample weights (MATLAB plots from Han's implementation)

### Bagging

non-online = 79.65% The increase in the accuracy observed by a bagging based ensemble is similar to the increase observed in simple support vector machines. This can be attributed to the fact that at heart, bagging is just a combination of SVM classifiers, with a majority vote deciding the label of the instance. This can be seen observed in Figure 7.3. A peak value of 79.97% is observed when the ensemble has to classify 6000 data points.

On the other hand, Figure 7.4 suggests that the accuracy of a bagging based ensemble does not vary a lot with the total number of models being used underneath. A maximum accuracy of 81.33% is observed for 7 classifiers, while the minimum value is 72.43% for 2 classifiers.

### Boosting

Boosting follows a different approach to ensemble learning, by combining classifiers in a way that each successive classifier gets more incentive to classify correctly the points which the previous classifier misclassified. As shown in Figure 7.5, the performance of a 9-classifier ensemble peaks at 80.38%, and does not vary a lot as the number of instances increases.

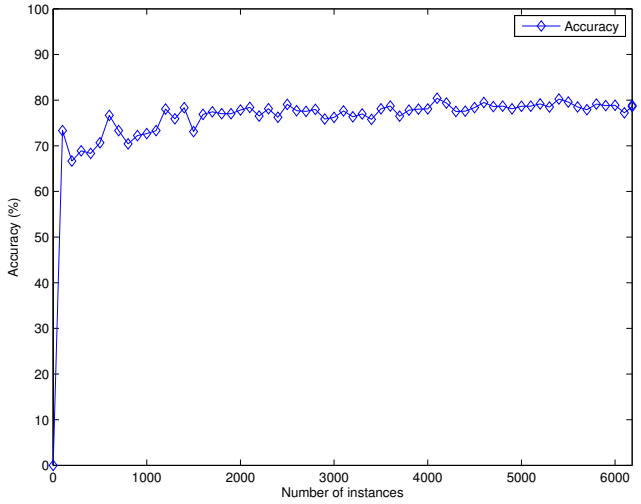



Figure 7.3.: Accuracy of linear kernel SVM based bagging ensemble as measured against the number of instances

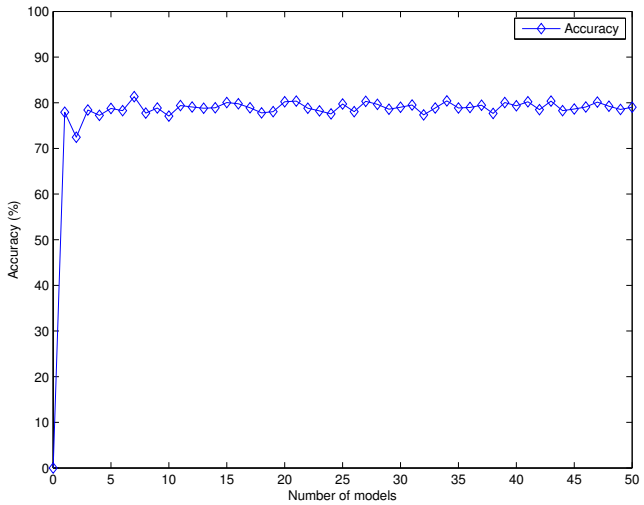



Figure 7.4.: Change in accuracy of a bagging based ensemble with the number of models

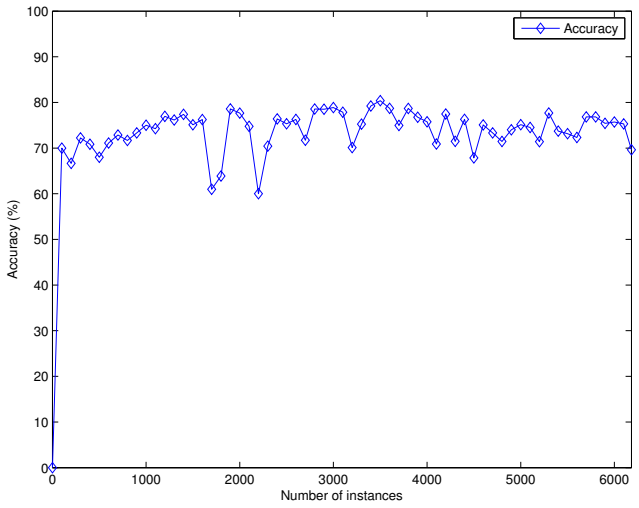



Figure 7.5.: Accuracy of linear kernel SVM based boosting ensemble as measured against the number of instances

#### Stacking

Stacking follows a more meta-level appraoch to classification, by training two levels of classifiers to perform the final classification. It can be seen in Figure 7.6 that the performance of stacked classifiers increases with the total number of instances, similar to the other two methods of ensemble learning.

## 7.2. System Details

The second major contribution of this thesis is a web based system that allows users to perform (broadly) the following functions - help build the training data, evaluate a level of distress amongst the general public, and find out particular people who have been posting content that appears to be distressed and qualifies as needing-attention.

Specifically, users can assign labels to stories fetched from reddit, which helps in building training data while tapping into crowd intelligence. The *monitoring* module then presents information about a general level of distress amongst people who are posting on Twitter (grouped by date), and about certain tweets that have been classified as depressed. Showing information about particular tweets which have been classified as depressed presents authors of the tweet which may need further attention in the form of psychological help.

The system is divided into two modules - *Ratings* and *Monitoring*. This choice is presented to a user as soon as they visit the front page of the web interface, as shown in Figure 7.7.

The flow of the entire system can be seen as being divided into two parts. The first part, which is more manual, includes users fetching stories (from reddit) and labelling them,

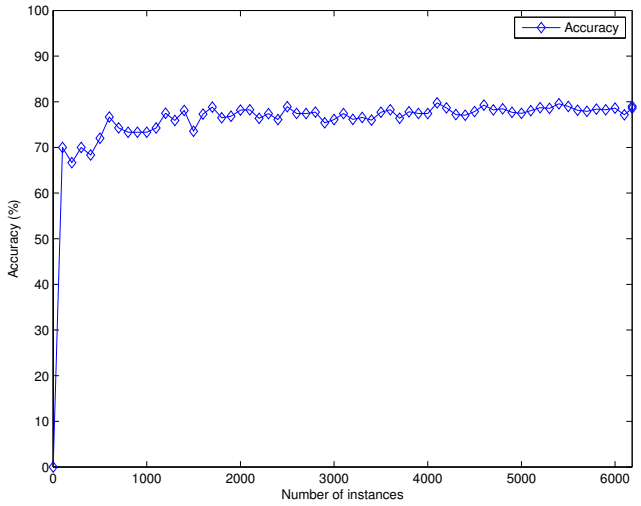



Figure 7.6.: Accuracy of linear kernel based stacking ensemble as measured against the number of instances

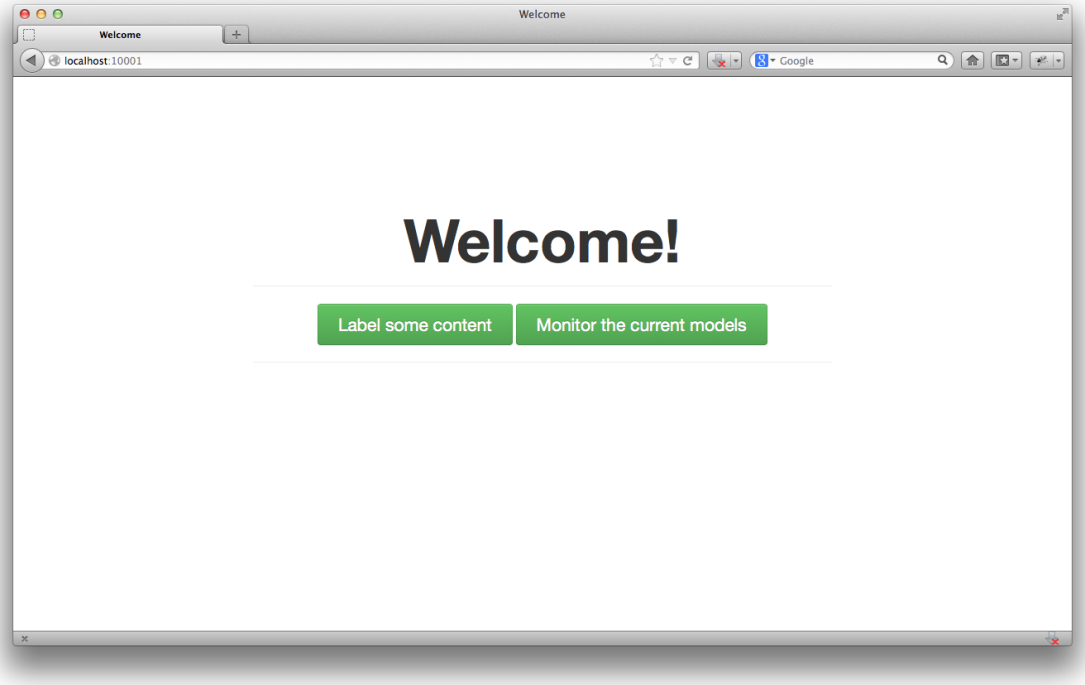



Figure 7.7.: Landing page

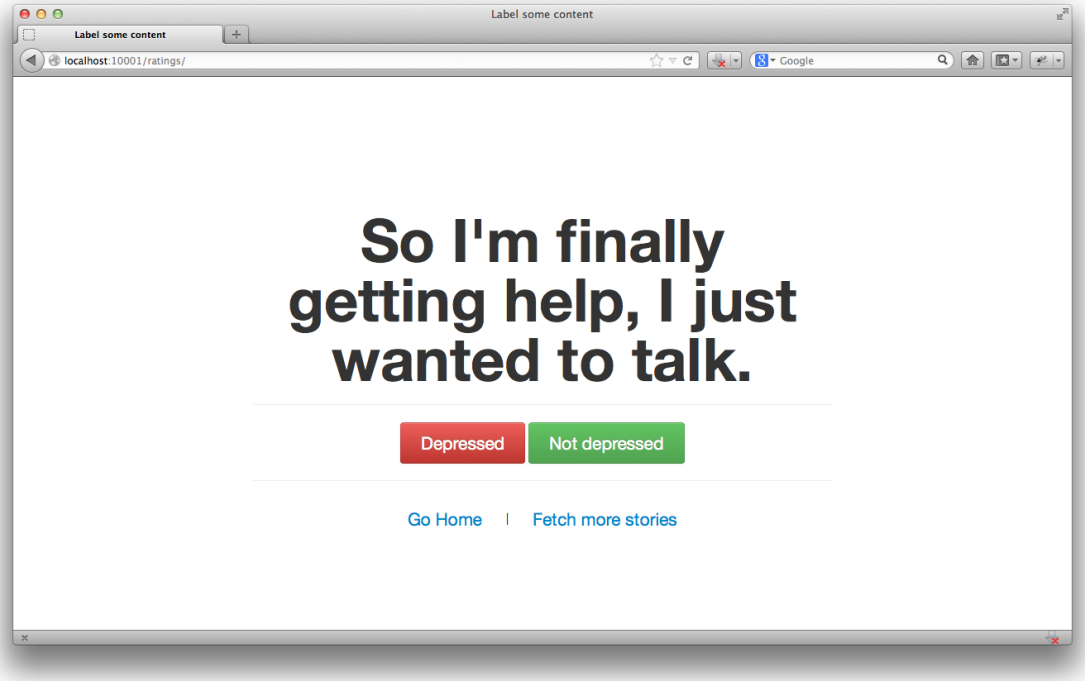



Figure 7.8.: Ratings module

which helps in building the training data. The second part is built using automatic cron jobs (pieces of code that are executed after specific intervals of time), and involves the following -

- every 3 hours - fetching 100 tweets from the public stream of Twitter
- every morning - assigning labels to all tweets fetched from Twitter according to the updated model

## 7.3. Ratings

The *Ratings* module is built to help consolidate the training data. Users are provided with options to label the unlabeled data (as "depressed" or "not depressed") that is already present in the database. If in case there are no unlabeled stories left, then there also exists an option to fetch more data from Reddit. When invoked, this option fetches 500 stories each from `/r/happy` and `/r/suicidewatch`.

As shown in Figure 7.8, the user is presented with a random piece of text that was fetched from Reddit, and two buttons for assigning this story a label.

## 7.4. Monitoring

# Part IV.

# Conclusion

Chapter 8

# Conclusion

Conclude

# Appendix

Appendix A

# Appendix

Appendix

# Bibliography


[1] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):105–139, 1999.

[2] Stephan Bloehdorn and Andreas Hotho. Text classification by boosting weak learners based on terms and concepts. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 331–334. IEEE, 2004.

[3] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[4] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[5] David D. Lewis. Reuters-21578, Distribution 1.0. `http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection`.

[6] Harris Drucker, Donghui Wu, and Vladimir N Vapnik. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054, 1999.

[7] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, 2004.

[8] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[9] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

[10] Mehmet Gönen and Ethem Alpaydin. Localized algorithms for multiple kernel learning. *Pattern Recognition*, 46(3):795–807, 2013.

[11] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[12] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 151–160, 2011.

[13] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine learning: ECML-98*, pages 137–142, 1998.

[14] Thorsten Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*, volume 186. Kluwer Academic Publishers Norwell, MA, USA:, 2002.

[15] Kaggle. Detecting Insults in Social Commentary. `http://www.kaggle.com/c/detecting-insults-in-social-commentary`.

[16] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154, 2002.

[17] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 2010, 2010.

[18] Bo Pang and Lillian Lee. *Opinion mining and sentiment analysis*. Now Pub, 2008.

[19] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[20] Reddit. reddit: the front page of the internet. `http://www.reddit.com`.

[21] Reddit. /r/happy. `http://www.reddit.com/r/happy`.

[22] Reddit. /r/suicidewatch. `http://www.reddit.com/r/suicidewatch`.

[23] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(3):660–674, 1991.

[24] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[25] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval*, pages 841–842. ACM, 2010.

[26] Twitter. Twitter Streaming API. `https://dev.twitter.com/docs/streaming-apis/streams/public`.

[27] Giorgio Valentini and Francesco Masulli. Ensembles of learning machines. *Neural Nets*, pages 3–20, 2002.

[28] Giorgio Valentini, Marco Muselli, and Francesca Ruffino. Bagged ensembles of support vector machines for gene expression data analysis. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1844–1849. IEEE, 2003.

[29] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.