Universität Ulm
The Faculty of Mathematics
and Economics

Determining Information Quality in Social Networks using Message Classification Techniques

Master Thesis

in Management and Economics

Leonid Edelmann

August 22, 2017

Supervision

Prof. Dr. Mathias Klier

Prof. Dr. Leo Brecht

Table of Contents

List of Figures

1 Introduction

1.1 Motivation

placeholder

1.2 Current State of Research

2 Basic Concepts

2.1 Machine Learning

A sub-branch of computer science that rose to prominence and started evolving during the 1950s as part of research in the field of artificial intelligence. Machine learning refers the development of algorithms, which allow computers to learn from presented examples. The computer is thereafter supposed to learn from its collected experience and automate the process of solving similar tasks. This process is referred to by term *training*.

One official definition as coined by Tom M. Mitchell [13] is "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

The most common use of machine learning algorithms is the analysis of real-world data for certain tasks, when a concrete programmer-written application would be ineffective in solving. Such is the case for example with problems, which a human would be able to solve, but would not be able to determine the rules for solving explicitly. Or alternatively, where the rules are not constant, but rather evolving as time progresses. The purpose of teaching a machine to solve such tasks, is modeling, prediction or detection of details or certainties about the real world.

Vivid examples of real world uses are speech recognition as used in cellphones or in call routing system as well as visual recognition, where and algorithm is trained to recognize graphic patterns and used in medicine or in hand written text recognition.

2.2 Text Mining

Text mining refers to the practice of extracting facts out of raw metadata, which comes in the form of text corpora or other unstructured data. Initially the data must be structured into a form compatible for its statistical analysis. This includes but not limited to, segmenting the text to more basic building blocks such as paragraphs or sentences. This practice is known as **stemming**. Cleaning up the data by removing non-informative words, which serve a grammatical role in human language,

but tend to be of no use for language processing done by machines. In the next phase the words are normalized to their **stem** by removing inflection which modifies the word's tense, case number and other grammatical properties. In the professional nomenclature, this is referred to as **stemming** or **lemmatizing**.

Methods used in text mining involve statistical pattern recognition,tagging-annotation, information extraction and frequency analysis. The end goal of text mining is namely, the production of qualitative information out of raw text, often automatically using machine learning.

Most of the usage of text mining methods in this work will be with the use of the NLTK module for the Python programming language. NLTK is a suite of libraries developed in the Department of Computer and Information Science at the University of Pennsylvania for Natural Language Processing [3].

2.3 Unstructured data

Data which is derived from the Internet, namely social networks is oftentimes unstructured. Unstructured data refers to information which is not organized according to a unison manner or in a data model, that fits the structure in which the data is to be utilized. This information tend to be heterogeneous in its data types and may contain texts, number, dates as well as media. Additionally, the data can vary highly in its integrity, recurrently having missing or only partial data.

In our case, the data comes in the form of Tweets originating from the Twitter servers. Tweets arrive packaged in the form JSON objects. These objects have no standard morphology, and many of the Tweets' fields are subject to variation. The most common approaches to dealing with such data are either restructuring it to a new data-model or conducting a textual analysis aimed at recognizing patterns in the information. Text Mining and Natural Language Processing are two prevailing mechanisms employed for this purpose. More on those in Chapter 3, *Collecting the Data*.

2.4 Classification

The correct assignment of (often unstructured) data to a category out of a set of predetermined categories, in the case of supervised learning or creation of new categories which best segment the data, in case of unsupervised learning. Supervised and Unsupervised Learning, are different approaches in Machine Learning, where the former requires continues input from the user whereas the latter is more autonomous.

Supervised Learning

Unsupervised Learning In the case of Unsupervised Learning, the task of classification is commonly similar to the task of *Clustering*. during the process of *Clustering*, the Machine does not require the input (training-) data to be anteriorly **labeled**, or classified to predetermined categories. Instead, the *Clustering* algorithm is usually given the number of categories (with some approaches, not even that). Next, the data is fractured according to what the Machine observes to be distinguishing features of the each category. Thus extracting and selecting features is of importance, since it further enhances the prediction power for future data. The features are often artificial constructs or rather functions transforming the original data from its original dimensions, in order to create distinguishing attributes of input data. Examples of such features vary from binary presence indicators of some word or character, interaction variables of two or more features to transformations, linear or otherwise. Two very common models are Hierarchical Cluster Analysis (HCA), K-Means and Mean Shift.

2.5 Information Quality

Social Networks incorporate the interactions of millions of individuals and organizations, thus allowing such an information flow to be classified as Big Data. Such data could be confined to the the widely acknowledged 5 characteristics of Big Data, also knowns as the 5 **V**'s[8]. Of interest here is the data's *Veracity*, referring to the trustworthiness and statistical reliability of said facts, originating from a plethora of sources and presenting little to no accountability for its correctness.

The questionable quality of such information makes basing critical business decisions on it, risky at best. One solution proposed to overcome these shortcomings would be to attach alongside the information quality metrics, which would describe its correctness, completeness and topicality as proposed by Klier and Heinrich(2016)[11]

3 Methodology

The approach I undertake in this study is similar to previous works in the field. Initially, data is gathered on a given theme. In the case of my research, the main query I wish to study would regard Tweets relating to the subject of E-Commerce platforms.

On of the motives of this research is to observe whether information originating from social media can be evaluated in real-time. With this consideration in mind, the data should bear as much resemblance to a live information torrent on Tweeter as possible. Acknowledging this consideration, the raw data in gathered in the form of Tweets originating from the Twitter databases. I collect the data synchronously, as it is intercepted by the servers. The *Streaming* Application Programming Interface (API)[18] is usually implemented in such use-cases. Using the *Streaming* API, a connection is established to the servers, which captures a narrow stream (about 15%) of all Tweets relevant to a given search term.

The captured data, must then be restructured to fit the data-model of this study, namely appropriate input for Machine Learning algorithms. This includes a preliminary analysis of the data and removal of incomplete observations. The data structure in which Tweets are stored allow for a dynamic non-standard structure, which in turn means they vary in properties. Therefore some Tweets must be adapted to conform to a unitary pattern. It is also quite common for Tweets to retrospectively be removed, either by themselves or by Twitter moderators. Such Tweets cannot be analyzed, since they are no longer available on-line. The process of gathering data and cleansing it is discussed in Part [4.1].

The next stage entails representing the data in a form which is suitable for Machine Learning Algorithms. For this purpose, I build *features*[1] for each observation. *Features* are unique properties of the data, which are used to describe it. Different types of *Features* are used in accordance with the Learning approach undertaken. These approaches we discuss in Part [4.2].

These features are then analyzed for consistency and correctness. Subsequently, the features are passed into different Machine Learning algo-

[1]Machine Learning nomenclature

rithms, in order to *train*[1] them. *Training* is the process of deducing the decision rules for classifying the data into one of the categories. This deduction is based on the information the algorithm draws from the input data. Afterwards, the empirical success of these different algorithms will be statistically measured the summarized. Additionally, implications are to be drawn about other use-cases, which are out of the scope of this research.

3.1 Training Classifier

The purpose of *training* is to create Classifiers, which automatically recognize and *label* new observations. The intrinsic decision algorithm, through which the Classifier will decide how to classify a novel observation, usually remain hidden and operate as a black-box of sorts. Since the decision rules could be numerous and far from intuitive for human readers. This is especially the case, when said algorithms are convolutional. Convolutional machine learning schemes, such as Deep Neural Networks, may incorporate numerous stages of parameter construction which renders them practically incomprehensible to human users. The actual implementation of all the algorithms would be programmed in the Python programming language and will be primarily making use of the scikit-learn[14] module.

The data used for the purpose of this study consolidates 12.520 unique Tweets. For the purpose of reaching robust results, the *Hold-out Method* is used as follows. For each instance of *Algorithm training* the entire data corpus is split into two parts - a training set and testing set. The training set would usually be allocated the larger portion of the data (about 70%) and would be used, as the name suggests for training the Classifiers. The rest of the data (about 30%) would be used for testing the Classifiers accuracy. During each such training-testing session, the data corpus is shuffled. This in turn means, that the training and testing sets constantly differ. This process of splitting, training and testing using different data in each iteration should produce statistically significant results. This method of constantly splitting the data randomly ensures the results robustness.

The following paragraphs expand on the different Machine Learning schemes. Figure 3 illustrates these schemes according to their types.

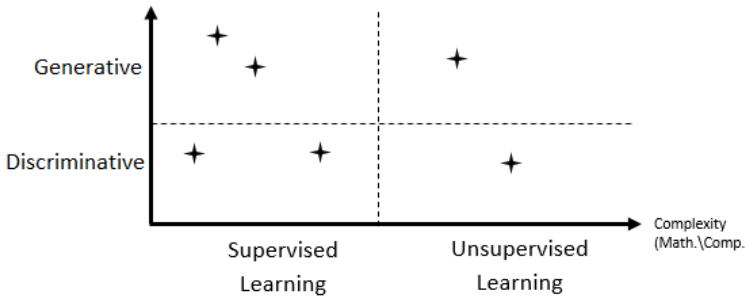


Figure 1: Clustering of Different Classifiers

3.2 Supervised Learning

3.2.1 Regressions

One approach and probably the most basic one is simply running a regression [1] with all the features as the independent variables and the a numeric representation of the classes as the dependent variable. Some threshold value for the classes must be determined since the estimate for the explained variable, will have a non-deterministic value.

$$\mathbb{E}(\hat{y} \mid \mathbf{x}) \approx \begin{cases} label = \text{News}, & \hat{y} \geq y^* \\ label = \text{Not News}, & \hat{y} < y^* \end{cases} \tag{1}$$

Linear Regression This method builds a linear dependency system between the explained variable (the *Class* in this case) and the explaining variables (*features*). With the *Bag-of-Words* approach, each variable x is a dummy variable indicating whether a given word m is present in Tweet i. The estimations parameters, which are denoted with β_j are derived using Ordinary Least Squares. In itself, the linear regression estimation is a weak predictor for the purpose of classification, but it allows for calculating other useful statistics such as the coefficient of determination, commonly known as R^2. This static demonstrates the part of the variance that is explained using the provided variables and is also sometimes called 'goodness of fit'. We therefore strive to maximize it as much as possible. The more of the variance that is covered by our variables, the better can we predict the out come of the classification. Another point of interest is the distribution of predicted classifications ($\hat{y}$). In an ideal scenario, the distribution of $\hat{y}$ would be concentrated

around the numerical representations of the two classed, as follows:

The basic premise for the use of linear regressions as classification tools, where n is the number of observations, m is the number of features [2].

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_m x_{mi} + \epsilon_i \qquad \forall\ i \in [1, n]. \tag{2}$$

Logistic Regression Despite its name, the actual regression model executed here is linear, and is used primarily for classification rather than regression analysis. This regression scheme differs from linear firstly, in the fact, that the possible outcomes of the dependent variable are discreet rather than continuous. Secondly, the probabilities which describe the different outcomes of a regression instance, are modeled using a logistic function. The discrete outcomes can in turn be converted to labels, which in allows for a much more fluent application as a classification tool and is therefore commonly employed in Machine Learning. Logistic regression analysis is formally represented as follows in [3]. The x vector represents all explanatory variables, in our case, the Tweet features. The term error ϵ follows the standard logistic distribution, which also gives the regression its name.

$$y = \begin{cases} 1 & \beta_0 + \beta_1 x + \epsilon > 0 \\ 0 & \text{else} \end{cases} \qquad \forall\ x = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} \tag{3}$$

3.2.2 Support Vector Machines

Background The original Support Vector Machine algorithm was formulated by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 while working at the institute of Control Sciences in Moscow. Despite the theory behind SVMs being far from new today, it actually remained mostly theoretical until quite recently. The state of technology at the time of its conception was far behind the theory, and it would take decades until computers would reach the computational abilities that could facilitate SVMs.

Implementation SVM aims at segmenting an input dataset to binary categories [6] . For example, positive and negative observations or analogously membership to a certain category or the lack thereof. A common example is whether a given Email should be classified as Spam or not. The rational behind SVM could best be visualized by imposing the training observation on two dimensional Euclidean Space. The algorithm strives to define a dividing line, which would create a separating threshold between the two groups. In two-dimensional space, the separator is demarcated using a simple line. Said line is then positioned in a manner, creating maximum separation between the two groups. This is done by drawing parallels through the the closest positioned members of both groups. The linear divider lays in the middle between the two parallels. The observations which lay on the parallels themselves, define the *curbs* of the so-called *dividing street*, and are called *support vectors*. From them in turn, the algorithm derives its name. Any new observation presented to the trained classifier for classification, will be associated to the according group, as is defined by its position relative to the separating hyperplane.

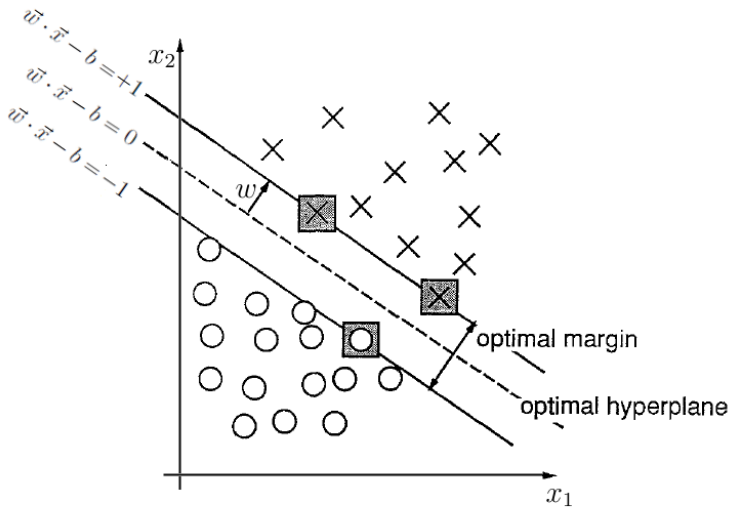


Figure 2: Euclidean Imposition of data. The support vectors, marked with grey squares, define the margin of largest separation between the two classes.

[7]

A dataset of size n as in [4] is to be used as a training set, in which y_i stands for the actual class of the observation i and can have one of either values $y_i \in [+1; -1]$. The values represent a positive and negative clas-

sification accordingly. A positive (+1) classification in this case implies the sought-after class, *News* in our case. The vector $(\vec{x}_i)$ is an array of size p, representing the features which compose each observation. In the 2-dimensional example, the 2 features of $(\vec{x}_i)$ could be graphically represent in euclidean space as the (x, y) coordinates $\vec{x}_i = [x_i, y_i]$.

$$\begin{gathered}(\mathbf{x}_1, y_1),..., (\vec{x}_n, y_n), \quad y \in \big[-1; +1\big] \\ \mathbf{x} = (x_{i1}, x_{i2}, ..., x_{ip})\end{gathered} \tag{4}$$

The optimal hyperplane segregating the two data classes, creating a maximally wide barrier between the closest observations of the opposing classes is presented in [5]. The vector $\vec{w}$ is the normal vector [2] to the dividing hyperplane and the parameter b represents the normal vector's offset along the axis. The area between the 2 hyperplanes, which pass through the closest observations is referred to as the *margin* and is defined by the supporting vectors. These support vectors are illustrated in [6].

$$\begin{aligned}&\vec{w} = (w_1, ..., w_p) \\ &\vec{w} \cdot \mathbf{x} - b = 0\end{aligned} \tag{5}$$

$$\vec{w} \cdot \mathbf{x} - b = \begin{cases} +1 \\ -1 \end{cases} \tag{6}$$

The distance between any point and the separating line [5] is shown in equation [7]. The distance between the 2 hyperplanes is equal to twice the distance between the support vectors and the separating line, hence $\frac{2}{||\vec{w}||}$. The optimization problem at hand is thus the maximization of this distance, in order to create a maximally distinct margin between the classes. Furthermore, no observation from the training data can be positioned inside the margin [8].

$$\begin{gathered}\text{Dist. for point i :} \\ \frac{|\mathbf{x}_i \cdot \vec{w} + b|}{||\vec{w}||},\end{gathered} \quad \Rightarrow \quad \begin{gathered}\text{Dist. for support vectors:} \\ \frac{w^T\mathbf{x} + b}{||\vec{w}||} = \frac{\pm 1}{||\vec{w}||}\end{gathered} \tag{7}$$

$$\begin{gathered}\vec{w} \cdot \mathbf{x} - b \geq 1 \;\; \forall \;\; y_i = +1 \\ \text{or} \\ \vec{w} \cdot \mathbf{x} - b \leq 1 \;\; \forall \;\; y_i = -1\end{gathered} \quad \Rightarrow \quad \begin{gathered}y_i(\vec{w} \cdot \mathbf{x}_i - b) \geq 1 \\ \forall \; 1 \leq i \leq n\end{gathered} \tag{8}$$

[2] A vector perpendicular to a given object, the diving hyperplane in our case

Finally, the core optimization problem is to enlarge the margin as much as possible, such that all the training data points are located on the correct side of the margin, given their actual class. As denoted in [9].

$$\begin{aligned} &max\left(\frac{2}{||w||}\right) \Rightarrow min\left(\frac{w^T w}{2}\right) \\ &subject\ to\ \ y_i(\mathbf{x}_i \cdot w + b) \geq 1 \end{aligned} \tag{9}$$

The optimal solution is array of values α, which are the learned weights to the x's and are equal zero for all but the support vectors ($y_i = \pm 1$). Incorporating the line [4] results in [10].

$$w = \sum_{i=1} \alpha_i y_i \mathbf{x}_i \quad \overset{(4)}{\Longrightarrow} \quad w \cdot \mathbf{x} + b = \mathbf{x} \cdot \sum_{i=1} \alpha_i y_i \mathbf{x}_i + b \tag{10}$$

It is now possible to derive a classification function based on the previous results. When classifying new data, unseen novel observations might fall inside the $[-1 : +1]$ margin, and are therefore classified according their sign (positive/negative) as in equation [11].

$$f(x) = \text{sgn}(\vec{w} \cdot \mathbf{x} + b) = \text{sgn}\left(\sum_{i=1} \alpha_i \underbrace{\overbrace{\mathbf{x}_i}^{\substack{\text{New} \\ \text{Obs.}}} \cdot \overbrace{\mathbf{x}}^{\substack{\text{Support} \\ \text{Vectors}}}}_{dot\ product} + b \right) \tag{11}$$

Notice that the results of the classification function depends solely on the dot product between the vector of the new point and the support vectors ($\mathbf{x} \cdot \mathbf{x}_i$). It is hence possible to classify as shown in [12].

$$\text{class}(x) = \begin{cases} positive, & f(x) > 0 \\ negative, & f(x) < 0 \end{cases} \tag{12}$$

The basic SVM classification algorithm in itself seems to present a solution to a very limited spectrum of classification problems, that is two-dimensional, linearly-separable binary class classification problem. The following paragraph present solution to each on of these hurdles. Notice that replacing the dot product from [11] can be generalized to any number $\mathbb{N}$ of dimensions, clearing the dimensionality restriction. The **Kernel Trick** resolves the linear-separability problem, by mapping to a higher dimension.

Kernel Trick A key element for SVM statistical and computational power is the Kernel Trick [9]. Using the Kernel-Trick, the linearly dimensional training data x are extrapolated onto a higher plane using a mapping function $\Phi : \mathbf{x} \rightarrow \varphi(x)$, with the assumption that on the new plane a better linear separating hyper plane can be found. The mapping is carried out using a *kernel trick*, where by the internal dot product of the linear separator is replaced with a Kernel Function. This Function simulates the redistribution of the original Support Vectors in a richer space, barely incurring an additional computational cost in the process. The linear separator in the new space is not bilinear on the original plane, thus the classification of new observation is also done using the Kernel Function. Whence the data is still not separable using a hyperplane in the new space, the dataset can be mapped to higher still dimension space. Mapping to a higher dimension is illustrated in equation [13].

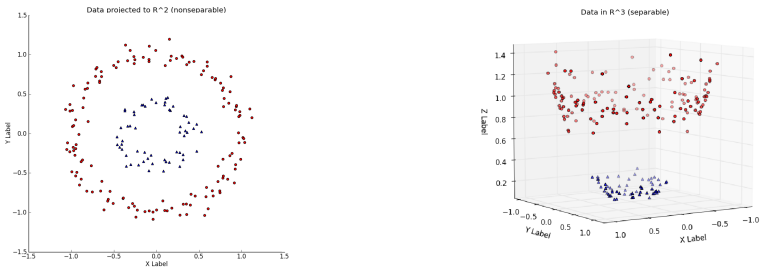


Figure 3: (Left) A dataset in , not linearly separable. (Right) The same dataset transformed by the transformation: $\varphi(x_1, x_2) \Rightarrow [x_1, x_2, x_1^2 + x_2^2]$. *Source*:[10]

$$\begin{aligned} \Phi : \mathbf{x} &\rightarrow \varphi(\mathbf{x}) \\ K(\mathbf{x}) &\rightarrow \varphi(\mathbf{x}_j)^T \varphi(\mathbf{x}_j) \end{aligned} \tag{13}$$

$$\sum_{i=1} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b \rightarrow \sum_{i=1} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{14}$$

The function K [14] represents a *kernel function* and acts as a similarity measure that corresponds to the inner product in some higher feature space. As long as there exists a dimensional space of greater magnitude, in which the kernel function is the dot product of that higher dimensional space, such a kernel is in fact a dot product, and as such can be classified using the normal linear classifier without loss pf generality.

Linear Kernel The kernel stays just the dot product [15] and must not be mapped to a higher dimension. Therefore, the number of dimensions stays the same $\mathbb{N} \to \mathbb{N}$.

$$K(x_i, x_j) = x_i^T x_j \tag{15}$$

Polynomial Kernel Training data is extrapolated using a polynomial kernel function [16] into a higher (six) dimensional space. The kernels represents the similarity of training samples in a feature space over polynomials of the original variables, allowing learning of non-linear models. Additionally, the polynomial kernel add an interpretive intuition by applying interaction variables (products of x_i and x_j).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \left(r + \gamma \cdot \mathbf{x}_i^T \mathbf{x}_j\right)^d, \quad \forall \gamma > 0 \tag{16}$$

In [17,18] it is demonstrated that substituting the dot product with a kernel function results in the dot product on a higher dimension. Specifically I demonstrate it with a polynomial function with $d = 2$.

$$\text{Let } K(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \mathbf{x}_i^T \mathbf{x}_j\right)^2 \quad \text{such that } \mathbf{x} : \mathbb{R}^2 \Rightarrow \mathbb{R}^6 \tag{17}$$

$$\begin{aligned}
\overbrace{K\left(\mathbf{x}_i, \mathbf{x}_j\right)}^{\mathbb{R}^2} &= \left(1 + \mathbf{x}_i^T \mathbf{x}_j\right)^2 \\
&= 1 + x_{i1}^2 x_{j1}^2 + 2\big(x_{i1} x_{j1} + x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2} x_{j2}\big) + x_{i2}^2 x_{j2}^2 \\
&= \big[1 + x_{i1}^2 + \sqrt{2} x_{i1} x_{i2} + x_{i2}^2 + \sqrt{2} x_{i1} + \sqrt{2} x_{i2}\big]^T \cdot \\
&\qquad \big[\underbrace{1 + x_{j1}^2 + \sqrt{2} x_{j1} x_{j2} + x_{j2}^2 + \sqrt{2} x_{j1} + \sqrt{2} x_{j2}}_{\mathbb{R}^6}\big] \\
&= \varphi(x_i)^T \varphi(x_j)
\end{aligned}$$

such that :

$$\varphi(x) = \overbrace{\big[\, 1 + x_1^2 + \sqrt{2}(x_1 + x_1 x_2 + x_2) + x_2^2 \,\big]}^{\mathbb{R}^6} \tag{18}$$

Gaussian Radial Basis Kernel The radial basis function is a prevalent kernel in Machine Learning algorithms.

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)} \begin{matrix} \gamma = \frac{1}{2\sigma^2} \\ \downarrow \\ = e^{-\left(\gamma \cdot ||\mathbf{x}_i - \mathbf{x}_j||^2\right)}, \end{matrix} \quad (19)$$

$$\forall \gamma \ > 0$$

The RBF kernel ranges between zero (in the limit) and one (when $\mathbf{x}_i = \mathbf{x}_j$) and declines as the distance between the points grows smaller. RBF is oftentimes interpreted as a similarity measure, which is why it converges well to the kernel function idea. The feature space of the kernel is of infinite dimensionality.

Sigmoid Kernel The sigmoid function, also known as the Hyperbolic Tangent function, bares semblance to the behavior of the logistic regression by creating an "S"-shaped curve.

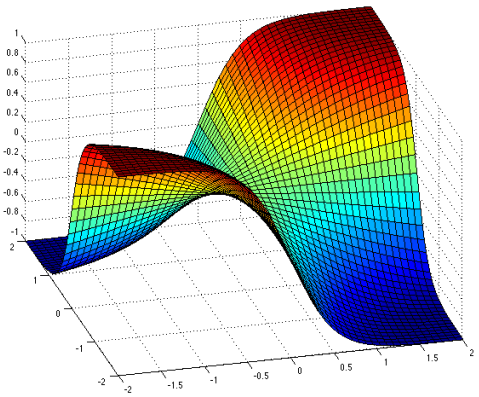


Figure 4: Sigmoid Kernel visualization in d = 3

This kernel originates from Neural Networks, another Machine Learning algorithm. The kernel, when implemented, makes the SVM equivalent to a two layer perceptron neural network.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \cdot \mathbf{x}_i^T \mathbf{x}_j + r) \quad (20)$$

Margin Rigidity or Slack Current research of SVM usage revolves around the concept of *Soft-Margins* [7]. In the case of not linearly separable data, which is usually the case, a *hinge loss* function is added to formula. Its purpose is to penalize for observations, which are on the 'wrong' side of the margin, between the 'sidewalk' and 'separator' that

is. Using this method, observation are allowed to break the constraint present in the hard-margin case [21]. In other words, observations are allowed to cross to the "wrong side" of the margin, but they are being penalized for it, as demonstrated through the variable c in equation 19. A trade-off between margin width, which represents the distinctness of each class, and the amount of misclassified observations can than be specified by the user.

$$max\ (c - y_i(\vec{w} \cdot \mathbf{x}_i - b)), \quad \forall\ c = \begin{cases} 0 & \text{if (8) holds true} \\ 1 & \text{else} \end{cases} \tag{21}$$

The optimization problem with the soft-margins approach is represented in Equation [22]. The variable λ controls the penalty magnitude for observations being on the wrong side of the margin. λ therefore, represents the trade off between margin size and the observation being correctly classified. As the λ approaches zero, the problem converges into the hard margin scenario.

$$\left[\frac{1}{n}\sum_{i=1}^{n} max(c - y_i(\vec{w} \cdot \mathbf{x}_i - b))\right] + \lambda||\vec{w}||^2 \tag{22}$$

Multiclass classification The classification model in itself is natively used for binary stratification, therefore 2 classes only. However, as may be the case in many classification scenarios, data has more then 2 dimensions. For the purpose of Multiclass classification using SVMs, a process known as *class-reduction* is implemented [1]. This is done by either a 'one-vs-all' or 'one-vs-one' approach. With the former method, classifiers for each given class c are trained to differentiate it from the rest of the set. The testing data is then inputed into each of the classifiers and the highest scoring one (having the largest distance from the decision boundary) determines the class for each observation. This is also known as the *winner-takes-all* approach. Using the latter method consists of training classifiers for every combination of two classes out of all the classes. In the case where there are n classes, $\frac{n(n-1)}{2}$ classifiers will be trained as in [23] a max-wins voting strategy, whereby every observations is denoted to one of two classes by each classifier. Afterwards, the final classification is determined by a tally of the votes on each new data point.

$$\begin{aligned} \text{Classes:} \quad & \mathbf{c} = (c_1, c_2, c_3, c_4) \\ \text{Classifiers:} \quad & \mathbf{Cl.} = (Cl_{12}, Cl_{13}, Cl_{14}, Cl_{23}, Cl_{24}, Cl_{34}) \end{aligned} \tag{23}$$

Strengths and Weaknesses Among the advantages of SVMs one can mentions several factors. Firstly, the usage of different kernel - especially user-specified ones, allow for expert knowledge to be integrated into the classification problem. Secondly, the optimization space for the categorization problem is convex, thus making it easily and usually swiftly solvable with appropriate solving algorithms. And thirdly, the usage of soft-margins, or penalizing for error in classification makes the user more aware of the imminent over-fitting, which is the bane of classification problems. This awareness encourages the user to take a more cautious approach and avoid over-specification. The disadvantages include but are not limited to the following. The choice of kernel is left to decide for the user, which could add to the element of human error, especially with dilettante users. Secondly, the problem of Multiclass classification is not solved directly, but rather is adopted for by creating a multitude of partial SVMs, making it not the most effective tool for such scenarios. Finally, the algorithm tends to be memory intensive during when training. Because a value needs to be calculated for each pair of n points, a matrix of size n^2 must be constructed. For example, when training with a dataset of size $n = 1,000$ - the kernel values matrix will be $n^2 = 1,000,000$ large. In optimized modules though, you can get away with only calculating one half of the matrix, since its symmetrical along its diagonal

3.2.3 Artificial Neural Networks

An Artificial Neural Network (ANN) [12] is a mathematical computational model, whose development was inspired by cognitional processes in the brains' nervous system. This sort of network typically consists of numerous interconnected input and output information units. The form of connectivity between these units embodies the connection strength, in a fashion similar to how neurons are linked in the brain. ANNs are predominantly used in Cognitive Sciences and related software such as Artificial Intelligence Systems. Common tasks for such systems are character recognition, facial recognition, financial markets prediction and test mining among other uses.

Intuition A neural network is composed of linked processors called neurons, each capable of executing a simple mathematical operation. However, when combined said networks are capable of sophisticated problem solving. Any given neuron receives inputs i with according weights w. A sum of all weighted inputs is then calculated. When this sum exceeds a threshold value T the output O is equal to one and zero otherwise as in [24]. This operation resembles the operation of a biological neuron, which releases an electric signal when its agitation transcends a certain limit.

$$\langle i, w \rangle = \sum_j i_j w_j = \begin{cases} 1 & , \sum_j i_j w_j \geq T \\ 0 & , \sum_j i_j w_j < T \end{cases} = O \tag{24}$$

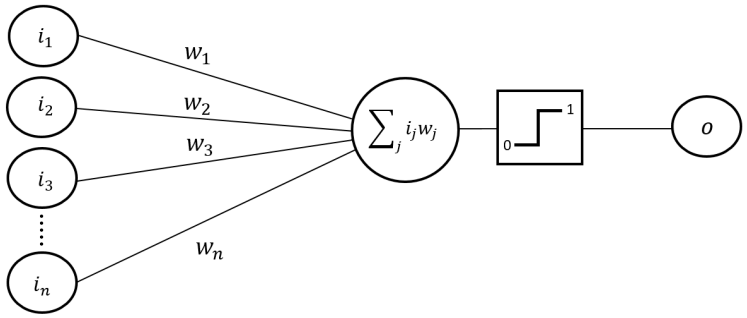
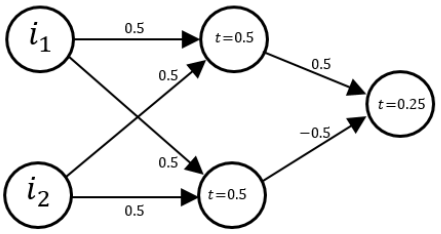


Figure 5: Structure of a Perceptron with n input nodes i and output o. Output 1 when threshold is surpassed and zero otherwise.

The organization of neurons, also called Perceptrons, is crucial to the networks operation and the the input weights are the ones, which determine the pattern to be recognized. Therefore the prime objective when constructing an ANN is determining proper values for the weights. The neurons are then used as logical gateway, which can carry out the "AND" and "OR" operations. A major obstacle in the early days of ANNs was that it was demonstrated that a logical gate of the type "XOR" (exclusive OR) could not be achieved irregardless of the weights on the inputs. This difficulty was later overcome with the introduction of multi-layered ANN.

Multi-layered ANNs are build from several layers of neurons, where all layers except the first(input layer) and the last (output layer) receive their inputs from the lower level of neurons and output into the higher layer of neurons, which in turn use this output as their input. It was

than demonstrated that a "XOR" gateway could be constructed using a 3-layered network [6].



i_1	i_2	o
0	0	0
1	0	1
0	1	1
1	1	0

Figure 6: A three layer Network, creating a XOR gateway.

[4]

Stochastic Gradient Descent

placeholder

Nearest Neighbors

placeholder [2]

Naive Bayes

placeholder [16]

Decision Trees and Ensemble Methods

placeholder [15] [5]

Stochastic Gradient Descent

placeholder

3.3 Unsupervised Learning

placeholder

Gaussian Mixture

placeholder

Clustering

placeholder

Covariance estimation

placeholder

Neural network models (unsupervised)

placeholder

4 Application

4.1 Information Procurement

The main theme of the collected data would concentrate around the topic of Internet sales platforms also known as E-Commerce. Several aspects make the topic beneficial for this research. Firstly, the very nature of E-Commerce. By nature of E-Commerce Platforms, I refer to the fact, that such companies are almost exclusively web-based. It is therefore probable that most of the company's marketing efforts as well as overall news relating to such a company, would circulate first and foremost in the Internet. Secondly, having all company-relevant news attainable foremost from the web, means that such news would seep to social media faster in comparison to news, which are usually covered initially by traditional media such as television and Newspapers.

Search Terms

The data was collected in the form of relevant Tweets from the Twitter Stream API. A Tweet would be considered relevant if it contained a search parameter related contextually to E-Commerce. The initial efforts were concentrated around the web-store Amazon. Amazon appears to the most fruitful search parameter in terms of the quantity of Tweets relating to it. Additional search words that were tested were **Alibaba, Zalando** and **Groupon**. The widespread mention of Amazon in Tweets is however somewhat over-inflated due to the extensive use of Amazon gift cards. Amazon gift-cards have become prominent due their variety of uses. A few examples of common practices involving Amazon gift cards are rewarding users for services, such as polls and questionnaires, enticing people to take part in events or groups, and being offered as general rewards in competitions and games. The plethora of uses, facilitates Amazon gift cards to be viewed as a sort of pseudo-currency in the Internet. In turn this means, that Amazon could be mentioned in a Tweet, despite the context only indicating the Gift-card and being completely unassociated to the E-Commerce platform whatsoever.

Collecting the Data

The gathering of Tweets was executed using a program written by me in the Python programming language. The main module being used in the program was a Twitter Streaming API called Tweepy. Tweepy is an open source interface, which allows communicating with the Twitters servers and sending queries requesting specific information from Twitter's databases. The interface allows for two main type of queries, Rest and Streaming. The former allows looking up information posted on Twitter in the past whereas the latter, as the name implies connects to an active data stream containing a narrowed down flow of Tweets being actively published by Twitter users. Both types of API's are being offered for free to a certain extent, whereas almost unrestricted versions of the same API are offered as a proprietary fee-based product of Twitter. The free version of the REST API is restricted to only looking up Tweets posted in the last two to three weeks. And the gratis version of the Streaming API is restricted to a fire-hose narrowed down to about 15% of the total Bandwidth of all current Tweets. The Tweets from the Twitter servers come in form of JSON strings, which allows for embedding other JSON objects in them, which allows for multi-level storage of Tweet properties. For example, one of the JSON objects integrated in each Tweet JSON object is the USER object for the Tweet-poster. The USER object in turn contains all data publicly available in Twitter about a Twitter account such as, location, date of registration, homepage etc. An additional object of interest is the ENTITIES JSON object, which contains all outside references from the Tweet's text such as, URLs, Multimedia, References to other Tweets or other users. This structure greatly eases the construction an analysis of a Tweet and its features, since most of the necessary data is available from the Tweet self and no further queries about the Tweets and its posting-user are necessary.

elaborate about stages of data collection:

1. Initial data: full of duplicates
2. Filtered data - fewer duplicates and Spam, but still rather very one-subject-centered. Leads to overfitting when classifying
3. still untried - collect up to x (400) Tweets per day, for a duration of 2 weeks

Data cleansing

It was observed that numerous Tweets were being posted more than once and in several occurrences even hundreds of times. These duplicates were being primarily posted by bots, as was evident from a short observation of user profiles belonging the Tweets original posters. Evidently, additional effort was being made by the programmers of the bots to try and mask them by slightly altering the content of the Tweets, or the user account. This was usually done by changing or adding characters to the text, which carry no lingual significance in themselves. Moreover, in furtherance of increasing the bots' credibility as an actual people (Fig. 1), often times entire nets of such bot could be observed, wherein the bots would maintain friendship and following connections among themselves. This in turn, further adding to each of them having a multiplicity of friends and followers contributing to their veil of disguise as real human users of Tweeter. Upon closer observation such accounts reveal their true essence, since most of the content propagated by them is commercial in nature and is repeated verbatim time and again across many of the related accounts *followers* and *friends*, it would be safe to assume that no actual people are behind them.

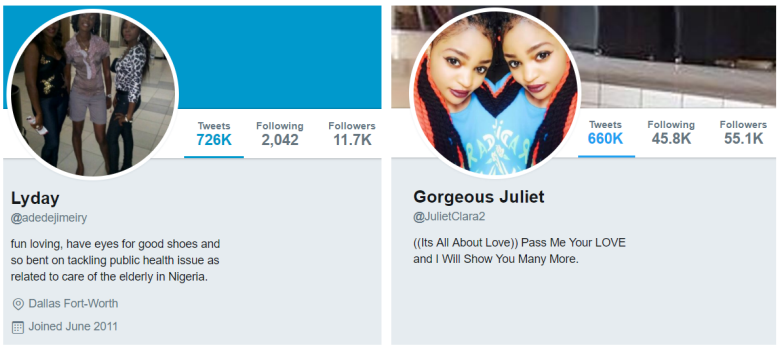


Figure 7: Twitter accounts, which present themselves as actual people

Several precautions were undertaken to try and filter out such bots. A passive precaution which was implemented, was blacklisting users, which had priorly been observed posting Tweets, which were verbatim copies of other Tweets within the same query. The suspected users were added to a suspect database and content originating from them was ignored in future queries. Another action made with the same purpose in mind was a retrospective cleanup of the collected Tweets, based on their content similarity. After closing a collection query, a maximum similarity measure for each Tweet in relation to all other recorded Tweets was calculated using a simple . Following, Tweets which were found to have a maximal similarity score to other previously captured Tweets higher than a predetermined threshold were classified as non-unique copies and were disregarded. As a measure of similarity, the Sørensen-Dice coefficient[17] (Fig. 2) was implemented using the *SequenceMatcher.ratio()* function from the difflib Python module also try Levenshtein . A round of cleanup using this procedure would usually reduce a data set by from one quarter and up to one half of its original size.

$$QS_{XY} = \frac{2|X \cap Y|}{|X| + |Y|}, \qquad QS_{XY} \in [0:1]$$

* $|X|$ and $|Y|$ are the numbers of elements in given Tweets X and Y accordingly

** QS ranges from 0 (completely different) to 1 (identical)

Figure 8: Sørensen-Dice coefficient

4.2 Building the Datasets

Once a list of labeled Tweets is obtained, the next stage is constructing a feature-set to be later passed on as input for training an ML Classifier. The features contained if the feature sets describe certain aspects and characteristics of a Tweet and its owner. Two main approaches to feature sets are found in the literature, *Word-Based* also often called *Bag-of-Words* as can be observed in **insert citation** and *Descriptive* **insert citation**.

4.2.1 Word-Based Features

The former approach simply converts the entire text corpus to a frequency charts of all the words contained within. Words are then selected to act as features in incoming data, which is to be classified. The features are hence a variable list (usually of several thousands in length), where each variable is a boolean representation, indicating the presence or absence of a certain word. Usually the words undergo preprocessing as is common in Natural Language Processing prior to being used as features. The corpora are segmented to lists of words, often omitting articles, proposition and punctuation. Such grammatical structures are critical in human speech and writing ind order to convey ones meaning clearly and explicitly, however for the purposes of more ambiguous classification as in our case, such nuances are avoided for the sake of simplicity. Words are then *stemmed* or *lemmatized*, meaning their are reverted to their grammatical stem - dropping all prefixes and suffixes. This eases the enumeration of words, since it is preferable that the same words in different inclinations would be counted as the same. For example, the word pair *eating* and *ate* would be reverted to their stem *eat*, as well as *apple* and *apples* would be considered as one and the same. Finally, words would be assigned their part of speech (noun, verb, adjective ..) and could be either ignored or incorporated into the feature set, according to the conceived importance of a given part of speech.

4.2.2 Descriptive Features

The latter approach mentioned is based of more generalized view of the Tweet, instead of concentrating on the actual textual content. Descrip-

tive features are aimed at describing the Tweet implicit properties, such as attitude, sentiment, seriousness and trustworthiness. These features detect presence of different symbols, their frequency and consecutiveness. Additionally, unlike the Web-Based approach, non-textual objects such as multimedia, links and mentions of other users and Tweets are also taken into account. Furthermore, features of a Tweets owner are included alongside. Since Tweeters API provides a complete user profile incorporated inside the Tweet data object itself, constructing features describing the user is done simultaneously to features describing the content of the Tweet itself. This approach might be viewed as an *indirect* one, since less obvious properties of the Tweet are used to characterize it.

Descriptive features could be segregated into three distinctive groups. The first will be referred to as text-based features. As the name suggests, the features will mostly denote the presence or lack of specific characters such as emoticons and signs in the Tweets text. Whether a Tweet contains combinations or sequences of certain symbols as well as ratios defining the text also befit this category.

The second tier of features describe any special *Entities* (Tweeter's nomenclature) contained within a Tweet. *Entities* refer to non-textual contents of a Tweet such as media (in form of pictures, sound or videos), URL's linking to external websites, Mentions or ReTweets (Referring to other Tweets or to Tweeter user profiles) and finally Hashtags. A word or phrase preceded by the Hashtag symbol # indicate the association of web content (such as a Tweet or other micro-blogging post) to a specific theme such as an event, news, gossip or any other tidbit [19]. Hashtags are used primarily to simplify looking up Tweets or other social media content by technically associating them with the *hashtagged* topic.

The third tier - subject

5 Genaral Use Cases

placeholder

References

[1] Aly, M. Survey on multiclass classification methods. vol. 19.

[2] Bay, S. D. Combining nearest neighbor classifiers through multiple feature subsets. 37–45.

[3] Bird, S., Klein, E., and Loper, E. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

[4] Bishop, C. M. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and regression trees*. CRC press, 1984.

[6] Burges, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery 2*, 2 (1998), 121–167.

[7] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[8] Demchenko, Y., Grosso, P., De Laat, C., and Membrey, P. Addressing big data issues in scientific data infrastructure. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on* (2013), IEEE, pp. pp. 48–55.

[9] Guyon, I., Boser, B., and Vapnik, V. Automatic capacity tuning of very large vc-dimension classifiers. 147–155.

[10] Kim, E. So, what is a kernel anyway?

[11] Klier, M., and Heinrich, B. Datenqualität als erfolgsfaktor im business analytics. *Controlling 28*, 8-9 (2016), pp. 488–494.

[12] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[13] Mitchell, T. *Machine Learning*. McGraw-Hill, 1997.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duch-

ESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[15] QUINLAN, J. R. *C4. 5: programs for machine learning*. Elsevier, 2014.

[16] RISH, I. An empirical study of the naive bayes classifier. 41–46.

[17] SØRENSEN, T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr. 5* (1948), pp. 1–34.

[18] STREAMING APIS. Twitter Developer Documentation. `https://dev.twitter.com/streaming/overview`. Accessed: July 13. 2017.

[19] "TWEET". *https://www.merriam-webster.com*. Merriam-Webster, Accessed: June 20. 2017.