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

October 11, 2017

Supervision

Prof. Dr. Mathias Klier

Prof. Dr. Leo Brecht

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Social networks are playing an ever-growing role as an information source for a growing segment of the population. The influence of these networks is spreading out to new demographics, becoming more approachable to a variety of audiences of all ages and ranging from the tech savvy to technophobes. However, unlike traditional news outlets , social networks tend to offer anonymity to some extent to its users. This in turn, might eliminate the accountability, which traditional news outlets bear and is likely to be detrimental to the quality of information being spread. Another aspect of the content to be consumed is its relative and subjective interest to specific demographics. Different individuals are interested in different content and the ability to customize the materials presented to them should greatly increase both the user experience and the effectiveness of the news-delivery channel.

The proposed study would examine such information exchanges in the form of messages (referred to hereinafter as *posts*) in social networks while trying to classify the information to a category as well as evaluating its reliability. The purpose is to present a novel method to classify information to custom created labels based on a range of features of the content itself as well as features of the author. The final product would be a classification of the post to a label {News; Not-News} and a measurement of the reliability of such an assigned label could be ascertained on a numeric scale.

The methodology of categorizing information to different labels corresponding to their usefulness was inspired by the work of Castillo, Mendoza, and Poblete [2011]. For the purpose of studying data reliability, Text Mining techniques as demonstrated by Go, Bhayani, and Huang [2009] are to be explored. The field of data quality in social networks has not yet been throughly explored and novel methods of quantifying such quality, if found, could prove to be very useful. Especially in light of the growing dependence of different interest groups on said networks.

An interesting example of social network influence could be observed in the case when the official Twitter account of *The Associated Press* was hacked and a post saying that an attack on the white house occurred, and that president Obama was injured. Due to the automatization of many

stock trading systems, the **S&P500** index underwent a crash causing about 130B $ to be wiped. The capital was mostly restored, however the magnitude of such fluctuations is not to be understated Wang, Kisling, and Lam [2013]. This study would provide a survey of the existing state of research as well as examine, compare and experiment with implementing the different proposed techniques on sample data from social networks. A comparative survey in statistical terms, could then be compiled.

As a case study, the micro-blogging platform **Twitter** would be used. The platform offers an API which allows tapping into the Twitter's data stream and sending queries to their servers. These services are being offered free of charge to some extent. Namely, it is possible to tap to up-to 1% of the real-time stream of data, which flows through Twitter. These should be more than sufficient for the spectrum of this study. Alternatively, several corpora of previously collected Twitter-Stream-Data are also available on the Internet. The bulk of research is to be conducted on a specific topic, mainly popular trends in the branch of E-Commerce platforms such as Amazon, eBay or Otto. With an additional aspect being, broadening the scope and using tapping to a non-thematized data stream from Twitter (the twitter Steaming API allows for a non-filtered query).

1.2 Current State of Research

This study will observe how subjectively important content may be extracted from the Internet in general and from social networks in particular. This ongoing replacement of old-fashioned sources of information by on-line sources is occurring in numerous fields. Namely, Gaskins and Jerit [2012] explore how and to what extent the Internet replaces traditional media such as newspapers and television. They report that the replacement is not uniform across the board, and that some news sources remain overall not replaced by, but rather complemented by the Internet. This is true however, for a smaller fraction of the population, whereas the majority replace traditional news sources completely. This holds especially true for older news outlets, such as radio and printed press. Consequently, the importance of fact checking and setting up filters on social media grows, since Internet news are not subjected to professional scrutiny and review before being diffused.

Due to the growing use and influence of Internet sources, the task of

fact-checking and credibility assessment is of great importance. Generally, when discussing on-line news outlets, the issue of reputation comes into play, in the sense that some news websites enjoy higher credibility, when representing known and well respected organizations or due to establishing such reputation by practicing and upholding truthfulness and objectivity consistently. However, when observing social networks, where every user has the capability to spread (mis-)information, such credibility tend to be a rarity. Castillo, Mendoza, and Poblete [2011] studied this phenomena on social networks, taking the micro-blogging platform Twitter as a case study. They concluded that social media users tend to be quite receptive to content stemming from social media, especially when users are less experienced with Internet use. The research further explores the possibility of using machine learning methods to automatically assess credibility of Internet content with significant results. A Descriptive approach to feature building is undertaken. With this approach, data is initially processed and condensed before being passed into the algorithms for training. Such an approach will also be tested in this study. However, in the case of this study, the rigid definition of credibility is replaced with a more abstract measure, or *subjective interest*.

The concept of using machine learning techniques to provide quality content has been also explored on other platforms. Specifically, Covington, Adams, and Sargin [2016] explored the function of Deep Neural Networks as a recommender system on Google's YouTube. The implemented algorithm learns through past experience which videos and categories are more likely to be favored by the user. To an extent, the users preferences are obviously subjective, which requires the recommender system to be trained and perpetually adjusted in a manner suiting each user personally. This is a classical example for a Neural Networks task. A core strength of such algorithms is learning classification schemes, which are difficult to put into strict rules, but appear intuitive to humans. Another strength of Deep Neural Networks is the ability to derive increasing marginal utility from the extremely large dataset, which Google possesses.

Go, Bhayani, and Huang [2009] also undertook a classification task with subjective or "noisy" labels, to borrow their terminology. They explored the use of non-basic elements of micro-blogging posts from Twitter (Tweets) such as Emoticons, words and details of the posting user for the purpose of classifying the *sentiment* of a given post. The research stud-

ies different combinations of features-approaches and algorithms, which produce very comparable results, with no significant advantages for any one technique. The algorithms they train achieve around 80% accuracy when classifying sentiment. These results show great promise and achieve significant forecasting power.

1.3 Possible Difficulties

Several problems might arise in the course of this study and will need to be addressed. Firstly, the procurement of data could prove to be inefficient since all the samples retrieved from the servers will have to be manually labeled. This process is likely to be very time consuming and limit the experimental data for this study. This difficulty is less likely to affect the practical implementation of any tools or methods produced by this work, since by that stage a validated and significant dataset would already be present. At the least, the very corpus, which I personally will build could be used to kick-start an automatic collection and labeling process.

Secondly, as mentioned priorly - the labels would be assigned by a single individual, myself, and according to my subjective perception of which observations qualify for the label *news* and which do not. This in itself is not problematic, since the very goal is to mimic human interests. The difficulty arises in that only a single person's preferences are being learned. More robust results would be achieved, if the entire experiment would be repeated in several instances with different individuals, each labeling the data according to her own interests. The setting proposed for this study should provide a working example of the method, however repeating the experiment on various people would increase the credibility of the resulting technique.

Lastly, several of the machine learning algorithms to be tested in the course of this work can be computationally expensive and as such might not scale well. For example, when constructing a Support Vector Machine classifier, all data samples must be extrapolated as vectors into euclidean space $\mathbb{R}^n$, with n representing the number of features. Following, the euclidean distance between each pair of vectors must be calculated. The calculation problem therefore grows in complexity exponentially. Nonetheless, such a scenario poses a problem only in the initial stage, since such complications are associated with the stage, in which

classifiers are constructed. Once the classifier has been completed, the action of classification in itself should not require expensive computational resources.

1.4 Research Question

The foremost goal of the proposed research is answering the question of whether Machine Learning algorithms could be implemented as tools for predicting the subjective (*fuzzy*) preferences of users, in regard to their consumption of news content, namely the subject matter which such users are being exposed to in social networks. To some extent an effort is evidently already being made by social network platforms to try and customize the substance and order of posts, which are presented to social network users. For example, such was the case when it was discovered that Facebook experimented on its users by changing how the content was presented to them (Hill [2014]). Tweeter adjusts the users' preferences according to the people and organizations, which users actively choose to follow on the platform. In other words, this customization is done manually by the user, whereas it could prove beneficial to present some degree of automatization to the process and improve the system without the users' explicit effort.

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 to the development of algorithms, which allow computers to learn from presented examples. The key assumption behind the concept being, that computers will thereafter be able to learn from their accumulated experience and automate the process of solving similar tasks. This process of machines learning from examples is coined by term *training*.

One definition of Machine Learning, formulated by one of the fields pioneers, Tom M. Mitchell "*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".

Nowadays machine learning algorithms are predominantly implemented as instruments for the analysis of real-world data at tasks, which an explicit human-written algorithm would prove ineffective. For example, such is the case with problems, which a person would be able to solve, but would not be able to delineate clearly the rules for solving explicitly. Or alternatively, where the rules are not constant, but rather evolve as time progresses. The purpose of teaching a machine to solve such tasks, customarily relates to modeling, prediction or detection of details or certainties about real world phenomena.

A noteworthy example of contemporary implementations of Machine Learning is speech recognition. Cellphones and call routing systems make extensive use of Machine Learning to convert human voice into text and code. Another common Machine Learning task is visual recognition of objects or characters. For such purposes, an algorithm is trained to recognize graphic patterns in images or diagrams. This practice is making breakthrough progress in medicine, namely recognizing malignant anomalies in X-ray or MRI scans.

2.2 Text Mining

Text Mining refers to the practice of extracting information from raw verbal corpora. Such data is obtained by surveying text bodies or other similarly unstructured sources of verbatim data. The initial stage of processing consists of restructuring the data into a form compatible for statistical analysis. This commonly includes but not limited to, segmenting the text into more basic building blocks, such as paragraphs, sentences or even single words. This practice is called **chunking** for phrases and **chinking** for breaking up *chunks* into words. The next stage consists of cleaning up the data by removing non-informative words, which usually serve a grammatical role and are therefore crucial in human language, but tend to be of no use for language processing done by machines. Thereupon, the remaining words are normalized to their base **stem** by removing inflections modifying the word's tense, case number and other grammatical properties. In the professional nomenclature, this is referred to as **stemming** or **lemmatizing**.

These preprocessing methods, which are used in text mining involve statistical pattern recognition, tagging-annotation and frequency analysis. The end goal of text mining is namely, the production of qualitative information out of raw text, often automatically, by using Machine Learning.

Most of the usage of Text Mining methods in this work will be conducted using the **NLTK** module for the Python programming language. **NLTK** is a suite of libraries developed for Natural Language Processing in the Department of Computer and Information Science at the University of Pennsylvania (Bird et al. [2009]).

2.3 Unstructured Data

Data which is derived from the Internet, specifically social networks is oftentimes unstructured. Unstructured data refers to information not organized uniformly or in a predefined data model of sorts. Such a model should be composited harmoniously to the structure in which the data is to be eventually utilized. The aforementioned information is prone to heterogeneity in its composition and as consequence, varies in data types and may contain texts, number, dates as well as multimedia objects simultaneously. Additionally, the data's integrity is susceptible to fluctuation, 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 and formatted as JSON (JavaScript Object Notation) objects. This type of construct has no standard morphology, and many of the Tweet's 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 its structure. Text Mining and Natural Language Processing are two prevailing mechanisms employed for this purpose. More on those in Chapter 4, *Data Collection*.

2.4 Classification

The task of correct assigning (often unstructured) data to a category (or class) from a set of predetermined categories. In the case of Unsupervised Learning, creation of new categories which best segment the data and in case of Supervised Learning, assignment of new examples to user-defined classes. Supervised and Unsupervised Learning, are different approaches to Machine Learning programming, where the former requires continuous input from the user, whereas the latter is autonomous.

Supervised Learning Using this approach, the algorithm is trained on a *labeled training set*. This means that a list of numerous examples along with their features and the predetermined labels, denoting their respective classes are fed as input into a learning algorithm. This algorithm than analyzes the data, and extracts common features, which best characterize every given class. The different algorithms all apply distinct approaches as to how better split these observations. The finished Machine is thereafter usually tested using novel data from the same source as the training data. The performance of the Machine is measured by classifying the testing data and comparing its classification to the actual labels. Some algorithms are even able to optimize themselves during the testing phase. Another common quality assurance test is called Cross-Validation. During Cross-Validation, the data is split to m segments and trains itself in each round using $m - 1$ segments and testing against the 1 remaining segment. In each round, a different segment is left out for testing. This measure assures the Machine's robustness.

Unsupervised Learning In the case of Unsupervised Learning, the task of classification is commonly similar to the task of *Clustering*. During *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 of importance, since it further enhances the prediction power for future novel data. The features are often artificial constructs or rather functions transforming the original data from its native dimensions, in order to create distinguishing attributes of the input data. Examples of such features vary from binary presence indicators of a word or character, interaction variables of two or more features to transformations, linear or otherwise. Some common models in this field are Hierarchical Cluster Analysis (HCA), K-Means and Mean Shift.

2.5 Information Quality

Social Networks incorporate the interactions of millions of individuals, groups and organizations. A torrent of information of such proportions easily qualifies for the qualification of Big Data. Social Networks data streams conform the the widely acknowledged 5 characteristics of Big Data, also knowns as the 5 **V**'s (as in Demchenko et al. [2013]) standing for *Volume*, *Velocity*, *Veracity*, *Variety* and *Value*. Of interest here is the data's correspondence between *Veracity* and *Value*, 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 and relevancy. How much value can indeed be extracted from this data ?

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].

3 Methodology

The approach undertaken in this study is similar to previous works in the field of applied Machine Learning. Initially, a main theme of the content is selected according to the quality and volume it might produce. In the case of this research, the main point of inquiry I wish to study regards Tweets relating to the subject of E-Commerce platforms. Such platforms usually see an abundance of Tweets fulfilling the volume requirement. However, additional cleaning steps are taken to insure the informations quality. These cleaning measures are elaborated in the Application chapter.

On of the main motives of this research is to observe whether social media data can be evaluated in real-time and distilled into practical knowledge. With this consideration in mind, the data used in this work should bear as much resemblance as possible to a live Tweeter data stream. Acknowledging this consideration, the raw data in the form of Tweets in gathered from the Twitter databases. Namely, the data is collected synchronously, as it is intercepted by the servers. The *Streaming Application Programming Interface* (Streaming APIs) is implemented for such use-cases. Employing 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 reduction of stream is due to the complexity in both sending and receiving such a volume of data. Twitter also offers a full non-reduced access through their proprietary *Firehose* API.

The captured data, must then be restructured to fit the data-model of this study, specifically input appropriated for Machine Learning purposes. This includes a preliminary analysis of the data and removal of incomplete observations. The JSON data structure in which Tweets are stored allows for a dynamic non-standard structure, which in turn translates to non-standardized data. Consequently, Tweets must be adapted to conform to a unitary pattern, usually by *flattening* the data into a table. It is also common for Tweets to retrospectively be removed from the Tweeter servers, either by their owners or by Twitter moderators. Such Tweets cannot be analyzed after the fact, 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 restructuring the raw captured Tweets into datasets. For this purpose, *features*[1] are extracted from each observation and converted to a standard array, representing the original Tweet. *Features* are unique properties of the data, which describe it and also possibly its owner, dependent on the approach. Different types of *Features* are used in accordance with the Learning approach undertaken. These approaches are further discussed in Part [4.2].

The previously constructed features are ensuingly analyzed for consistency and correctness. Subsequently, the features are passed into various Machine Learning algorithms, with the purpose of *training*[1] classifiers. *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 and summarized. Additionally, implications are to be drawn about other use-cases. The classification task observed here is subjective or *fuzzy*[2] in nature, since the labels themselves are abstract and arguable. Success in this experiment could prove that similarly subjective classification projects are practical.

3.1 Training Classifier

The purpose of *training* is to create Classifiers. A Classifier is a function, which is fed new observations and automatically recognizes and *labels*[1] them. The intrinsic decision algorithm, through which the Classifier will decide how to allocate a novel observation, usually remains hidden and operates as a black-box of sorts. Seeing that the decision rules could be numerous and considerably not intuitive for human readers, they remain unrevealed. Particularly so, when said algorithms are convolutional. Convolutional Machine Learning schemes, such as Deep Neural Networks, may incorporate numerous stages of parameter construction layer-upon-layer, 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 primarily make use of the scikit-learn module by Pedregosa et al. [2011].

[1] Machine Learning nomenclature

[2] Fuzzy Logic: A system of logic in which statements do not have to be entirely true or false - *Merriam-Webster Dictionary*

The data used for the purpose of this study consolidates 12.520 unique Tweets. Different approaches vary in the percentage of the data used from this corpus. In order to insure robust results, the *Hold-out Method* is used across the board. For each instance of *training* the data is split into two parts - a *training* set and testing set. The *training set* would usually be allocated the larger portion of the data (between 60% and 90%) and would be used, as the name suggests for training the classifiers. The remaining corpus chunk (between 10% and 40%) would be used for testing the classifiers success. Before 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 1 illustrates these schemes with relation to computational complexity and their interoperability.

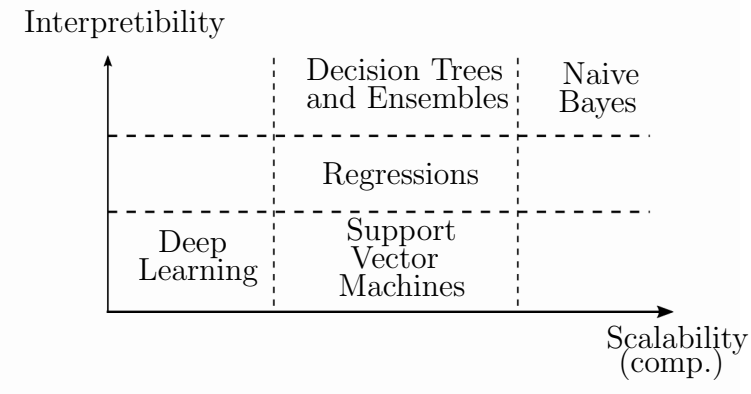

Figure 1: Clustering of Different Classifiers

3.2 Supervised Learning

3.2.1 Regressions

Probably the most basic approach is simply running a regression with all the features as the independent variables and the a numeric representation of the classes as the dependent variable. Such an approach is shown in Equation 1. Some threshold value between the classes must be determined since the estimate for the explained variable, will have a non-deterministic value. Regressions are primarily to be used as a benchmark

for the other algorithms to measure up against.

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

Linear Regression This method builds a linear dependency system between the explained variable (in this case, a numerical representation of the Class) and the explaining variables (Features). With the *Bag-of-Words* approach, each parameter 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 called 'goodness of fit'. We therefore strive to maximize it as much as possible. The more variance covered by our variables, the better we are able to predict the outcome of the classification.

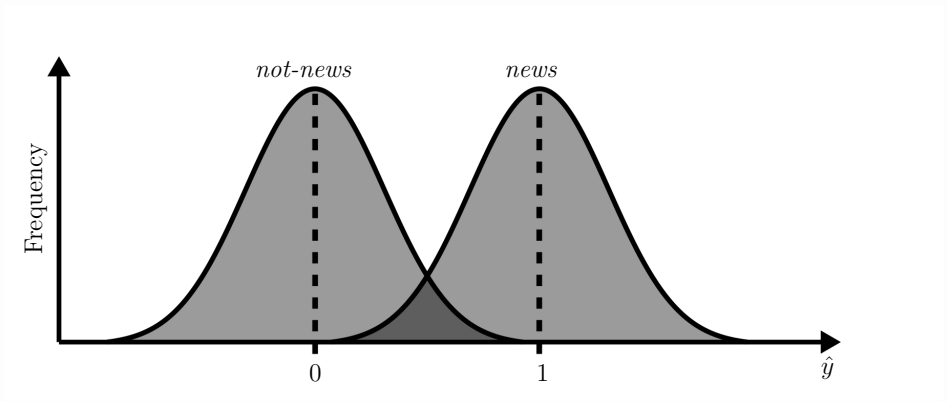

Figure 2: Distribution of predicted $\hat{y}$ around the labels

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 classes as in Figure 2. 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 as presented in Equation 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 (Bishop [2006]). 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 more fluent application as a classification tool and is therefore commonly employed in Machine Learning. Logistic regression analysis is formally represented as follows in Equation 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 Naive Bayes

Probably the most common classification algorithm and usually the go-to classifier when handling text classification tasks (Rish [2001]). The Naive Bayes algorithm is popular because its propensity to perform as good as much complexer classifiers, if not outperform them. This holds true, despite the overly simplified or "naive" assumptions being made at its core. This algorithm is prevailing for numerous classification problems ranging from sentiment analysis, spam filtration and up to classifying data to user-defined classes as is the case in this study. The attitude adopted in this scenario is probabilistic, since it classifies observations according to which class are they most likely to belong to, given their features. The word "Naive" in its name come from its base assumption, which is that the occurrence of a certain feature in a data point is independent from the occurrences of other features. As such, the probabilistic calculus is as in [4], where $\mathbf{X} = (x_1, ..., x_n)$ is a vector of the data features and C is the appropriate class.

$$P(\mathbf{X}|C) = \prod_{i=1}^{n} P(x_i|C) \tag{4}$$

Classification At its heart, the algorithm is based on Bayes theorem [5], which allows for the calculation of conditional probability. In the case of classification, it is the likelihood that a novel data point belong to a given class C_i, given its describing properties (features) $\mathbf{X}$.

$$P(A|B) = \frac{P(X|A) \cdot P(A)}{P(B)} \tag{5}$$

The algorithm thus calculates the posterior probability of the data point to be part of each given class i out of k classes. After which these probabilities are than compared [6]. The classification of the new point is determined by which class is most likely (has the highest probability). In pursuance of calculating these likelihoods, we must first calculate the numerator of [6] as in [7].

$$\begin{gathered} P(C_i|x_1, ..., x_n) = \frac{P(x_1, ..., x_n|C) \cdot P(C_i)}{P(x_1, ..., x_n)} \\ \forall \quad 1 \leq \ i \leq \ k \end{gathered} \tag{6}$$

In [7] we derive that the conditional probability term, $P(x_j|x_{j+1}, ..., x_n, C_i)$ is equal to $P(x_j|C_i)$ based on the assumption that the occurrence of certain features j is independent of the occurrences of all other features $(x_1, x_2, ..., x_{j-1}, x_{j+1}, ..., x_n)$ [4].

$$\begin{aligned} P(x_1, x_2, ..., x_n|C) \cdot P(C_i) &= P(x_1, x_2, ..., x_n, C_i) \\ P(x_1, x_2, ..., x_n, C_i) &= P(x_1|x_2, ..., x_n, C_i) \cdot P(x_2, ..., x_n, C_i) \\ &= P(x_1|x_2, ..., x_n, X_i) \cdot P(x_2|x_3, ..., x_n, C_i) \cdot P(x_3, ..., x_n, C_i) \\ &= \ ... \\ &= P(x_1|x_2, ..., C_i) \cdot P(x_2|x_3, ..., C_i) \cdot ... \\ &\qquad ... \cdot P(x_{n-1}|x_n, ..., C_i) \cdot P(x_n|C_i) \cdot P(C_i) \end{aligned} \tag{7}$$

The calculation in equation[7] allows us to formulate the classification results in equation [8].

$$P(C_i|x_1, x_2, ..., x_n) = \left(\prod_{j=1}^{n} P(X_j|C_i) \right) \cdot \frac{P(C_i)}{P(x_1, x_2, ..., x_n)}$$
$$\forall \ \ 1 \leq i \leq k$$

The expression $P(x_1, x_2, ..., x_n)$ is constant for all classes j : (8)

$$P(C_i|x_1, x_2, ..., x_n) \propto \left(\prod_{j=1}^{n} P(X_j|C_i) \right) \cdot P(C_i)$$
$$\forall \ \ 1 \leq i \leq k$$

Variations of the Algorithm Several derivative forms of the basic Naive Bayes algorithm are also common. These variations usually implement different distributions on the feature probability $P(x_j|C_i)$, thus adding a degree of sophistication to this simplistic idea.

Gaussian A natural assumption when handling continuous data is that the values associated with each class are also continuous and distributed normally. De facto, the data is first segmented by the class c_i. Following, the mean and variance of each continuous variable x_j are calculated for each class c_i. Let μ_i and σ_i^2 be the mean and variance of feature set x associated with class C_i accordingly. The algorithm assumes a normal (Gaussian) distribution for the features, where as the parameters σ_i and μ_i are derived from a maximum likelihood estimation, i.e.,

$$P(x_j|C_i) = \frac{1}{\sqrt{2\pi\sigma^2 c_i}} \cdot \mathcal{e}^{-\frac{(x_j - \mu c_i)^2}{2\sigma^2 c_i}} \tag{9}$$

Multinomial Here the features represent the frequencies of word occurrences which are distributed in a multinomial fashion such that p_i is the probability that an event i occurs in the multinomial $(p_1, p_2, ..., p_n)$. The feature set $\mathbf{x} = (x_1, x_2 m..., x_n)$ can than be visualized as a histogram, in which the frequency of event i (appearance of word i in a given data point) is represented by the hight of the appropriate column. This method is especially common when undertaking the *Bag-of-Words* approach. This will be further demonstrated in the Application part. Therefore, the likelihood of generating a histogram $\mathbf{x}$ is represented in [10].

$$P(X|C_k) = \frac{(\sum_i)!}{\prod_i x_i!} \cdot \prod_i p_{ki}^{x_i} \tag{10}$$

When extrapolated into log-space, [10] turns into [11], where $b = log\big(P(C_k)\big)$ and $\mathbf{w}_{ki} = log\big(p_{ki}\big)$. It is now evident that the classifier becomes linear in logarithmic-space.

$$\begin{aligned} log\big[P(X|C_k)\big] &\propto log\left(P(C_k)\prod_{i=1}^{n} p_{ki}^{x_i}\right) \\ &= log\Big(P(C_k)\Big) + \sum_{i=1}^{n} x_i \cdot log(p_{ki}) \\ &= b + \mathbf{w}_k \cdot \mathbf{x} \end{aligned} \tag{11}$$

Additionally, the product must be adjusted for the case when a certain event (appearance of word i) does not occur. Since non-occurrence will be denoted as a frequency equal to zero, such values must be smoothed out to prevent them from nullifying the entire equation.

Bernoulli This variant assumes multivariate Bernoulli distributions for the features. Particularly, each feature can be represented with a boolean variable. Hence, the classifier requires the feature sets to be passed in the form of vectors composed of binary variables each representing a feature. The classification is based on the decision rule in [12], for class C_k and feature x_i. The probability of class C_k producing the feature x_i is denoted by p_{ki}. We can see that the non-occurrence problem from the Multinomial variant is being explicitly addressed here by penalizing for non-occurrence of a given feature x_i.

$$P\big(\mathbf{x}|C_k\big) = \prod p_{ki}^{x_i}(1 - p_{ki})^{(1-x_i)} \tag{12}$$

3.2.3 Support Vector Machines

Background The original Support Vector Machine (SVM) 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 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 (Burges [1998]). For example, positive and negative observations or analogously membership in or absence from a certain category. A common example is whether a given Email should be classified as Spam or not by the mail server. The rational behind SVMs 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 achieved by drawing parallels through the closest positioned members of both groups. The linear divider lays in the middle between this two parallels. The observations which are positioned on the parallels themselves, define the *curbs* of the so-called *dividing street*, and are dubbed Support Vectors. From them in turn, the algorithm derives its name. Any new observation passed to the trained classifier for classification, will be allocated to the appropriate group, as is defined by its position relative to the separating hyperplane.

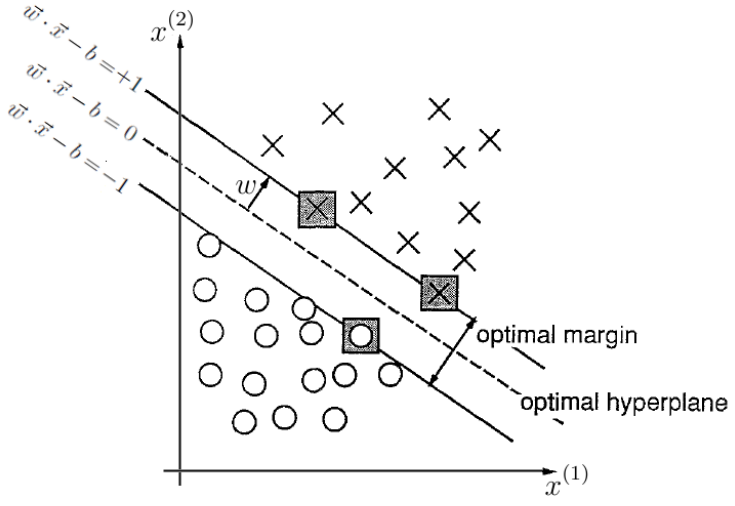

Figure 3: Euclidean Imposition of data. The support vectors, marked with grey squares, define the margin of largest separation between the two classes. *Source:* Cortes and Vapnik [1995].

A dataset of size n as in [13] 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 classification accordingly. A positive (+1) classification implies the presence of the sought-after class, $News$ in our case. The vector $(\mathbf{x}_i)$ is an array of size p, representing the features which compose each observation. In the 2-dimensional example, the 2 features of $(\mathbf{x}_i)$ can be graphically illustrated in euclidean space as the (x, y) coordinates $(\mathbf{x}_i) = [x_i^{(1)}, x_i^{(2)}]$.

$$\begin{gathered}(\mathbf{x}_1, y_1),..., (\vec{x}_n, y_n), \quad y \in \big[-1; +1\big] \\ \mathbf{x} = (x_{i1}, x_{i2}, ..., x_{ip})\end{gathered} \tag{13}$$

The optimal hyperplane segregates the two data classes, creating a maximally wide barrier between the closest observations of the opposing classes. This hyperplane is delineated in [14]. The vector $\vec{w}$ is the normal vector [3] 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 [15].

$$\begin{aligned}&\vec{w} \cdot \mathbf{x} - b = 0 \\ &\vec{w} = (w_1, ..., w_p)\end{aligned} \tag{14}$$

$$\vec{w} \cdot \mathbf{x} - b = \begin{cases} +1 \\ -1 \end{cases} \tag{15}$$

The distance between any point and the separating line [14] is shown in equation [16]. 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 [17].

$$\begin{gathered}\text{Dist. for point i :} \\ \frac{|\mathbf{x}_i \cdot \vec{w} + b|}{||\vec{w}||},\end{gathered} \quad \Rightarrow \quad \begin{gathered}\text{Dist. for support vectors:} \\ \frac{w^T\mathbf{x} + b}{||\vec{w}||} = \frac{\pm 1}{||\vec{w}||}\end{gathered} \tag{16}$$

[3] A vector perpendicular to a given object, the diving hyperplane in this case

$$\begin{array}{l} \vec{w}\cdot\mathbf{x}-b\geq 1 \;\;\forall\;\; y_i=+1 \\ \text{or} \\ \vec{w}\cdot\mathbf{x}-b\leq 1 \;\;\forall\;\; y_i=-1 \end{array} \Rightarrow \begin{array}{c} y_i(\vec{w}\cdot\mathbf{x}_i-b)\geq 1 \\ \forall\ 1\leq i\leq n \end{array} \tag{17}$$

Finally, the core optimization problem is to enlarge the margin as much as possible, subject to all the training data points being located on the correct side of the margin, given their actual class. As denoted in [18].

$$\begin{array}{c} max\left(\dfrac{2}{||w||}\right)\Rightarrow min\left(\dfrac{w^Tw}{2}\right) \\ subject\ to\ \ y_i(\mathbf{x}_i\cdot w+b)\geq 1 \end{array} \tag{18}$$

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 [13] results in [19].

$$w=\sum_{i=1}\alpha_i y_i\mathbf{x}_i \quad\overset{(13)}{\Longrightarrow}\quad w\cdot\mathbf{x}+b=\mathbf{x}\cdot\sum_{i=1}\alpha_i y_i\mathbf{x}_i+b \tag{19}$$

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 [20].

$$f(x)=\text{sgn}(\vec{w}\cdot\mathbf{x}+b)=\text{sgn}\left(\sum_{i=1}\alpha_i \underbrace{\overbrace{\mathbf{x}_i}^{\text{New Obs.}}\cdot\overbrace{\mathbf{x}}^{\text{Support Vectors}}}_{dot\ product}+b\right) \tag{20}$$

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 [21].

$$\text{class}(x)=\begin{cases} positive, & f(x)>0 \\ negative, & f(x)<0 \end{cases} \tag{21}$$

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 solutions to each on of these difficulties. Notice that replacing the dot product from [20] 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 of SVMs statistical and computational power is the Kernel Trick (Guyon et al. [1993]). 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 a higher still dimensional space. Mapping to a higher dimension is illustrated in equation [22].

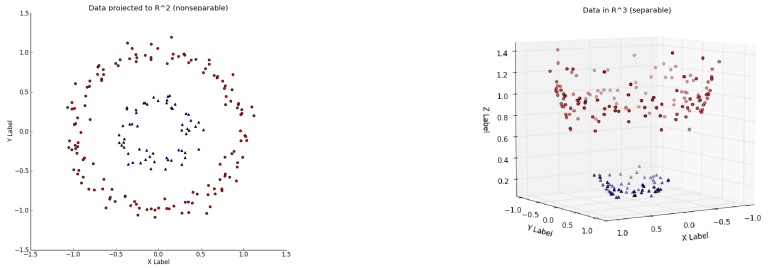

Figure 4: (Left) A dataset in , not linearly separable. (Right) The same dataset transformed: $\varphi(x_1, x_2) \Rightarrow [x_1, x_2, x_1^2 + x_2^2]$. *Source*:Kim [2013]

$$\begin{aligned} \Phi : \mathbf{x} &\rightarrow \varphi(\mathbf{x}) \\ K(\mathbf{x}) &\rightarrow \varphi(\mathbf{x}_j)^T \varphi(\mathbf{x}_j) \end{aligned} \tag{22}$$

$$\sum_{i=1} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b \rightarrow \sum_{i=1} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{23}$$

The function K [23] represents a *kernel function* and acts as a similarity measure which 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 of generality.

Linear Kernel The kernel is in simply the dot product [24] and must not be mapped to a higher dimension. Therefore, the number of dimensions stays the same $\mathbb{N} \to \mathbb{N}$.

$$K(x_i, x_j) = x_i^T x_j \tag{24}$$

Polynomial Kernel Training data is extrapolated using a polynomial kernel function [25] into a higher 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 adds an interpretive intuition by applying interaction variables (products of x_i and x_j).

$$K(\mathbf{x}_i, \mathbf{x}_j) = (r + \gamma \cdot \mathbf{x}_i^T \mathbf{x}_j)^d, \quad \forall \gamma > 0 \tag{25}$$

In [26] and [27] it is demonstrated that substituting the dot product with a kernel function results in the dot product on a higher dimension. Specifically [27] demonstrates the extrapolation of $d = 2$ space to $d = 6$ by employing a polynomial function.

$$\text{Let } K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 \quad \text{such that } \mathbf{x} : \mathbb{R}^2 \Rightarrow \mathbb{R}^6 \tag{26}$$

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

such that :

$$\varphi(x) = \overbrace{\big[\,1 + x_1^2 + \sqrt{2}(x_1 + x_1 x_2 + x_2) + x_2^2\,\big]}^{\mathbb{R}^6} \tag{27}$$

Gaussian Radial Basis Kernel The Radial Basis Function (RBF) is a prevalent kernel in Machine Learning algorithms.

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\left(\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)} \begin{matrix} \gamma = \frac{1}{2\sigma^2} \\ \downarrow \\ = e^{-\left(\gamma \cdot ||\mathbf{x}_i - \mathbf{x}_j||^2\right)}, \end{matrix} \quad (28)$$

$$\forall \gamma \ > 0$$

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 likewise 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.

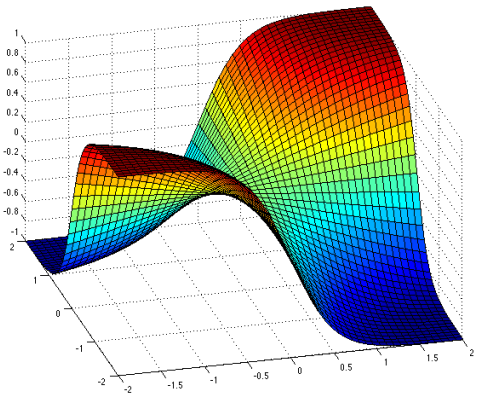

Figure 5: Sigmoid Kernel visualization in d = 3. *Source:* Carrington et al. [2014]

This kernel originates from Neural Networks, another Machine Learning algorithm. The kernel, when implemented, produces an SVM equivalent to a two-layer Perceptron Neural Network.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \cdot \mathbf{x}_i^T \mathbf{x}_j + r) \quad (29)$$

Margin Rigidity or Slack Current research of SVM usage revolves around the concept of *Soft-Margins* (Cortes and Vapnik [1995]). 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, that is between the 'sidewalk'

and 'separator'. Using this method, observation are allowed to break the constraint present in the hard-margin case [18]. In other words, observations are permitted to cross to the "wrong side" of the margin, but they are being penalized for it, as demonstrated through the variable c in [30]. 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.

$$max\ (c - y_i(\vec{w} \cdot \mathbf{x}_i - b)), \quad \forall\ c = \begin{cases} 0 & \text{if (8) holds true} \\ 1 & \text{else} \end{cases} \tag{30}$$

The optimization problem with the soft-margins approach is represented in Equation [31]. 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.

$$\left[\frac{1}{n}\sum_{i=1}^{n} max(c - y_i(\vec{w} \cdot \mathbf{x}_i - b))\right] + \lambda||\vec{w}||^2 \tag{31}$$

Multiclass classification The classification model in itself is natively used for binary stratification, that is 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 (Aly [2005]). 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 [32] 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.

$$\begin{aligned} \text{Classes:} \quad & \mathbf{c} = (c_1, c_2, c_3, c_4) \\ \text{Classifiers:} \quad & \mathbf{Cl.} = (Cl_{12}, Cl_{13}, Cl_{14}, Cl_{23}, Cl_{24}, Cl_{34}) \end{aligned} \tag{32}$$

Strengths and Weaknesses Among the advantages of SVMs one can mentions several factors. Firstly, the usage of different kernels - especially user-specified ones, allows 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 for the user to decide, 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 the training phase. 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 half of the matrix, since its symmetrical along its diagonal.

3.2.4 Artificial Neural Networks

An Artificial Neural Network (ANN) (McCulloch and Pitts [1943]) is a mathematical computational model, whose development was inspired by cognitional processes in a biological brain. AN Networks typically consist 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 applications, of which Artificial Intelligence is a prominent example. Common tasks for such systems are character and facial recognition, financial markets prediction and text mining among other uses.

Intuition A neural network is composed of linked processors called Perceprtons (combination of preception and neurons). Each Perceprton is capable of executing simplistic mathematical operations, however when combined into networks, they are capable of elaborate and sophisticated problem solving. Any given Perceptron receives inputs i with according weights w. These weighted inputs (products) are then summed up. When said sum exceeds a threshold value T, the output O is equal to one and zero otherwise, as in [33]. This operation mimics the operation of a biological neuron, which releases an electric signal when its agitation transcends a certain limit.

$$\langle i, w \rangle = \sum_j i_j w_j = \begin{cases} 1 & , \sum_j i_j w_j \geq T \\ 0 & , \sum_j i_j w_j < T \end{cases} = O \tag{33}$$

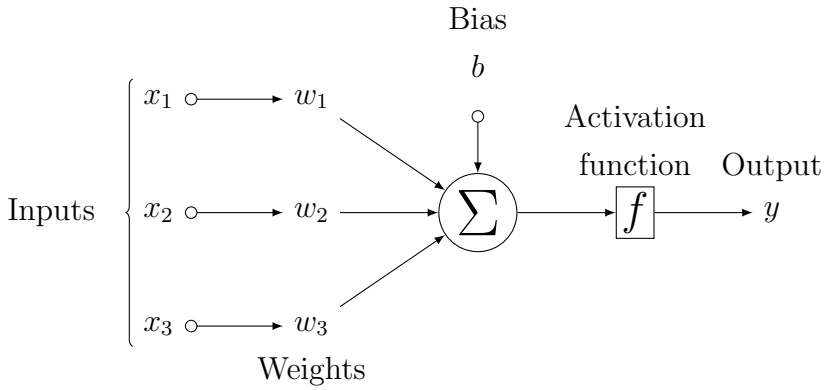

Figure 6: Structure of a Perceptron with 3 input nodes x_i and output y. Output 1 when threshold is surpassed and zero otherwise.

The organization of Perceptrons is crucial to the networks operation. The input weights are the ones, which determine the pattern to be recognized (Bishop [1995]). Therefore, the prime objective when constructing an ANN is determining proper values for the weights. The Perceptrons are used as logical gateway, which can carry out the "AND" and "OR" operations. A major obstacle in the early days of ANNs was the inability to construct a "XOR" (exclusive OR) logical gate, irregardless of the weights on the inputs. This difficulty was later overcome with the introduction of multi-layered ANN.

Multi-layered ANNs consist of several layers of Perceptrons. All layers except the first (input layer) and the last (output layer) receive their inputs from the lower level of Perceptrons and output into the higher

layer, which in turn uses this output as its input. It was demonstrated that a "XOR" gateway could be constructed using a 3-layered network [7].

Input_1	Input_2	Output
0	0	0
1	0	1
0	1	1
1	1	0

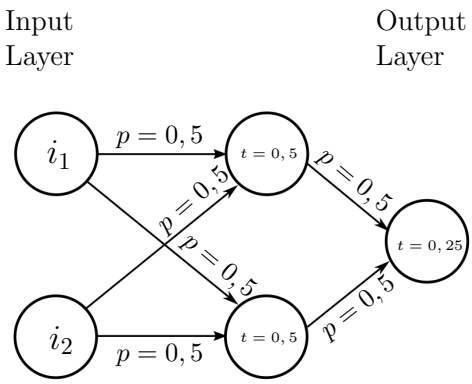

Figure 7: A three layer Network, creating a XOR gateway.

Classification The type of network used most commonly for classification is a Multilayer Perceptron Network (MLP). This falls into the category of feedforward ANNs. A feedforward network is one, which prohibits cycles between its nodes. Thus each layer only outputs information to the next layer, and not to the ones which came before. Typically, a non-linear sigmoid function is used for activation rather than a binary step function as described in [6]. A sigmoid function [34] (often called a Soft-Step) or another continuous function are usually used, since ultimately the algorithm will have to calculate gradients to maximize or minimize an optimization function. Therefore this function should be continuous, differentiable and Real-valued.

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{34}$$

The network is than optimized through the process of **Backpropogation** of errors. During the backwards propagation the initial weights of all the axons (input nodes of a neuron) weights are determined at random. Subsequently, the optimization problem is set by measuring the aggregate error terms in classification (misclassification). This error is calculated as a function of all afore mentioned weights. Next, the aggregate error function must to be minimized by observing the contribution of each weight to the error and reducing it. Finally, the gradient (derivative in vector space) is derived and the weights are adjusted by

Δw respectively as shown in figure 8. This process is repeated, pending we no longer get improvement (lower error terms) or the error reduction is lower than a predetermined threshold.

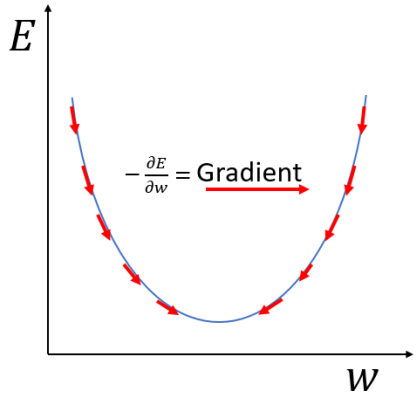

Figure 8: A descent down the error function, taking gradients of w to calculate the steps.

Stochastic Gradient Descent In the basic variant of ANN's, an objective function is optimized by tuning the weights in a manner which reduces the error term gradually with each iteration. In every repetition the weights are adjusted by Δw according to the gradient for each weight over all the observations. Every repetition requires the calculation of the function's derivatives with respect to every data point and their summation within every iteration. This base method is also referred to as Batch Gradient Descent (BGD). This means, that in order to tune a weight w_j we must go through all its derivatives in the gradient and sum them up. Applying this technique requires us to derive the function $m \cdot n$ times in each step of the algorithm, with m and n being the number of data points and the set size of features respectively. The problem with this procedure is its computational costliness which increases as the data set grows larger. Therefore, the standard BGD approach scales poorly. We can hence modify the algorithm using a method called Stochastic Gradient Descent (SGD) to address scalability issues.

To carry out SGD we first shuffle the training dataset to avoid any inherit order or structure present in the dataset. Subsequently, we derive each of the weight modification and adjust after each derivation before continuing to the next data point. Finally, we repeat this process for each data point. Noteworthy is that we are now adjusting after each derivation

instead for all derivations of a data point. Therefore, the descent tempo is now $(m \cdot n)$ $\frac{\text{improvements}}{\text{algorithm round}}$ as compared to the tempo (n) $\frac{\text{improvements}}{\text{algorithm round}}$ of BGD. Hence, SGD progresses m times as fast as BGD. Additionally, the loop over all the data points can be repeated more than once for better results.

Since each adjustment is being made for a single data point at a time, it is much more likely that the adjustment is not actually an improvement and possibly takes us further away (or not nearer) from the global minimum. Nonetheless, the algorithm does converge to the extremum and faster at that, then the BGD approach. Figure [9] illustrates graphically the convergence of both SGD and BGD. The stochastic approach is clearly recognizable with its "squiggly" line. One notable problem with this algorithm is that it might continuously overshoot the local minimum due to its high variance.

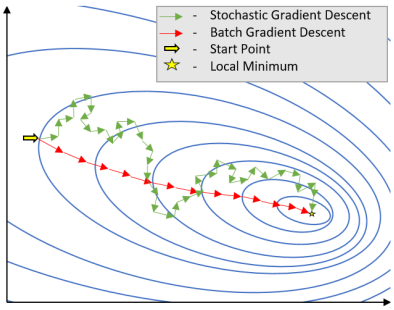

Figure 9: A comparison of the Batch and Stochastic Gradient Descent Approaches

In the example above, an SGD approach is described, in which we fit the weights for a single observation at a time. Another possible modification is to take minute batches of observations and fitting the algorithm on them instead of the entire dataset. This approach is also known as Mini-Batch Gradient Descent and combines the best of both approaches, since it simultaneously addresses the overshooting complication of SGD and is speedier than BGD. Thus, by adjusting the batch size one can control the trade-off between speed and accuracy, where a larger batch results in a more accurate step, but the computation time will also increase.

The mechanism of Stochastic Gradient Descent, at its core, is extremely inaccurate on a single step basis. This is due to the probability of improvement being much lower when compared to Batch Gradient Descent. However, this is compensated for with a more rapid improvement rate, making the accumulated advancement catch up and overtake that of the

normal scenario. The heightened convergence pace is what makes this approach more suitable for classification with extensive training data sets.

Adaptive Moment Estimation ADAM is another approach often used in tandem with SGD. Using ADAM we calculate the first (mean) and second (variance) moment of the gradients and modify the step size for each parameter (feature) in accordance with its frequency. Therefore, the update size is usually more significant for infrequent parameters and more fine for frequently occurring ones.

3.2.5 Decision Trees and Random Forests

Another popular type of classifiers stems from the idea of decision trees (Quinlan [2014]). That is, a flow chart consisting of decision intersections (called nodes) flows from a starting root (the tree stem). The data is further segmented in each following junction until a state is achieved, in which ideally all segments are "pure", containing observation belonging to a single class. Note that Decision Trees and other Ensemble Methods are not well suited for classification of sparse data, due to the large number of features. This means the information gain from each note is very minor. A segment of a Decision Tree implemented in sparse data is graphically illustrated in Figure 10.

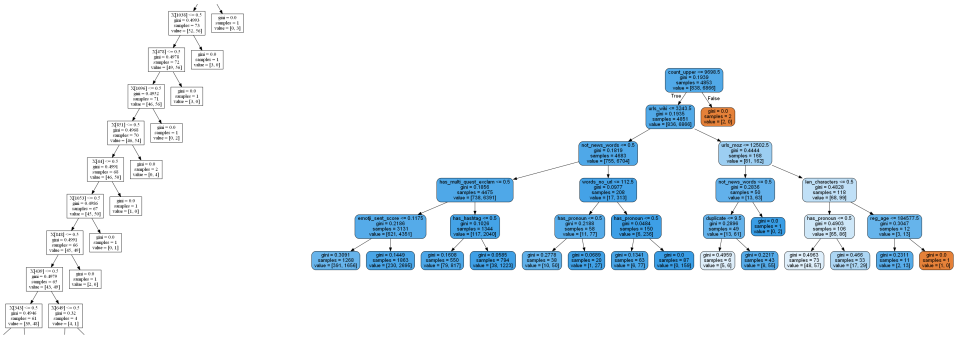

Figure 10: Decision Tree derived from sparse (Left) and non-sparse (Right) data

ID3 Algorithm This procedure of Decision Tree induction is known as the ID3 Algorithm (Quinlan [1986]). More formally, the design functions as follows. We start, as is common in classification problems, with a data training dataset. Each data point (observation) in the set has an

assortment of attributes (features) and is assigned to a certain class. At each stage of the tree construction, including the inception stage, a node is created. This node receives as input a group of observations which belong to 2 or more classes. The node than selects a feature of the data and divides the data according to this feature. The feature selection consideration will be explained subsequently. For each possible value of the previously selected feature, a node is created. If the data inside a node belongs to a single class, it will not be segmented any further and the node will be denoted as a *leaf*. If, on the other hand, the new node contains data points from different classes, we repeat the process of feature selection and segmentation accordingly. Finally, we want to reach a scenario in which, all leaves of the tree contain observations belonging to a single class each. The contents of such leaves may also referred to as *pure subsets*.

Entropy The priorly mentioned decision rule for feature selection, upon which the data will be split in a given node, can be determined by 2 approaches. Both hold the same principle at their core, this being the amount of certainty gained from each split. Hence, the goal is the maximization of certainty, that after a given split, a data point belongs to one distinct class. For example, a feature would be considered a weak feature for splitting, if at a given node it splits the data in a manner where all subsets have relatively similar probabilities of belonging to the different classes. In such an example, no progress was made in the direction of segmenting the data completely. The closer the separation of the data brings us to pure subsets, the higher its ranking as a decision rule at a given node will be. This means, that the lower the entropy values are, the purer the resulting subsets will be.

One measure which helps quantify the quality of this segmentation rule is **Entropy**. The calculation of Entropy is demonstrated in equation [35], with S being the subset passed to the node as input, $i \in \{1 : c\}$ denoting the different classes and p_i the percentage of data point corresponding to class i.

$$H(S) = \sum_{i=1}^{c} -p_i \cdot log_2(p_i) \tag{35}$$

Gini An alternative metric to entropy is the Gini Impurity coefficient. The coefficient denominates the probability of a randomly selected observation to be incorrectly labeled, if the label was set at random according to the actual distribution of labels in the dataset. We compute Gini by summing the probabilities of a given item belonging to a certain class i multiplied by the probability of misclassification. The Gini impurity is demonstrated in equation [36], with i denoting a given class, out of c classes.

$$\begin{aligned} I_G(p) &= \sum_{i=1}^{c} p_i(1-p_i) = \sum_{i=1}^{c}(p_i - p_i^2) \\ &= \sum_{i=1}^{c} p_i - \sum_{i=1}^{c} p_i^2 = 1 - \sum_{i=1}^{c} p_i^2 \\ &= \sum_{i \neq k} p_i p_k \end{aligned} \tag{36}$$

Information Gain Entropy allows us to further calculate the "Information Gain" from each split. This gain is calculated in equation [37], with S being the set of input examples, v a possible value of the attribute A and S_v the subset of S in which the A attribute of the examples is equal to v. The gain is therefore, a weighted average of the Entropies, with respect to the prevalence of a given value v in each subset. Thus, Information Gain also assigns importance to the progress made from a new node. Ergo, a node which classifies few items purely might be outweighed by one which does not split into completely pure subsets, but rather splits correctly more items.

$$Gain(S, A) = H(S) - \sum_{v \in values(A)} \frac{|S_v|}{S} H(S_v) \tag{37}$$

Pruning After the tree is built, it undergoes the process of *pruning*. This means, that branches which add little new information to the classification are removed. The algorithm is considered efficient and fast since, after pruning the amount of features actually used from a dataset is small. In addition, the algorithm is proficient in filtering noise and selecting the best features. Better data attributes are bound to have higher Information Gain values and are therefore more likely to be used in decision nodes.

Random Forest This algorithm builds a boot-strap system on the concept of Decision Trees. The algorithm by Breiman et al. [1984] builds K different Decision Trees. Each Decision Tree T is built using a randomly picked subset S of the complete dataset. Each such tree is trained using the previously mentioned ID3 algorithm. However each tree does not use all the datasets attributes D but rather ones selected at random $d << D$. The trees are not pruned.

As a consequence, each of the K trees is only suitable for a random subset S of the dataset and examines only a subset d of attributes. Each tree is not pruned, therefore it classifies its training data perfectly. When classifying novel data, the observation gets classified using all K tree, and the most commonly appearing classification decides the class. The process is therefore a voting classification using a *Forest* of Decision Trees.

4 Application

The outline of implementing the technique described in the previous chapter is as follows. Initially, a text corpus made out of a collection of tweets in queried from the Twitter databases. The Tweets are requisitioned according to a word parameter. All Tweets containing this parameter at the time of capture will be stored as part of the dataset. Next, the data will be cleaned of duplicates and possible entries made by bots. The cleaned data would then be labeled manually using a custom made Python application. The labeled data can be now used to construct features representing the data. Features are constructed in a fashion allowing them to be passed as input data for Machine Learning algorithms. The process of constructing classifiers based of features is named *training*. The completed classifiers can be further tweaked to improve their accuracy. Cross Validation is conducted to measure the quality of the complete classifiers. Finally, the ready classifiers can be deployed to classify data in real time.

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, 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 firm, would circulate first and foremost in the Internet. Secondly, all company-relevant news are attainable from the web before all other resources. Hence, such news will in all likelihood seep to social media faster than announcements and other stories, which are predominantly covered by traditional media such as television and Newspapers. Therefore, the E-Commerce theme is likely to be widely and swiftly covered on social media, which is beneficial for the purposes of this study, since social media is used as the source of all data.

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**, but proved to be impractical due to less data.

The widespread mention of Amazon in Tweets is somewhat over-inflated due to the extensive use of Amazon gift cards. Amazon gift-cards have become preponderate owing to 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 directly. Such Tweets are generally less valuable for the purpose of gaining knowledge about Amazon itself.

Collecting the Data

The gathering of Tweets was executed using a program written in the Python programming language for the purpose of this research. All software created for this project is available on *GitHub* [4]. A wrapper module called *Tweepy*[5] was implemented for interacting with the Twitter Streaming API. 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 scans the servers for 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 looking up Tweets posted in the last two to three weeks. Similarly, 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.

[4] `https://github.com/burningskies42/Twitter_Analytics`

[5] `http://tweepy.readthedocs.io/en/v3.5.0/`

The data units incoming from the Twitter servers are sent as JavaScript Object Notation (JSON) strings. JSONs are structured hierarchically, which allows for embedding other JSON objects in them recursively. This makes JSON beneficial for Tweets, since it allows multi-level storage of Tweet propertries and objects. For example, one of the JSON objects integrated in each Tweet JSON is the USER object for the Tweet-poster. A USER object, a JSON by itself, contains all relevant data for a Tweeter account. 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 analysis of a Tweet and its features, since most of the necessary data is available from the Tweet itself and no further queries about the Tweets and its posting-user are necessary.

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, often times entire nets of such bot could be observed, wherein the bots would maintain friendship and following connections among themselves. An example is shown in figure 11. This in turn, increases the amount of *friends* and *followers*, further contributing to their guise of 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*. For this reason, it is safe to assume that no actual people are behind these accounts.

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

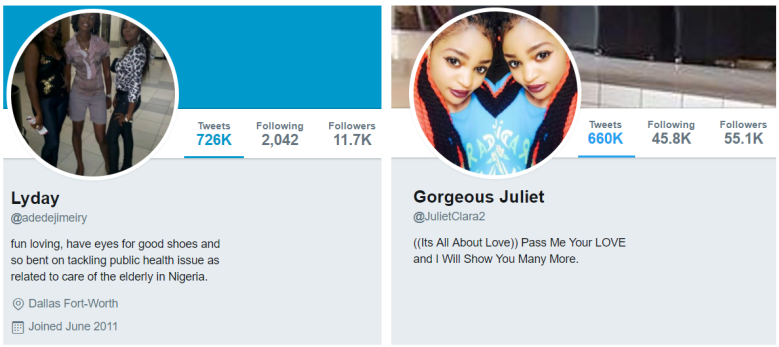

Figure 11: Twitter accounts, which present themselves as actual people. *Source: Twitter*

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 undertaken 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 rule. 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 (Sørensen [1948]) was implemented using the *SequenceMatcher.ratio()* function from the difflib Python module. The coefficient is demonstrated in figure 12 where $|X|$ and $|Y|$ are the numbers of elements in given Tweets X and Y accordingly. A higher QS_{XY} value indicates larger conformity between the X and Y groups, and ranges from 0 (completely different) to 1 (identical). 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.

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

Figure 12: Sørensen-Dice coefficient

4.2 Building Feature-Sets

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 *Word-Based* also often called *Bag-of-Words* and *Descriptive*. A further modification on *Bag-of-Words* is *N-Grams*. The different approaches to feature extraction are discussed in detail in Section 5.2.

4.3 Training Classifiers

Several configuration are to be applied when training the different classifiers, in the interest of quantifying and measuring the quality of classification. Hence, the size of the datasets used for training will vary, to observe the relation between set size and classifier accuracy. Additionally, different amounts of features should be tested, to determine whether accuracy grows linearly as the number of features increase or rather converges after some threshold. Furthermore, the training duration is to be measured to determine effectiveness of training as a time function.

The different classifiers are grouped according to their designation and type as discussed in Section 3.2. Additionally, another classifier type is one, which incorporates all previous classifiers and classifies with a majority vote. This classifier will be referred to as Vote Classifier. An apparent drawback of the Vote Classifier, is the classification duration which will be as long as the sum of classification durations of all its components.

5 Results and Discussion

In the section the classification that took place will be analyzed. The classification were conducted in different configurations of dataset size and number of features. For optimization purposes other characteristics will be discussed such as robustness of the results as well as training and classification time.

5.1 Grouping of Classifiers

The classifiers were grouped into 5 general groups. Additionally, a Vote Classifier, tallying the voted classifications was observed. The groupings were done according to the base algorithm of each classifier. This organization schema is demonstrated in Table 1.

Group Name	Abbreviation	Classifiers
Regressions		Logistic Regression*
Naive Bayes	NB	Simple NB, Bernoulli NB, Multinomial NB
Support Vector Machines	SVM	Linea Kernel, Polynomial Kernel, Radial Basis Function Kernel, Sigmoid Kernel
Artificial Neural Networks	ANN	Multi-Layer Percpetron, Stochastic Gradient Descent**
Ensemble Methods	Ensembles	Random Forest, Adaptive Boosting

* Note that Logistic Regression refers to a classification schema using logistic distribution for possible values of the fore-casted class, rather then an actual regression. Furthermore, Logistic Regression is a simplistic classifier and is demonstrated as a benchmark measure, rather then an actual classifier.

** Refers to Neural Networks which make use of SGD to minimize their cost function by gradually reducing the error terms

Table 1: Classifier Groupings

In the upcoming results analysis the name scheme used in Table 1 will refer to average values inside each such named group.

5.2 Approaches

5.2.1 Descriptive Features

Similar approaches were used to classify Information Credibility on Tweeter (Castillo et al. [2011]). In some sense, the mission of this study is also to quantify the quality of social media content, however, in a very subjective manner.

Every Tweet object originating from the Twitter database is a complex multi-layered structure, which simultaneously describes the posted Tweet itself along with the user profile of its author and any additional media objects related to the Tweet. This approach to feature extraction focuses on deriving the implicit meaning or sense of the tweet, rather than analyzing the verbatim textual content. As previously noted in Section 5.3, two tiers of features are extracted. The former type describes the Tweet object, whereas the latter draws information about the author himself. These features are described in Table 2 and are explained in the following paragraphs. Appendix C describes the general statistics of these features.

Scope	Feature	Description
Message	NUM_STOP_WORDS	Number of Stop-Words in the Tweet
	DUPLICATE	An almost identical tweet was captured in same query
	LEN_CHARACTERS	Number of characters in Tweet text
	NUM_WORDS	Number of words in Tweet Text
	HAS_QUESTION_MARK	Tweet contains question mark
	HAS_EXCLAMATION_MARK	Tweet contains exclamation mark
	HAS_MULTI_QUEST_EXCLAM	Tweet contains either multiple question marks or exclamation marks
	EMOJI_SENT	Sentiment of emoji symbols. TRUE for positive
	EMOJI_SENT_SCORE	Sentiment score magnitude of emoji symbols. Always non-negative
	HAS_PRONOUN	Tweet contains pronouns
	COUNT_UPPER	Percent of upper case letters out of Tweet text
	HAS_HASHTAG	Tweet contains Hashtags
	RETWEET_COUNT	The number of times the Tweet was reposted
	URLS_WIKI	The Tweet contains a URL, belonging to Wikipedia's top 100
	URLS_MOZ	The Tweet contains a URL, belonging to Moz's top 500
	SENTIMENT	Tweet's sentiment score
	SENTIMENT_CONF	Tweet's sentiment score confidence level
User	REG_AGE	Number of days since user registered on Twitter
	STATUS_CNT	Number of Status changes
	FOLLOWERS_CNT	Number of Followers the user has
	FRIENDS_CNT	Number of Friends the user has
	VERIFIED	Whether User account is verified by Twitter
	HAS_DESC	Whether User account contains a description
	HAS_URL	Whether User account has a homepage URL
	MSG_P_DAY	Average number of messages posted per day

Table 2: Descriptive Features

Number of Stop-Words The total number of Stop-Words in the tweets' text. Stop-Words are words, which carry grammatical meaning but usually little to no contextual weight. These words are also usually the most common words in the corpus. The full list of Stop-Words is available in Appendix B.

Duplicate Tweets are queried from the servers using a search term. This feature will receive the value *TRUE*, if another Tweet was found, which has a similarity score exceeding a predetermined threshold. The similarity of each pair of Tweets was calculated using the Sørensen-Dice coefficient as in Figure 12.

Emoji Sentiment All Emoji symbols are extracted from the Tweet and are scored according to the chart in the following URL: `http://kt.ijs.si/data/Emoji_sentiment_ranking/`. The ranking is based on a study by Kralj Novak et al. [2015]. In this paper, Emojis were labeled to have a positive, negative or neutral connotation. The feature EMOJI_SENT denotes the sign of sentiment (positive or negative), whereas the feature EMOJI_SENT_SCORE denotes the strength of the sentiment. When several Emojis are found, their appropriate values are averaged out.

Retweet Count Twitter content can be reposted as part of a new Tweet. This operation is named *retweeting* in the Tweeter nomenclature. When a Twitter user wishes to reference another Tweet in its entirety, the enclosed (Retweet) Tweet will be framed inside the new Tweet and called a Retweet. Additionally, the original post keeps track tracks of the number of times it was reposted by other users. This acts as a measure of the Tweet's diffusion.

Popular Domains Tweets often contain links to objects both inside and outside the Tweeter realm. External items are usually sourced using a Uniform Resource Locators (URL). These URLs, in turn, belong to domains. The features URLS_WIKI and URLS_MOZ denote whether an enclosed URL belongs to one of the top 100 or top 500 most visited domains accordingly. The list of domains are publicly available and are regularly updated. Top 100 most popular domains by Alexa are available on Wikipedia: `https://en.wikipedia.org/wiki/List_of_most_`

`popular_websites` and the top 500 most popular domains are provided by MOZ on `https://moz.com/top500`.

Tweet Sentiment The NLTK corpora contains a set of 10,000 Tweets labeled per hand to have either a positive or negative sentimental connotation. A Machine-Learning algorithm was trained using this set, to classify Tweets their sentiment. The classification in done using a Vote classifier. The feature SENTIMENT denotes the Tweet's sentiment (TRUE for positive or FALSE for negative) and SENTIMENT_CONF denotes the number of underling classifiers of the Vote classifier, which voted for the aforementioned label.

Status Count On the Tweeter platform every post by submitted is regarded as a new *Status* or a *Status Change*. The feature indicated the number of such Statuses since the inception of the Tweeter account. This number includes both Tweets and Retweets.

Verified User accounts are usually not limited when choosing a moniker to go by publicly on Tweeter. This leads to a plethora of accounts impersonating figures or organizations of public interest. Tweeter tries to verify the authenticity of accounts, which it is determines to be of public interest (see: `https://support.twitter.com/articles/119135`). Figure 13 demonstrates a verified and a unverified Tweeter account of U.S. president Donald J. Trump. *Source: Twitter*

Figure 13: Tweeter verified (left) and unverified (right) accounts of U.S. president Donald J. Trump. Notice the blue badge denoting a verified account next to the profile name.

Followers Count The number of Tweeter users actively following this user. A high number of followers suggests, that the person may be views as an Opinion Leader or a person whose carries substantial influence on social media.

Friends Count This features is the mirror image of the Followers Count. It specifies the number of User accounts the current User is following.

5.2.2 Bag-of-Words

This 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 written form in order to convey one's 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.

Therefore, this technique draws meaning from the verbal content of a corpus. From another perspective, one might say that each word carries certain information towards distinguishing different text samples along some trait. We strive to draw correlations between frequencies of certain words and classes to which, we wish to label the samples. Table 3 shows most frequent words to be encountered in the corpus.

	Word	Frequency		Word	Frequency
1	amazon	1857	7889	medieval	1
2	food	248	7890	npr	1
3	whole	212	7891	katerichards09	1
4	new	199	7892	champ	1
5	book	183	7893	prosecutor	1
6	amp	161	7894	khalids	1
7	echo	155	7895	explains	1
8	free	130	7896	commenting	1
9	buy	124	7897	george	1
10	prime	113	7898	lowcost	1

Table 3: Top 10 Most and Least Frequent Words Captured

5.2.3 N-Grams

A development on the Bag-of-Words concept. Using this approach, sequences of n words are extracted instead of single words and used as features. Such an approach has the added benefit of preserving some of contextual meaning of the text as well as discovering reoccurring phrases. This would prove even more beneficial is scenarios, in which the classification would be between positive and negative intonation, such as opinions and reviews. A down side to this method, is that much more feature are created therefore increasing the computational complexity.

N	Possible N-Grams
2	the movie ; movie was ; was not ; not great
3	the movie was ; movie was not ; was not great
4	the movie was not ; movie was not great
5	the movie was not great

Table 4: Possible N-grams of size 2 to 5

For example, using standard Bag-of-Words approach on the sentence "the movie was not great" results in 5 features {the ; movie ; was ; not ; great}. When applying *N-Grams* of sizes 2 to 5 on the same sentence, results in 10 features as seen in Table 4.

	2-Grams	Frequency	3-Grams	Frequency
1	on amazon	938	have just listed	260
2	via amazon	842	available on amazon	96
3	whole foods	366	399 via amazon	81
4	of the	333	for 399 via	81
5	in the	300	panini for 399	70
6	just listed	282	buying whole foods	65
7	have just	265	on sale for	60
8	check out	223	whole foods for	59
9	amazon prime	218	tune in buy	55
10	amazon echo	163	buypayoff amazon amazonproducts	55
	4-Grams	Frequency	5-Grams	Frequency
1	for 399 via amazon	81	2004 panini for 399 via	50
2	panini for 399 via	70	is now on sale for	49
3	tune in buy it	55	germany euro 2004 panini for	47
4	buypayoff amazon products bestbuy	55	might like pumpkinfarmer iartg novel	46
5	2004 panini for 399	50	amazon is buying whole foods	43
6	euro 2004 panini for	50	amazon to buy whole foods	31
7	is now on sale	50	pbscale ai big data cloud	23
8	now on sale for	49	check out this amazon deal	22
9	ai big data cloud	46	ai big data cloud boot	21
10	now available on amazon	39	looking for professional book cover	17

Table 5: Top 10 Most and Least Frequent N-Grams Captured

Table 5 exemplifies the most and least often occurring N-Grams of sizes 2 - 5 words each. The 4-word phrase "is now on sale" appeared 50 times in the large Tweet corpus.

5.3 Descriptive Statistics

This method offers an alternative approach to features, differing from the other two, by being based on a more generalized view of the Tweet. Instead of concentrating on the actual textual content, other non direct properties are observed. Descriptive features are aimed at describing the Tweets implicit properties, such as attitude, sentiment, seriousness and trustworthiness. These features detect the presence of different symbols, their frequency and consecutiveness. Additionally, unlike the Word-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 also included in the input. Since Tweeter's 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 two distinctive groups. The

first tier is text-based features. As the name suggests, these 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, audio or video), 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 # indicates an association of web content (such as a Tweet or other micro-blogging post) to a specific theme. These could be an event, news, gossip or any other tidbit ("tweet" [2017]). Hashtags are used primarily to simplify looking up Tweets or other social media content by technically associating them with the *hashtagged* topic.

The labeling of training examples was conducted by me personally in the process of several weeks. The labeling process was very subjective. Apart from holding up to a few rules of thumb, it is hard to formally model the labeling pattern. This in fact, makes this an ideal task for Machine Learning, since the logic behind it is intuitive rather than strict.

5.3.1 Data Size and Labels

The initial data set consisted of about 12 thousand labeled Tweets captured during a period of several days. The software would launch a listener, which would intercept all Tweets containing the word *Amazon*. The listener was also programmed to ignore all Tweets which contain one of the following words and Expressions: *gift*, *giftcard*, *giveaway*. These phrases were usually found in spam Tweets or Tweets posted by Bots and contained mostly advertising content. The captured Tweets were than labeled manually using custom made software. Out of the 12 thousand Tweets, about 1400 were labeled as "News" and the rest as "Not-News". These ratio between interesting and not-interesting content make practical sense as well, since we strive to boil the data down to only the most subjectively essential information.

Another issue with the data was trending topics. It was common for an announcement to be made about a new product or service, which would

result in a major percentage of all Amazon related traffic to concentrate around the topic. For example, this was the case with the Amazon Echo device. For unclear purposes, some bots would take such an announcement Tweet, which originated from a certified publisher, and retweet it numerous times with minor adjustments to the text. This is turn, escalates the problem of the data being prone to concentrate heavily on some topics and much less on others. A classifier trained on such data is prone to overfitting.

Overfitting Overfitting is a very common issue in the classification field. this means that a classifier is trained in a matter fitting the data very specifically. Therefore, any slight deviation in the new data from the rigid constraints of the trained classifier results in misclassifications. Overfitting could also be viewed as the inability of the algorithm to generalize the rules learned from the data onto new information, which in essence is the purpose of Machine Learning Classification. To prevent this phenomena, the data should he heterogeneous in its content and properties and present as many different learning scenarios for the algorithm to learn from. Overall, overfitting is accepted to some degree in classification, since it is impossible to completely eliminate it. The alternative to overfitting is *underfitting*, which means an algorithm which over simplifies and generalizes to an extent which makes classifications useless.

5.3.2 Data Sparsity

A property which is often encountered when researching Big Data. From a purely technical perspective, Data Sparsity (or Density) describes the amount of zeros in the data, when addressing binary attributes. Stated more simply, when data is immensely heterogeneous an often repeating feature is less likely to occur. In other words, the probability of a feature to be unique is higher. If we compare both approaches to textual features used in this work, *Bag-of-Words* and *N-Grams*, applying the former approach results in features indicating the presence of a single word, whereas with the latter every feature indicates the presence of a string of words. Intuitively, the appearance of a specific single word is much more likely than the appearance of several words in a specific order. For example, the single word "buy" is likely to appear in more samples than the phrase "people often buy", which is a *3-Gram*. Thereupon,

we one may deduce, that data is sparser for *N-Grams* in comparison to *Bag-of-Words*.

5.3.3 Number of Features

Initially, the classification simulation were conducted with the goal of reaching the most precise classifier irregardless of cost and duration. Thereby, the largest possible data set was used with 5000 features, when implementing the *Bag-of-Words* or *N-Grams* approach. The number of features, thus indicates the amount of sought-after words of expressions. Naturally, having more features results in more precise classifiers but at the same time incurs higher computational costs. Furthermore, the greater the training data set is, the more words must be scanned and tabulated by the algorithm, which also has an increasing effect on the computational time. An additional not obvious disadvantage to training classifiers with a large number of features is the risk of overfitting the classifier, as discussed previously.

Table 6 demonstrates the average duration in seconds to train a classifier. The training dataset size is denoted on the horizontal axis and the number of identifying features on the vertical axis. Moreover, the table is split to two parts, each representing another approach to feature construction, *Bag-of-Words* and *N-Grams* are positioned above and below accordingly. Notice the large standard errors, indicated by the numbers in brackets. This strong variation is due to the differences in the training time of the different algorithms. Furthermore, some algorithms (such as SVMs) entail quadratic complexity which makes them scale in a non-linear fashion, causing much longer training times. For example, training a classifier with 3000 features and a training dataset size of 2000 takes an average of 17,63 seconds with a standard error of 16,65 seconds.

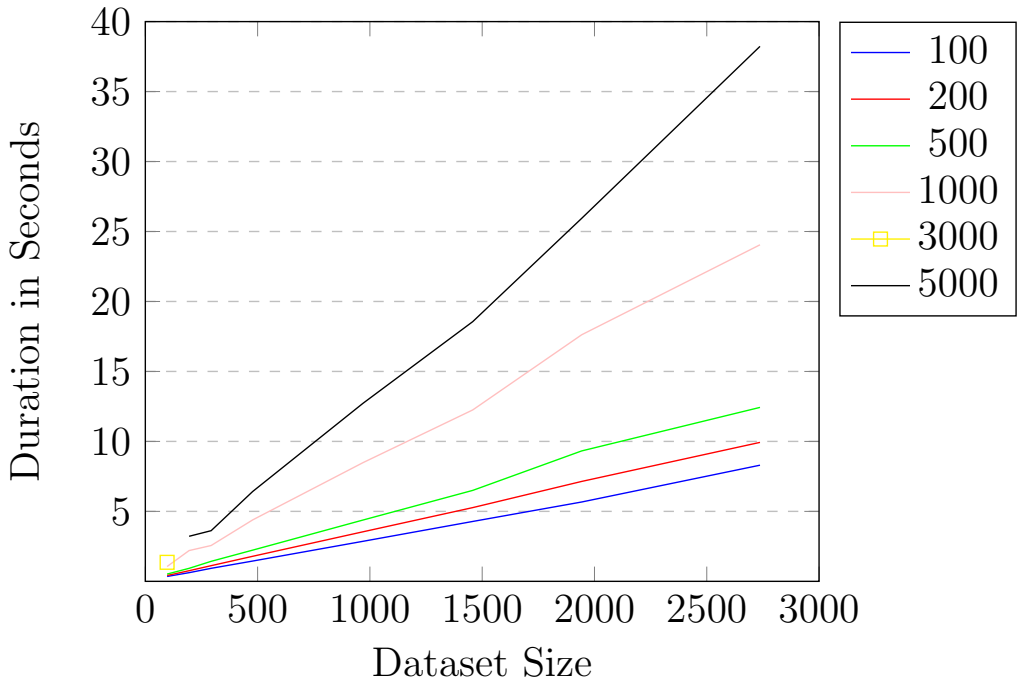

Figure 14: Training Duration as Function of Set Size and Number of Features with the Bag-of-Words Approach

The question of how training duration scales cannot be adequately answered within the scope of this work due to the dataset size limitation. However, the duration seems to grow in a geometric fashion as the dataset size or number of features increase. This growth is illustrated in Figure 14. Nonetheless, the overall duration time seems manageable and would scale even better still when conducted on a more powerful computer. This holds especially true when implementing the mathematically simpler models (not SVMs). A more detailed view of training duration for each classifier is available in Appendix A.

Number of Features	Dataset Size 100	200	300	500	1000	1500	2000	2800
	Bag-of-Words							
100	0,37 (0,94)	0,64 (1,76)	0,90 (2,58)	1,49 (4,17)	2,89 (8,33)	4,25 (12,47)	5,65 (16,61)	8,07 (23,30)
500	0,44 (0,94)	0,76 (1,76)	1,14 (2,57)	1,83 (4,14)	3,61 (8,27)	5,24 (12,41)	7,02 (16,50)	10,34 (23,12)
630*	0,48 (0,94)							
1000		0,93 (1,75)	1,43 (2,56)	2,31 (4,12)	4,65 (8,24)	6,54 (12,33)	8,78 (16,39)	12,70 (23,06)
1149*		0,98 (1,75)						
1597*			1,80 (2,56)					
2319*				3,91 (4,34)				
3000					9,48 (8,65)	12,48 (12,37)	17,04 (16,37)	24,42 (22,83)
3872*					10,29 (8,79)			
5000						18,71 (13,23)	26,26 (17,75)	42,21 (26,37)
	N-Grams [2:5]							
100	0,35 (0,94)	0,62 (1,76)	0,93 (2,58)	1,45 (4,17)	2,86 (8,33)	4,28 (12,49)	5,67 (16,60)	8,30 (23,37)
500	0,42 (0,94)	0,76 (1,76)	1,12 (2,58)	1,79 (4,15)	3,54 (8,29)	5,27 (12,44)	7,14 (16,51)	9,93 (23,24)
1000	0,52 (0,94)	0,93 (1,76)	1,43 (2,57)	2,23 (4,13)	4,38 (8,24)	6,50 (12,40)	9,32 (16,41)	12,43 (23,13)
3000	1,07 (1,29)	2,20 (2,14)	2,56 (2,66)	4,40 (4,50)	8,49 (8,45)	12,25 (12,41)	17,63 (16,65)	24,06 (23,05)
4185*	1,36 (1,53)							
5000		3,23 (2,80)	3,62 (2,84)	6,43 (5,20)	12,73 (9,39)	18,57 (13,23)	25,94 (17,70)	38,23 (24,98)

*The dataset size was too small to extract the denoted number of features, so the maximal number of features was extracted

Table 6: Classifier Training Duration

5.4 Success Measures

During the training simulation numerous classifier types were trained using differently sized training sets and using different amounts of features. Additionally, various approaches to feature construction were tested. Furthermore, the simulations were repeated at least 10 times with each possible constellation of sizes and approaches. This was done to achieve statistically significant measures. In each such simulation round the bootstrap technique was implemented to split the dataset into train-

ing and testing segments with a ratio of 30% : 70% in favor of training. This precaution should assure the results robustness. In order to quantify the qualitative strength of a trained classifier several statistics were observed.

5.4.1 Accuracy

Probably the most fundamental criterion to measure a classifier's quality. As mentioned before the testing segment of the data is passed to the classifier to be classified by it. Following, the algorithm identifies which percent of all classifications were accomplished correctly, thereby corresponding to the actual labels. Thus, this metric denotes simply the part of correct classifications out of all classifications. An accuracy measure of around 50% would mean the classifier does not posses prediction power greater than a random guess or coin toss, making such a classifier practically useless. Measures of less than 50% would mean that a classifier is actually performing worse than a random guess. However, in such cases the algorithm simply reverses the classifiers decision to the complimentary label (opposite) of the classifiers prediction, since obviously the classifier does posses some knowledge about differentiating the classes, but labels them incorrectly. In such cases, the classifier is inverted (in the case of binary classes) and its accuracy is recalculated as $(1 - A_c)$ with A_c denoting the accuracy of the original classifier.

When measuring accuracy, the possibility of accidental correct classifications is not taken into account. To boot, no weight is given to the percentage of False Positives or False Negatives. These measures could be crucial, when it is compelling to capture all positive classifications despite the cost of also classifying more false positives as a byproduct or vice versa. Since accuracy is too rudimentary a measure, it alone is inadequate to measure quality. Table 7 demonstrates the accuracy rating in percent, averaged over the general algorithm groups. For instance, using Support Vector Machines yields an average accuracy of about 79,4% with a standard deviation of 5,48% when applying the *Bag-of-Words* approach to feature construction.

	Naive Bayes	SVM	ANN	Ensemble Methods	Logistic Regression	Vote Classifier
Bag of Words	86,7 (0,06)	79,4 (5,48)	82,8 (1,27)	80,8 (4,02)	85,2 (0,00)	85,7 (0,21)
N-Grams	71,8 (2,04)	73,8 (4,22)	77,1 (1,35)	71,9 (5,70)	79,4 (0,00)	78,1 (0,48)
Descriptive Features	54 (0,00)	72 (5,37)	57,2 (4,37)	77,6 (0,34)	62,8 (0,00)	70,7 (2,49)

Table 7: Classifier Type Average Accuracy in %

Table 7 illustrates that the Bag of Words approach to feature construction outperforms both N-Grams and Descriptive Features when measuring accuracy. Note however that N-Grams has sparser features. Having a more sparse feature system means, that each feature is less likely to appear in numerous samples. When using a relatively compact training set, as in my case, N-Grams might be lacking the sheer volume of data which it requires to achieve robust results.

5.4.2 Cohens Kappa

The Kappa coefficient was first introduces by Cohen [1960] a measure the accordance between numerical sets. In the context of classification, kappa is used for estimating agreement between different labelers, specifically how well the labels given by separate people correspond to one another. Additionally, the kappa may be interpreted as an improved success rating for a classifier compared to the true labels (Castillo et al. [2011]). This measure more informative than accuracy, since it also accounts for the probability of coincidental true classifications. In the case of this study, the Kappa measure can be views as the accord of classification between the human labeler (myself) and the labels assigned by the classification algorithms. The calculation of Cohen's Kappa for two raters classifying N samples into C classes is demonstrates in equation 38, with p_o being the relative agreements of the rates same as accuracy and p_e the estimated coincidental agreement between the raters, as in equation 39.

$$\kappa \equiv \frac{p_0 - p_e}{1 - p_e} = 1 - \frac{1 - p_o}{1 - p_e} \tag{38}$$

The calculation of the likelihood of an coincidental agreement between 2

labeling schemes, with k classes, N samples and n_{ki} denoting the amount of labels k by labeler i.

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2} \tag{39}$$

The Kappa values of commonsense between the actual labels and the machine classifications are presented in Table 8. For this table the largest **weighted** training set was used. The descriptive as well as N-Grams approaches seem to perform poorly when measuring Kappa values. Note however, that the N-Grams approach improves as the number of features increases. This growth alludes to the fact, that N-Grams as features might prove to perform still better, if a larger dataset was available and a larger number of features could be extracted. Note that when using 5000 features, the data in the last 2000 features becomes sparser as compared to the classifiers using 3000 features.

	N-Grams 2:5					Bag of Words					Descriptive Features
№Features	100	500	1000	3000	5000	100	500	1000	3000	5000	
Logistic Regression	33,8 (0,00)	54,8 (0,00)	60,9 (0,00)	59,2 (0,00)	59 (0,00)	53 (0,00)	65,1 (0,00)	64,4 (0,00)	68,3 (0,00)	70,5 (0,00)	14,2 (11,51)
Naive Bayes	32,6 (0,79)	37,8 (8,99)	42,3 (4,24)	50,4 (8,32)	43,9 (3,89)	49,1 (6,69)	63,3 (1,57)	67,5 (0,37)	72,0 (0,73)	73,3 (0,12)	18,7 (15,62)
Support Vector Machines	32,9 (0,61)	48 (1,53)	53,7 (2,44)	53,4 (7,47)	47,9 (8,31)	50,3 (1,27)	57,3 (5,34)	54,9 (7,29)	57,2 (12,78)	58,9 (10,83)	27,7 (18,63)
Neural Networks	32,5 (1,84)	53,3 (1,54)	57 (2,45)	58,9 (2,24)	54,3 (2,68)	51,2 (2,93)	62,6 (2,66)	62,1 (2,72)	65,9 (2,06)	65,5 (2,55)	11,1 (9,07)
Ensemble Methods	33,8 (1,08)	47,2 (9,11)	47,5 (11,33)	47,1 (12,7)	44,1 (11,23)	50,6 (1,46)	56,7 (7,61)	55,4 (7,5)	58,3 (9,82)	61,8 (7,96)	40,3 (15,23)
Vote	34,6 (0,34)	53,1 (0,56)	58,6 (0,34)	59 (0,62)	56,3 (0,95)	52,6 (0,7)	64,7 (0,39)	65,6 (0,51)	70,7 (0,45)	71,5 (0,41)	26,6 (15,04)

* Standard deviation is given in the brackets

Table 8: Average Kappa Values per Classifier and №of features used in training

5.4.3 Confusion Matrix

Sometimes also referred to as *contingency table* (Fawcett [2006]). This table presents a more in-depth analysis for a classification quality. Typically, the confusion matrix shows several calculated measures. *True-Positive rate* (*recall*), *False-Positive rate*, *precision* and the F_1 measure. Whereas, the partition of classes to positive and negative categories is relative to the target of the classification. For example, in this study I wish to extract news-worthy content out of the whole data-stream. Therefore, the Positive notation refers to the class *news* and Negative to *not-news*.

		True Class	
		Positive	Negative
Predicted Class	True	**T**rue **P**ositive	**T**rue **P**ositives
	False	**F**alse **N**egatives	**F**alse **N**egatives

Recall also called True-Positive rate, is the relative percentage of correctly classified *news* samples out of all the samples, which were classified as *news*.

False-Positive rate accordingly refers to the relative part of *not-news* labels, which were mistakenly classified as *news* out of all *not-news* items.

Precision indicates the relative part of actual *news* out of all items, which were classified as *news*. Precision shows how accurately the *positive* class is assigned. These calculations are shown in equations 40 and 41.

$$\text{fp-rate} = \frac{\text{FP}}{\text{N}}, \quad \text{tp-rate} = \frac{\text{TP}}{\text{P}} = \text{recall}, \quad \text{specifity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{40}$$

Additionally, the confusion matrix is instrumental for the construction of the derived effectiveness, denoted by F (or F_1). The value is a weighted average of precision and recall, ranging from 0 (worst) to best (1). The following values are presented in Table 9. From these results, we can note that both precision, recall and F_1 values increase as the number of features increases, thought with a diminishing effect. The Descriptive features approach lags far behind both Bag-of-Words and N-grams, out of which Bag-of-Words seems to achieve more prominent results. All classification algorithms produce very comparable results with the Bag-of-Words approach, around 80% for all 3 measures. These values infer us, that the classifier are significantly better than an arbitrary classifier (coin-toss).

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}, \quad \text{F}_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{41}$$

		N-Grams 2:5					Bag of Words					Descriptive Features
Number of Features:		100	500	1000	3000	5000	100	500	1000	3000	5000	
Logistic Regression	Prec.	0,82	0,86	0,86	0,91	0,89	0,81	0,83	0,85	0,84	0,86	0,64
	Recall	0,45	0,65	0,73	0,67	0,68	0,68	0,82	0,80	0,84	0,85	0,34
	F_1	0,58	0,74	0,79	0,77	0,77	0,74	0,82	0,82	0,84	0,85	0,34
Naive Bayes	Prec.	0,93	0,81	0,86	0,95	0,88	0,75	0,80	0,85	0,83	0,84	0,56
	Recall	0,37	0,58	0,57	0,55	0,55	0,75	0,85	0,84	0,90	0,91	0,25
	F_1	0,53	0,64	0,66	0,69	0,65	0,74	0,82	0,84	0,86	0,87	0,31
Support Vector Machines	Prec.	0,85	0,86	0,86	0,88	0,85	0,81	0,81	0,84	0,84	0,85	0,57
	Recall	0,42	0,57	0,66	0,63	0,59	0,65	0,75	0,71	0,72	0,74	0,45
	F_1	0,56	0,69	0,74	0,73	0,69	0,72	0,78	0,76	0,76	0,77	0,45
Artificial Neural Networks	Prec.	0,83	0,85	0,84	0,89	0,85	0,81	0,81	0,84	0,83	0,84	0,40
	Recall	0,42	0,64	0,72	0,68	0,67	0,66	0,81	0,79	0,83	0,82	0,46
	F_1	0,56	0,73	0,77	0,77	0,75	0,73	0,81	0,81	0,83	0,83	0,39
Ensemble Methods	Prec.	0,84	0,88	0,89	0,90	0,90	0,82	0,83	0,84	0,88	0,87	0,68
	Recall	0,43	0,54	0,56	0,54	0,50	0,65	0,72	0,71	0,68	0,73	0,50
	F_1	0,57	0,66	0,67	0,66	0,63	0,72	0,76	0,76	0,76	0,79	0,54
Vote	Prec.	0,87	0,89	0,89	0,92	0,90	0,81	0,83	0,86	0,85	0,86	0,75
	Recall	0,42	0,60	0,68	0,66	0,63	0,68	0,82	0,80	0,85	0,86	0,31
	F_1	0,56	0,72	0,77	0,77	0,74	0,74	0,82	0,83	0,85	0,86	0,40

Table 9: Confusion Matrix Measures

5.4.4 Receiver Operating Characteristic

Receiver operating characteristic (ROC) provides a graphic visualization and a static measure, which is derived from it, and both incorporate most of the elements of the Confusion Matrix into a single value. ROC has seen increasing use in the fields of machine learning and data mining in recent years and often serves as an unequivocal measure for a classifier's effectiveness.

ROC graphs present the true-positive and false-positive rates on the y-axis and x-axis accordingly. Every point plotted on the graph depicts the trade-off between the 2 rates of detection. Points A through D in Figure 15 represent 4 different classifiers in ROC space. Classifier A, for example, is superior to both B and C, since it has a higher TP-rate and at lower or equal FP-rate. The dotted line represents classifiers, with equal FP and TP rates. Such classifiers practically assign equal probabilities to classifying correctly and incorrectly, making them no better then a random guess. Therefore, all points below the dotted line are classifiers worse than a random guess. However, when a classifier (point D) does in

fact consistently misclassifies samples, it can be inferred that the classifier has some knowledge to differentiate between the different classes, but applies it erroneously. Such a classifier can be inverted (simply, in a binary classification scenario). For example, the classifier A is an invert of classifier D. In order for a classifier to rise above the diagonal, it must make use of the information gained from the data.

Some classifiers assign binary deterministic labels to sample (Positive or Negative), as is the case with Ensemble methods for the example. Such algorithms produce a single classifier, which can be mapped to ROC space and presented in a single confusion matrix. Whereas, other classifiers assign probabilistic labels, which assigns probabilities for each label.With uch probabilistic classifiers, we can manipulate the threshold between a positive and negative classification and plot the the classifiers TP and FP rates in ROC space. The threshold varies between zero (classify all as negative) and one (classify all as positive). Subsequently, a function curve will be plotted on the graph. The area under this plotted graph will be called Receiver operating characteristic - Area Under Curve or RAUC. This value will always represent the relative part of the unitary square, which is below the curve and as such will range between zero and one. Calculating this measure allows us to condense the quality measure of a classifier to a single metric.

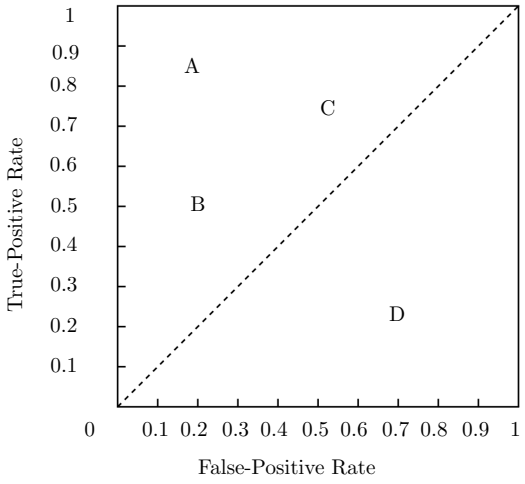

Figure 15: ROC Graph

One half of the unitary square, as in Figure 15, will always be under the diagonal, which is equivalent to a random classifier. As follows, all trained classifiers must be to some degree better than a completely ran-

dom classifier. Therefore, a RAUC score higher than 0.5 is a requirement for a classier. The RAUC represents the goodness of a classifier, by measuring the probability that the classifier will label a random Positive sample higher than a randomly chosen Negative sample.

Naive Bayes :	N-Grams	Bag of Words	Descriptive
Bernoulli	69,57 (0,00)	86,51 (0,00)	66,20 (9,69)
Multinomial	74,06 (0,00)	86,64 (0,00)	56,37 (0,14)
Regular	72,44 (0,00)	86,65 (0,00)	53,17 (1,17)
Support Vector Machines :			
Linear Kernel	76,19 (0,00)	81,57 (0,00)	63,06 (12,56)
Polynomial Kernel	67,24 (0,00)	86,62 (0,00)	66,82 (2,64)
Radial Basis Function Kernel	74,74 (0,00)	86,62 (0,00)	66,82 (11,36)
Sigmoid Kernel	77,93 (0,00)	77,57 (0,00)	58,73 (6,51)
Artificial Neural Networks :			
MLP	78,42 (0,28)	83,89 (0,25)	56,27 (4,21)
SGD	76,05 (0,78)	81,65 (0,80)	54,73 (4,71)
Ensemble Methods :			
Adaptive Boosting	66,71 (0,21)	77,15 (0,00)	67,58 (9,62)
Random Forest	77,59 (0,61)	84,77 (0,62)	69,14 (9,04)
Logistic Regression	79,58 (0,00)	85,25 (0,00)	56,79 (6,04)
Vote Classifier	78,27 (0,47)	85,76 (0,20)	62,28 (8,68)

* RAUC with largest weighted dataset (~2800 training samples) and 5k features, where applicable

** Number in brackets is standard deviation

Table 10: RAUC values

The values in Table 10 demonstrate the RAUC values across the different feature approaches and algorithms. The strongest performer yet again is the Bag-of-Words approach. N-Grams presents slightly weaker results, but as mentioned before, with a larger dataset it could have proven to be as strong as Bag-of-Words. Descriptive features seems to be a weak approach along all measurements with RAUC being no exception.

An additional point of interest is that Naive-Bayes, Polynomial and RBF SVM kernels seem to perform best according to the RAUC. This comes as no surprise, since the other measuring metrics hinted at an identical conclusion. Naive Bayes, despite having very simplistic assumptions at its core, performs at least as well, if not better, then the more complex models.

6 Conclusion

The utilization of the Internet as a news source has expanded rapidly in the last few years. In particular, social networks have been out-competing traditional outlets, due to their abundance of constantly updated information, which is laid at the fingertips of practically any computer user. However, unlike conventional old-fashioned resources the likes of television and printed press, publishers of Internet content are not subjected to rigorous review. As a result, such publishers are not bound by reliability and accountability for their content. Any private individual is free to spread any sort of content she wishes, as long as no laws are infringed upon. Another perspective of this widely consumed content is not its factual truthfulness but rather its relevance and subjective interest for the user. Analogously, traditional media offers journals and TV programs covering a specific topic. Veritably, the user experience of social networks could be greatly improved by customizing the posts which is presented to them to fit their personal interest scheme. Furthermore, this customization is to be done in real-time and on a per-user basis to maximally improve the user experience.

The data used for this experiment was collected automatically from an API provided publicly by Tweeter. The collection in itself did not pose any difficulty. However, the quality of the content obtained was inherently low. By way of illustration, the data was riddled with numerous reposted copies of the same Tweets as well as a plethora of advertisement material. In fact, the very type of content that I wished to reduce in a users data-feed was significantly present. For this reason, the data was cleaned by removing copies and obviously commercial posts. The process of cleaning was also automated, so that future collection of data we already be able to avoid low-quality content, as it is being gathered. Furthermore, post containing words obviously associated with commercials (Giveaway, Giftcard etc.) were also removed. The resulting data had to be than manually labeled by a person to be used as input for the machine learning algorithms. This process proved to very time consuming leading to a moderately sized dataset. However, a key benefit of such an approach is that once the manually labeled dataset allows for sufficiently accurate classifiers, the data can be collected, labeled by the machine and used as training material for future iterations of the classifier. Thus, the machine becomes self improving.

This research analyzed different machine learning algorithms as well as their construction for the purpose of adjusting a subjective user-interest scheme. Various aspects of training were inspected. Primarily, the cost of construction (in terms of duration) and accuracy. It was established that the different algorithms perform very comparably. A compound voted classifier, composed of sub-classifiers - each trained using another method, was also experimented on. Such a classifier provided more stable and yielded consistent results but at a substantially higher computational cost. Moreover, the theoretically simpler models proved to produce slightly higher accuracy scores, which is promising since their computational simplicity provides intuition to the process, namely Logistic Regression and Naive Bayes.

Several techniques of feature extraction were examined. Descriptive features did not manage to show encouraging results and did not appear to be reach significant forecasting irregardless of configurations. This might be to the very fuzzy or ambiguous labels. The N-Grams approach performed exceedingly better and showed promising results, reacting positively to larger datasets. Future experiments should try and harvest more data, since the potential of the algorithms did not seem to peak with the available dataset.

6.1 Theory and Practice

The methodology presented here was shown to be a functional tool in th task of classifying social network content to better suit the preferences of individual users. Through the same process and with practically unchanged training procedures, the algorithms might be adjusted to gather useful content for different interest groups with various goals.

One such possible example is observing the public perception of a company or a newly released product. The technique developed here would allow to closely monitor relevant posts and through an through an extra step summarize the public Sentiment (Go et al. [2009]) regarding the company or product. This could be used by investors, who consider investing in a company of branch, or by the company itself as an instrument for feedback from the public.

6.2 Strengths and Weaknesses

The algorithm presents significant classification power but an initial phase of labeling is rather time consuming meaning a base line classifier must be first completed as a vanilla classifier. This vanilla classifier could be later integrated into the social network user account, where it could behind the scenes to mine further knowledge regarding the preferences of unique users. Thus, the algorithm can not be deployed fully customized but rather generally tuned for specific population segments sharing similar characteristics and further adjusted as more data is collected about the habits of the user.

The methods described in this study describe classification of extremely *fuzzy* or vaguely defined classes. As such, the machine is tasked with mimicking human intuition. This scenario requires large masses if heterogeneous data to succeed. The procurement and transformation of such data time extensive. Subsequently, avoidance of the classical over-fitting problem of machine learning classification proves to be a challenge. In other works, finding the balance between filtering out too much or too little content is another obstacle. On the other hand, since the algorithm is theoretically self-improving the problem of over-fitting or rather adjusting for the user's preference should be resolved or at least alleviated over time.

6.3 Future Research

The general classification theme was tried on a single platform, Tweeter that is. As such, the study was subjected to the unique system, which characterizes it, including the post size limit and the ability to embed objects and different media in post. An alternative system which is often used in a similar fashion is Reddit, which also is widely used social news source. Using Reddit might prove beneficial because of its more simplistic and less closely monitored interactions between the user. Another possible source of social network content is stockTwits, which is a platform closely resembling the design and interface of Twitter, but concentrating thematically on the field of business and financial markets. StockTwits might therefore prove to be an essential resource in constructing tools for investors following trends around traded companies.

An additional direction, which was out of scope for this study, is un-

supervised learning. This approach should be prove to be simple in the preprocessing stage. The data must not be labeled and categorized before passed on as in put for the algorithms. Moreover, a clustering algorithm might shed light on possible groupings of data previously not explored. Furthermore, these grouping are not bound by the binary class system of the current study, further segmenting the data into finer subcategories.

References

Mohamed Aly. Survey on multiclass classification methods. volume 19, 2005.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc., 2009.

Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

André M Carrington, Paul W Fieguth, and Helen H Chen. A new mercer sigmoid kernel for clinical data classification. In *Engineering in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE*, pages 6397–6401. IEEE, 2014.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM, 2011.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

Yuri Demchenko, Paola Grosso, Cees De Laat, and Peter Membrey. Addressing big data issues in scientific data infrastructure. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages pp. 48–55. IEEE, 2013.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Benjamin Gaskins and Jennifer Jerit. Internet news: Is it a replacement for traditional media outlets? *The International Journal of Press/Politics*, 17(2):190–213, 2012.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(2009): 12, 2009.

Isabelle Guyon, B Boser, and Vladimir Vapnik. Automatic capacity tuning of very large vc-dimension classifiers. pages 147–155, 1993.

Kashmir Hill. Facebook manipulated 689,003 users' emotions for science, June 2014. URL `https://www.forbes.com/sites/kashmirhill/2014/06/28/facebook-manipulated-689003-users-emotions-for-science/#466fecb7197c`. [Online; posted 28-June-2014].

Eric Kim. So, what is a kernel anyway? `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`, 2013.

Mathias Klier and Bernd Heinrich. Datenqualität als erfolgsfaktor im business analytics. *Controlling*, 28(8-9):pp. 488–494, 2016.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. Sentiment of emojis. *PLoS ONE*, 10(12):e0144296, 2015. URL `http://dx.doi.org/10.1371/journal.pone.0144296`.

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

T. Mitchell. *Machine Learning*. McGraw-Hill.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1): 81–106, 1986.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Irina Rish. An empirical study of the naive bayes classifier. 3(22):41–46, 2001.

Thorvald Sørensen. 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:pp. 1–34, 1948.

Streaming APIs. Twitter Developer Documentation. `https://dev.twitter.com/streaming/overview`. 2017.

"tweet". *https://www.merriam-webster.com*. Merriam-Webster, 2017.

Lu Wang, Whitney Kisling, and Eric Lam. Fake post erasing 136 billion shows markets need humans, April 2013. URL `https://www.bloomberg.com/news/articles/2013-04-23/fake-report-erasing-136-billion-shows-market-s-fragility/`. [Online; posted 23-April-2013].

Appendices

A Training Duration per Algorithm

Naive Bayes :	N-Grams	Bag of Words	Descriptive
Bernoulli	19,52 (2,11)	21,72 (4,94)	0,28 (0,19)
Multinomial	18,55 (1,35)	21,44 (5,6)	0,15 (0,10)
Regular	67,86 (5,03)	79,03 (20,13)	0,33 (0,25)
Support Vector Machines :			
Linear Kernel	32,08 (3,3)	36,37 (4,98)	12,79 (12,89)
Polynomial Kernel	32,89 (5,62)	43,27 (5,76)	265,61 (262,56)
Radial Basis Function Kernel	42,48 (4,03)	46,83 (5,73)	1,90 (1,62)
Sigmoid Kernel	29,40 (4,65)	31,30 (3,37)	1,42 (1,20)
Artificial Neural Networks :			
MLP	44,64 (6,52)	50,71 (5,38)	0,51 (0,36)
SGD	20,48 (4,04)	21,93 (3,63)	0,15 (0,10)
Ensemble Methods :			
Adaptive Boosting	26,42 (5,95)	25,71 (2,55)	7,14 (4,69)
Random Forest	105,85 (2,68)	106,52 (3,04)	253,91 (159,40)
Logistic Regression	18,60 (1,5)	21,62 (4,71)	0,35 (0,28)
Vote Classifier	732,14 (751,18)	675,82 (531,47)	310,82 (600,11)

* Durations with largest weighted dataset (~2800 training samples) and 5k features, where applicable

** Number in brackets is standard deviation

B Stop Words

The following word list is built into the Natural Language Toolkit (NLTK) Corpora. It represents the list of Stop-Words for the English language as of the 25th of September, 2017. The list can be directly downloaded from the following URL: `https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/packages/corpora/stopwords.zip`.

a	between	hadn	its	once	that	we
about	both	has	itself	only	the	were
above	but	hasn	just	or	their	weren
after	by	have	ll	other	theirs	what
again	can	haven	m	our	them	when
against	couldn	having	ma	ours	themselves	where
ain	d	he	me	ourselves	then	which
all	did	her	mightn	out	there	while
am	didn	here	more	over	these	who
an	do	hers	most	own	they	whom
and	does	herself	mustn	re	this	why
any	doesn	him	my	s	those	will
are	doing	himself	myself	same	through	with
aren	don	his	needn	shan	to	won
as	down	how	no	she	too	wouldn
at	during	i	nor	should	under	y
be	each	if	not	shouldn	until	you
because	few	in	now	so	up	your
been	for	into	o	some	ve	yours
before	from	is	of	such	very	yourself
being	further	isn	off	t	was	yourselves
below	had	it	on	than	wasn	

C Descriptive Features General Statistics

Feature	Value Type	Mean (Mode)	Standard Deviation	Minimal Value	Median	Maximal Value
NUM_STOP_WORDS	Integer	5	2.9	0	5	20
DUPLICATE	Boolean	FALSE				
LEN_CHARACTERS	Integer	116	26,2	6	125	160
NUM_WORDS	Integer	9	3,1	0	9	21
HAS_QUESTION_MARK	Boolean	FALSE				
HAS_EXCLAMATION_MARK	Boolean	FALSE				
HAS_MULTI_QUEST_EXCLAM	Boolean	FALSE				
EMOJI_SENT	Boolean	TRUE				
EMOJI_SENT_SCORE	Float	0,007	0,1	0	0,1	0,7
HAS_PRONOUN	Boolean	FALSE				
COUNT_UPPER	Float	0,12	0,1	0	0	0,9
HAS_HASHTAG	Boolean	TRUE				
RETWEET_COUNT	Integer	77	1013,5	0	0	48.601
URLS_WIKI	Boolean	TRUE				
URLS_MOZ	Boolean	FALSE				
SENTIMENT	Boolean	0,4	0,5	0	1	1
SENTIMENT_CONF	Float	0,92	0,1	0,6	1	1
REG_AGE	Integer	1.762,3	1.013,5	80	1.732	4.091
STATUS_CNT	Integer	127.956,6	315.018,9	1	21.970	2.871.995
FOLLOWERS_CNT	Integer	111.084,7	1.562.658,6	0	1.888,5	101.849.293
FRIENDS_CNT	Integer	8.473.8	26.517,1	0	1.104	521.442
VERIFIED	Boolean	FALSE				
HAS_DESC	Boolean	TRUE				
HAS_URL	Boolean	TRUE				
MSG_P_DAY	Float	88,2	226,8	0,001	14,7	2172,1

* Variable Types: Integer $\in \mathbb{Z}$, Float $\in \mathbb{R}$, Boolean $\in$ {TRUE; FALSE}