Date of acceptance Grade

Instructor

# Does Sentiment Analysis of Social Media Contradict the Efficient Market Hypothesis?

Niclas Joswig

Helsinki April 3, 2019

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Laitos — Institution — Department |
| --- | --- |
| Faculty of Science | Department of Computer Science |

Tekijä — Författare — Author
Niclas Joswig

Työn nimi — Arbetets titel — Title

Does Sentiment Analysis of Social Media Contradict the Efficient Market Hypothesis?

Oppiaine — Läroämne — Subject
Computer Science

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidoantal — Number of pages |
| --- | --- | --- |
| | April 3, 2019 | 23 pages + 0 appendices |

Tiivistelmä — Referat — Abstract

Stock market prediction is and was always a hot topic in any regard: Technical Analysis, Fundamental Analysis and many more approaches developed during the years of active financial markets. Now, that data is the new 'gold' of our century, and computing resources increased exponentially, it can only be reasoned to try and predict financial markets with algorithms and artificial intelligence. Even though research in this area has been going on for quite some time, the introduction of more complex models based on recurrent neural networks implied a huge step forward for time series prediction. In this work i will try to evaluate weather new structures of recurrent neural networks can outperform the human and ultimately beat the market on a constant basis.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

# Contents

# 1 Introduction and Goalsetting

Stock market prediction is already an old and broad area of research since existence of financial markets and stock exchanges over 600 years ago. Millions of people tried their luck, tried to get the big ticket to financial freedom, many thought they can beat the market, be cleverer than the market. But in fact most failed. At this moment only 1% of all private traders are profitable, meaning that they earn more than they lose. An even smaller number of those actually reach a high enough percentage gain to be able to make a living from trading income. All this numbers elucidate the difficulty of beating the market and extracting money from it on a consistent basis. Traders evolved a magnitude of ways or more specific 'systems' over the years which are supposed to be profitable long term. Most of them fall into one of two categories: Technical Analysis or Fundamental Analysis. Former technique is all revolved around price charts and the history of charts, analyzing trends and patterns therein. The fundamental analysis on the other hand engages with all kind of environmental facts concerning the markets. To those belong economical data, political data and other related facts that somewhat relate to the market objective.

The idea of outsourcing the prediction of future price movements to machines and algorithms is not novel. In fact, researchers already try to fit models and algorithms onto this problem for a long time. Examples for those researches on *Support Vector Machines* and *Naive Bayes* are [Mba18, Wan14]. In very recent years with advances in computing power and the rise of neural networks a new model for time-series prediction arose: *Recurrent Neural Networks*. They seem to be the most promising approach for such time-series problems, so why don't we use them for price prediction? To feed the past price data as time series into such a network seems like the logical solution, which already got carried out by various researchers [RPV17, NPdO17, SVG+17]. While the analysis of past price data with machine learning fits into the category of technical analysis, in this work i am going to analyze two approaches from these two articles 'Predict Effect of Trump's Tweets on Stock Price Milestone' [YY17] and 'Stock Market Prediction Using Neural Networks through News on Online Social Networks' [Liu18], which aim at predicting stock prices from social media data. The idea behind that approach is the undeniable fact that economic states and especially the sentiment of people towards economy is reflected in their social media activity. Therefore, the researchers try to extract sentiments from social media posts through word embedding and recurrent networks, hoping to find an accurate model to predict future price. Since we argue about eco-

nomical factors and feelings about the economic state, this way of predicting tries to implement the technique of fundamental analysis with artificial intelligence.

In this work i will start with the basic understanding of financial markets and its possibility to be predicted. Then, an in-depth introduction to neural network is presented, followed by the network architecture *Word-2-Vec* for generation of word embeddings and the advanced *Recurrent Networks* for sentiment analysis. Finally it will be evaluated how models proposed by the authors perform in the markets.

# 2 Application markets

In this chapter i am going through all the mandatory knowledge to understand the basics if the stock markets and its price building. Then, the Efficient Market Hypothesis is introduced as key thesis which this research tries to prove wrong.

## 2.1 Market Basis

Before taking a look at advanced market theories and possible impact of social media on the market, the question of 'How does a market even work and what is price?' needs to be answered. Firstly, a market is defined as an area where people (trader) can exchange one commodity for a commodity of another trader. When a trader enters a market he makes an offer to the market, where he states his wanted commodity and his price or exchange commodity. For example one trader comes into the market and says that he wants to buy a barrel of oil. When now a second trader enters the market and wants to sell a barrel of oil, the price is negotiated and agreed on so that a trade can happen.

To explain the price building in public markets based on the principle of offer and request, we look at the oil market where we trade oil for Euro. Therefore, the price is the amount of euros to buy one barrel of oil. Further, it is assumed that no news or events from outside, like for example the *Brexit* for the Euro or the discovery of new oil resources in case of oil, affect the price.

If now the situation appears that one buyer and two seller for oil are in the market, they will negotiate over the price. Because of the higher amount of sellers in the market, the price is likely to be lower than in a situation with one buyer and one seller. Reasons for that lie in the basic *request and offer* theory: if offer exceeds request the price falls, if request exceeds offer the price rises.

An example of a real market situation: Starting with a price $x$ we assume that an equal amount of buyers and sellers are trading in the market and agree on the current price $x$. If suddenly more buyers enter, the market competition of buyers increases, thus, the sellers can rise the price until the balance between buyers and sellers is equal again (the buyers who don't want to pay the higher price drop out) and the price can be agreed on. Summarizing the price building process, the price is the point where buyers and sellers are in balance and agree on a trade under this price conditions.

## 2.2 Efficient Market Hypothesis

With the knowledge on the process of price building from the previous chapter it will be now looked on the predictability or randomness of exactly this process. The aim of prediction models in the area of stock market and finance products is to forecast the future market movements and, therefore, get ahead of the market in terms of profitability. If one is able to correctly predict market movements, one can simply schedule his buying- or selling-orders for the current financial asset according to the prediction and profit from the market. That a forecast of the market is possible is already a not yet solved discussion since the opening of the very first stock market in the world. Several years ago, this discussion was held to answer the question if a human can predict the market and, thus, be a trader as a full-time reliable job. This discussion can be summarized as the question, whether the so called *Efficient Market Hypothesis* is true or false. This hypothesis states that the current price of an asset reflects all information, which is available at that specific point in time. That assumption implies that it is not possible to obtain an asset that is *undervalued* or *overvalued* because the price perfectly reflects all facts that are available and could influence the price. For the hypothesis to apply, certain factor about the market and its traders must be satisfied. One requirement is that the market participants on average even out their behavior in terms of buying and selling. As long as the traders behavior can be described as normal distribution, too high bids and too low bids for the same product will even out and the price will be in balance all the time.

In previous times, when talking about informations, it was mostly economic data of a certain company or a country or past price data. To beat the market with machine learning based on past stock data is a still ongoing research area as an application area for time-series based solutions. In this work work it will be evaluated weather the Efficient Market Hypothesis can be verified or discarded by a newer kind of

information: *Social Media Posts*.

## 2.3 Social Media

To find a relationship between social media and stock market prices, it first needs to be asked, what is the actual information which we want to extract from social media posts. Social media posts are known to give insides into the overall feeling of individuals and whole societies, shown for example in [AgMK+18] or [SS17], where infectious diseases get predicted through social media. This leads us to the question: What factors influences the choices of investors and traders in the stock market and how do they transform this feelings into text in form of social media entries?

Due to the relation of stock prices to the economic situation, most researches try to extract the overall opinion on the current economic situation from social media posts. This means that it is needed to perform text analysis in a way that posts regarding economic factors such as companies or governments need to be scanned for basic human emotions and attitudes towards those. A classic example for this analysis is described in article [MV17], where Donald Trumps tweets get investigated on weather they influence the US Dollar/Mexican Peso exchange rate. One of the approaches of this work (article [YY17]) is also based on Trump's tweets. Therein, however, the predicted market is the American Stock Index S&P 500 instead of the national currencies. In the second article [Liu18] the biggest Chinese social network *Sina Weibo* is used to collect data, due to the fact that twitter is forbidden in China.

# 3 Neural Network Architectures

In this chapter the technical foundations are lied out in order to understand the underlying processes and to finally reason about techniques and future advancements. In the first step the foundation of neural networks is described detailedly. Then, the technique to generate sentiments from plain text *Word-2-Vec* based on a three layer neural network is in depth analyzed. Finally we will advance to the more advanced network techniques, *Recurrent Network* and a special implementation thereof: *Long-Short-Term-Memory Cells*.

## 3.1 Feed Forward Networks

Before getting to the more advanced *Recurrent Neural Network*, the basic concept and structure of neural networks have to be explained. The classical model for neural networks is the Multi Layer Perceptron [Del05], shown in figure 1.

It consists of so called *Layers* build of a variable amount of neurons. The basic structure of any MLP always involves one *input-layer*, at least one *hidden-layer* and exactly one *output-layer*. Every layer can be noted down as a function, which receives the output of its previous layer as input and calculates the input for the following layer.

In the following part, we assume that the input $\mathbf{x}$ and the output $\mathbf{y}$ of the network are vectors instead of matrices based on the equivalence of matrices and vectors. The function of any layer can be displayed with the input-vector $\mathbf{x} \in \mathbb{R}^m$ and output-vector $\mathbf{y} \in \mathbb{R}^n$ as follows:

$$f(\mathbf{x}) = \mathbf{y} \tag{1}$$

.

The output-vector $\mathbf{y}$ has exactly as many dimensions $m$, as the layer has neurons. This property goes back to the fact that the function, which characterizes the layer, consists of all individual functions of every single neuron. Each neuron is connected to all inputs (*Fully Connected*), so that every neuron is characterized by a function with input $\mathbf{x} \in \mathbb{R}^m$ and exactly one scalar $y$.

For the understanding of the learning process of a MLP, it is mandatory to understand the functioning of a single neuron [Roj96] which depicted in figure 2:

A neuron receives a variable but static number $n$ of inputs, has a weight-vector $\mathbf{w}^T \in \mathbb{R}^n$, a body in which the weighted inputs get summed and finally an activation function $f$ which will be applied to the sum. In accordance to the previous mentioned function for layers in equation 1 the function of one single neuron can be written as follows:

$$f(x) = y \tag{2}$$

The weights are the central point inside each neuron where the learning process takes place and the learned information is stored. Those weights will be adjusted during training process in order to extract knowledge out of the data. The applied activation

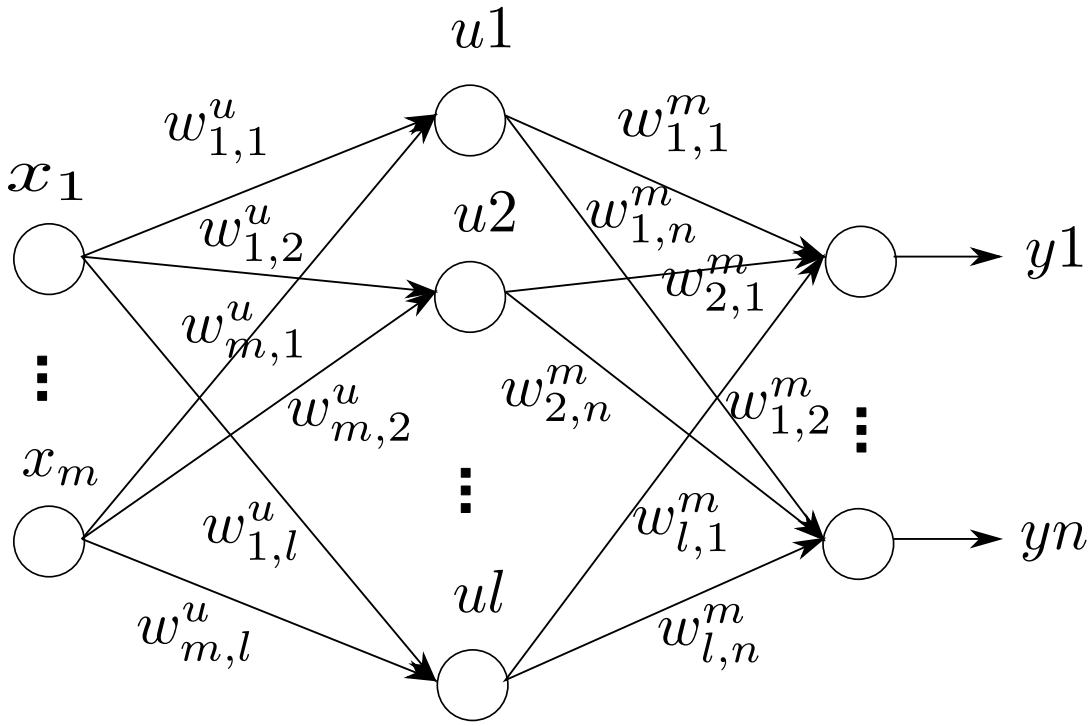



Figure 1: Model eines Multi Layer Perceptrons

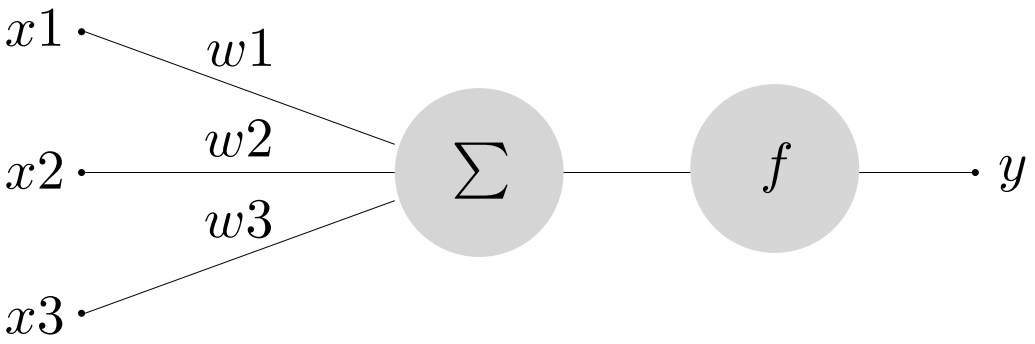
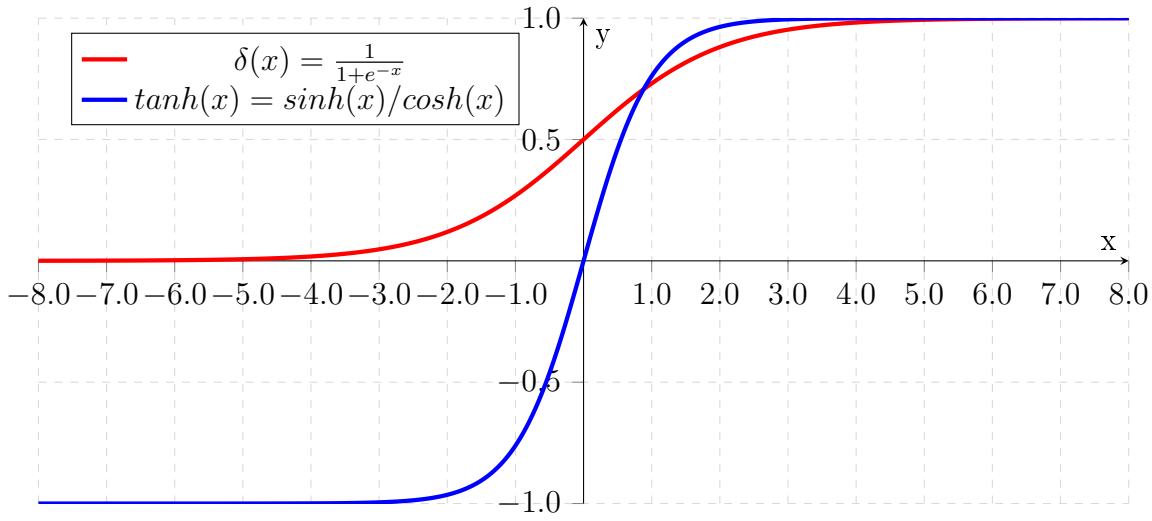



Figure 2: Struture of a basic neuron

function plays a key role in how a layer functions and how the learning process works. The most used functions in recurrent networks are the *logistic function* [McAM93] and the *tanh* function defined in equation 3:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tanh(x) = \sin x / \cosh x \tag{3}$$

As it can be seen in plot —, the logistic function satisfied the need for a mapping into the area between 0 and 1 and the *tanh* function maps onto the interval between $-1$ and 1. This two functions enable networks to model the on/off characteristics of a natural human neuron with the logistic functions as well as a three-way split with the *tanh* function which enables evaluations like 'stock rising/falling/staying same'.




After taking a look at the functioning of an individual neuron, it is time to zoom back out to the MLP again. When initializing the network every neuron's weights are initialized at random. Then, pairs of $(x, y')$ with $x$ being the input- and $y'$ being the target output-vector are feed into the model. The chain wise calculation throughout the layers is called *Feed Forward* and produces the output-vector $y$, shown for one layer in equation 4:

$$f(\mathbf{w}^T \mathbf{x}) = y. \tag{4}$$

The central goal is to improve the model's performance. Thus, it requires a measurement for the quality of the models predictions. Based on true label $y'$ and prediction $y$ the error function *Mean Square Error (MSE)* [Roj96] is defined as follows:

$$SSE : f(\mathbf{x}) = \frac{1}{2}(y' - f(\mathbf{w}^T\mathbf{x}))^2 = \frac{1}{2}(y' - y)^2 \tag{5}$$

The calculation of MSE puts out an adequate measure on how good the model is, trivially the goal of the learning process is to reduce the error as much as possible. This reduction can be achieved by using the *Backpropagation-Algorithm* [RN16, LBH15, Wer94] based on the error-gradient. Deriving the loss-function with the *Delta-Rule* will not be further explained here, it will be assumed that the gradient $\Delta$ is calculated so that it can be plugged into the weight-update shown in formula 6:

$$w_i^{t+1} = w_i^t - \Delta w_i \tag{6}$$

Training itself is then an iterative process where the weight update and therein the gradient gets calculated and applied for every single weight in the network, until the loss function converges.

## 3.2 Word-2-Vec

Sentiment building is the essential key to performing any kind of natural language processing task with machine learning techniques. Since the words are not usable for machine learning techniques like numbers are, it is mandatory to transform words into vectors. A *One-Hot* Vector is a vector of any dimension $m$ with the property that exactly one scalar is 1 and all others are 0. This kind of vector leads to a trivial approach, which would be to one hot encode all words in the text and feed it into the classifier. Even though this is a possible approach, it does not fulfill the property that close words or different grammatical forms of the same words end close to each other in the vector space. Without this property the understanding of the machine would be much worse because it would need to make sense of a sheer unlimited amount of distinct words with no implicated meaning or relationship.

The technique *Word-2-Vec* [MSC+14] originated exactly this idea that motivates methods for sentiment building: The need of transformation of text data into a vector space. Therein, the aim is to design the vector space in a way that close related words like 'dog' and 'cat' are close to another inside the vector space. The key idea behind it is the use of a standard feed-forward neural network, which is trained on a group of words as input and produces the embedding vectors, which satisfy the

above stated requirement, as output. When using neural networks, a technique in the subfield of *Supervised-Learning*, one might ask how applying neural networks to this problem of vector building is possible when we obtain any knowledge about how the result-vectors (the labels) should look alike. This challenge can be overcome by using the context of the word as label for supervised learning. Reasoning behind that thought is the fact that related words like 'cat' and 'dog' often appear and the same context, e.g. 'i love my cat' and 'i love my dog'. Transforming words into the vector space like above specified also makes it possible to perform mathematical operations on words, e.g. the difference between the vectors for France and Paris will be approximately the same as for Germany and Berlin.

The network architecture of the Word-2-Vector network is depicted in figure 3. One can see that the network consists of three layers, the input layer, one hidden layer and the output layer. The first thing one might observe is the fact that input layer and output layer have the same amount of neurons. This property results from the given fact that before words can be trained by the network, they have to be converted into one-hot vectors. This happens in a way, that every distinct word has its own one-hot encoding. Therefore the input word as well as the output word (the context) are represented as a one-hot vector of size $v$, also called the vocabulary size.

Based on the outer conditions, that we need to map a one-hot vector representing one word to another one-hot vector representing the context word, we can now focus on the logic inside the hidden layer, where the mandatory vector transformation happens. Let us use the example of the two words 'cat' and 'dog' as input words and 'love' as context word. The central idea is that the output for both training pairs ('cat','love') and ('dog','love') have the same output vector ('love'). To map both inputs ('cat' and 'dog') onto the same output, the output of the hidden layer needs to be the same for both words, because it both will get multiplied with the same weight matrix of the output layer. Now that it is known that the output of the hidden layer has to be the same, the network will train the weight matrix of the hidden layer in a way that it maps the vectors for 'cat' and 'dog' to approximately the same vector.

The size of the hidden layer, in figure 3 denoted as $b$, is called the *Embedding Dimension*. The embedding dimension is a hyper parameter which defines the vector dimension into which each word gets transformed. The behavior of higher and lower values as embedding dimensions is the same as with most machine learning

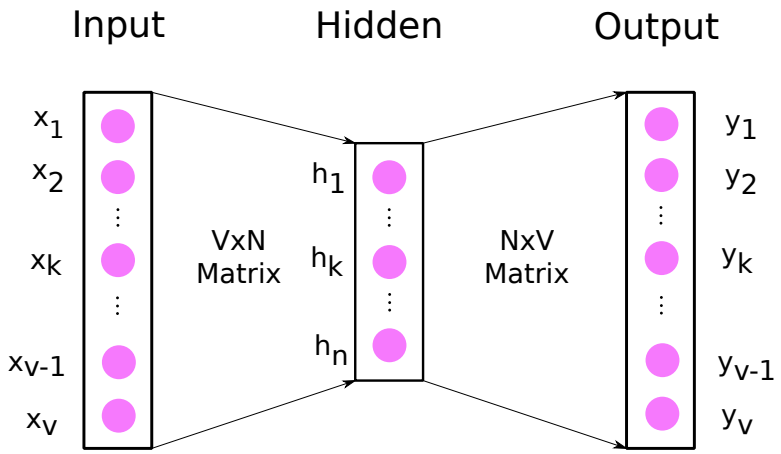



Figure 3: Structure of a Word-2-Vec-Network

parameters: the bigger the embedding dimension, the bigger the vector space and, therefore, a more detailed vector space can be created with every vector representing more information. At the same time, when it is chosen too high, the performance will start stagnating or at worst decreasing, because the problem to solve does not make use of such a broad vector dimension. Thus, tuning of this hyper parameter is an essential part of every sentiment project and the results will be explained in chapter 4.2.

After training this network with all available text data, the weight matrix of the hidden layer fulfills the previously stated properties, which the transformation from words to vectors requires. If we want to transform the text into its embeddings, the words need to be transformed into the distinct one-hot vectors and thereafter have to be multiplied with the weight matrix to receive an embedding of the word such that it can be used as input for further network layers.

## 3.3 Recurrent Networks

As it can be seen in figure 1 of the MLP, the network receives an input at time $t$ and returns an output just based on the input at time $t$. Therefore, the model has no prior knowledge about older states or inputs and can only evaluate based on the current information at time step $t$. In this research the data to process is social media entries or more concrete sentences, which are basically a time series of words,

or a time series of past stock price data.

To extend neural networks to this application field *Reccurent Neural Networks (RNN)* got introduced [Jor86, Pea89]. The key idea thereof is to expand the neurons with a connection to themselves, which is called *Hidden State* or *Cell State*. When now a neuron is feed an input, it will produce an output just like before, but as addition it will update its own state by using the edge to itself.

Figure 4 shows the basic structure of a simple recurrent neuron with its edge to itself.

For illustration it is possible to imagine a recurrent neuron as a chain of the same neuron, as it is shown in figure 5, which receives the input of time $t$ as well as the output from previous time step $t-1$.

Mathematically, the simple recurrent cells calculate their hidden states with the following function:

$$h_t = f(W * x_t + U * h_{t-1}) \tag{7}$$

Inside equation 7 $h_t$ stands for the hidden state matrix of the neuron at time $t$, $x_t$ is the input at time $t$ and $W$ as well as $U$ are two distinct weight matrices for both separate inputs. Compared to the basic neuron a new weight-matrix $U$ for weighting the cell state appears in the formula. This matrix gets trained similar to the matrix $W$ in backpropagation to utilize the previous cell state for a minimization of error.

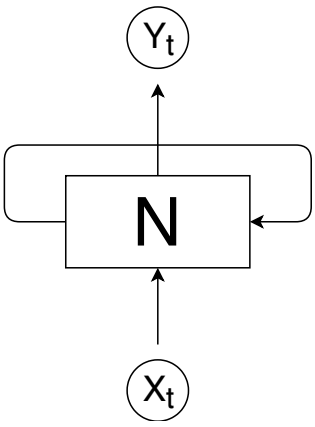



Figure 4: Basic Structure of an Recurrent Neuron

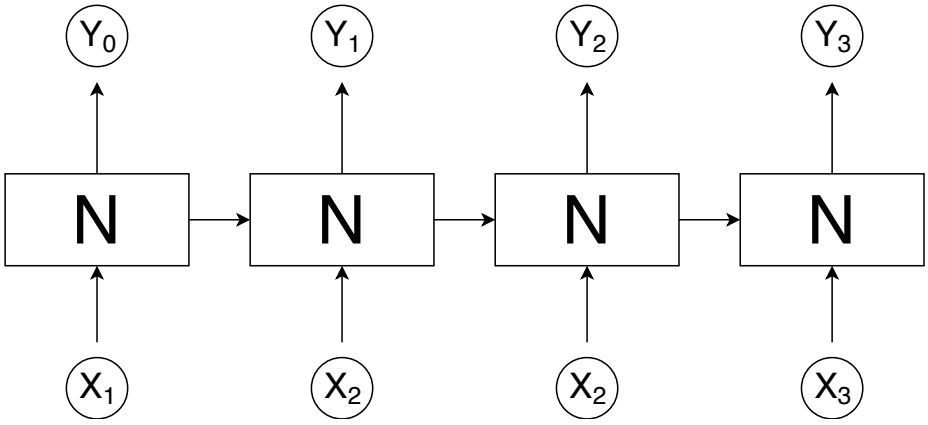



Figure 5: Chain-Representation of a Recurrent Neuron

## 3.4 Long-Short-Term-Memory Cell

The previously described RNN cell suffers from the issue of gradient vanishing, a phenomena that occurs when information 'vanishes' over a long iteration of time steps due to the smaller getting gradient in each time step. A more in depth explanation o the vanishing gradient problem can be found here [Hoc, PMB].

The need for a more complex neuron which is able to store information for longer periods of time led researchers to the development of the *Long-Short-Term-Memory Cell (LSTM)*.

The LSTM-Cell is depicted in figure 6. One can already see that the structure is more complex than the basic recurrent neuron.

Before going over all the gates in detail, the first step is the concatenation of the

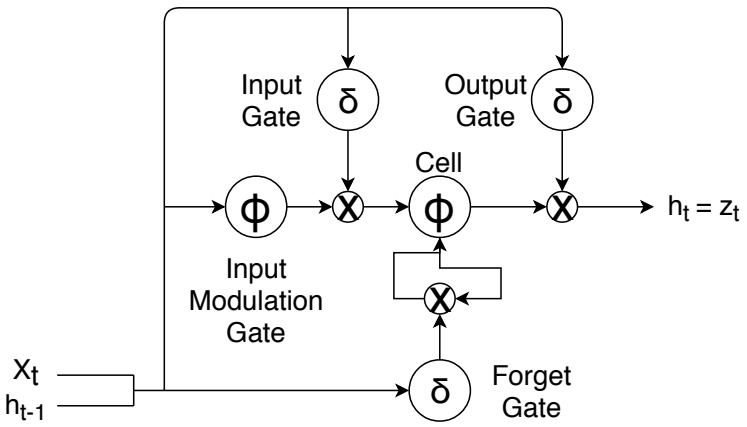



Figure 6: Architecture of a LSTM-Cell

input $x_t$ and the previous time step output $h_{t-1}$ to one matrix.

The nodes which are denoted as 'Gate' in figure 6 are all by themselves an own neural network layer. Every distinct gate has his own weight matrix $W$ and an activation function. Starting with the *Forget Gate*, the concatenated input is multiplied with the weight matrix $W$ of the forget gate and afterwards the sigmoid function gets applied to it. Subsequently the forget gate output gets multiplied with the cell state $c_{t-1}$ and result is written into the cell as new state. For our application area of sentiment analysis, a possible scenario would be that the person which is talked about in the text changes and, therefore, the gender which could be a part of the cell state might need to be reset. Sigmoid function is therein used to map the information to keep onto a 1 and the ones to forget to a 0. Mathematically the function of the forget gate can be modeled as follows with $f_t$ denoting the forget gate-output at time step $t$, $W_f$ the weight matrix of the forget gate and $[h_{t-1}, x_t]$ the concatenation of both inputs:

$$f_t = \delta(W_f * [h_{t-1}, x_t]) \tag{8}$$

The input and input-modulation gate are constructed in a similar way as stand-alone neural networks. The input gate works exactly as the forget gate with a logistic function while the input modulation functions applies a tanh function. Those two gates combined result in the values that should be changed and how much they should change. In terms of formula the input gate is defined as follows:

$$I_t = \delta(W_I * [h_{t-1}, x_t]) \tag{9}$$

and the input-modulation gate:

$$Im_t = \delta(W_{Im} * [h_{t-1}, x_t]). \tag{10}$$

Those three described gates form the new cell state $C_t$:

$$C_t = f_t * C_{t-1} + I_t * Im_t \tag{11}$$

The last gate of the LSTM cell is the *Output Gate*. It is used to direct the focus of the output to a special area of the information available in the cell state. It works just as the input gate works on the input: it filters and lets through information with zeros and ones at specific positions. The output gate can be computed with the following formula:

$$O_t = \delta(W_o[h_{t-1}, x_t]) \tag{12}$$

That was the last single component to calculate, the final output can be calculated as follows:

$$h_t = O_t * tanh(C_t) \tag{13}$$

Overall, this architecture is one of the most advanced technologies in the field of recurrent networks because it has so many possibilities to keep track of certain areas of data as well as distinct between information to store inside the cell for the future and to put out at the current time step. While it offers this broad range of possibilities, the downside lies in computational costs: Having 4 weight matrices for the 4 gates trivially requires four times the computation effort to train the network. Nevertheless, to keep LSTM network practical, companies like for example *Nvidia* developed a highly optimized LSTM implementation in their *CuDNN* library speeding up training (see [1])

# 4 Evaluation of Approaches

In this chapter the approaches of the articles 'Predict Effect of Trump's Tweets on Stock Price Milestone' [YY17] and 'Stock Market Prediction Using Neural Networks through News on Online Social Networks' by Tong Yang and Yuxin Yang are analyzed deeply. First, the environment for the experiment including dataset and sentiment building will be described. Then the chapter proceeds with the individual structures of the proposed recurrent networks and ends with the evaluation of results the authors achieved in their researches.

## 4.1 Experimental Structure

**Article [YY17]** Article [YY17] revolves as the title already stresses around finding correlations between Tweets of the current US president Donald Trump and fluctuating stock prices. In their experiment the authors feed their preprocessed data into a word-2-vec network. As weight basis for their word-2-vec network they import the *GlOve* [PSM14] weights to acquire already pretrained a basic vocabulary

[1] https://developer.nvidia.com/cudnn

word mapping. In the trump-dataset most tweets are less than 20 words, therefore, the authors choose the input length of their LSTM-network inputs to be 30 words. In all tweets with less then 30 words the missing part of the input gets filled by the GlOve encoding for a missing word. The in chapter 3 described parameter of the embedding-dimension is chosen to be in an interval of $[50, 100, 200, 300]$. In further model analysis it will be evaluated which values performed best and which performed worst. This article also investigates the performance difference of recurrent networks compared to other statistical models like *Support Vector Machines* and *Naive Bayes*. For this research we step away from this comparison and just note that recurrent networks outperform the other models consistently.

**Article [Liu18]** Article [Liu18] goes a step further and uses both past stock prices as well as social media posts as inputs for their recurrent network. In comparison to the previous article, this one is focused on the Chinese markets as well as Chinese social media instead of the US. Specifically the authors use a dataset of news from the biggest Chinese social network *Sina Weibo* and as stock data they use the stock index *China Shanghai Shenzen 300 Stock Index*.

The first thing to notice is that the authors here use a *Gated Recurrent Unit* [CGCB14]instead of LSTM neurons. The difference between those two types of recurrent neurons is that the GRU has no output gate, thus, it outputs the cell state straight away. Overall the difference between performances thereof is not big enough to be considered relevant for this work.

The authors here choose a different process of transforming the words into a vector space. Instead of using word-2-vec, they use a probabilistic approach based on counting words in positive and negative news for the stock market. First step in this process is the separation of all social media entries into two groups: posts who made stocks go up and posts who made stocks go down. Then, the abundances of each word in both classes is counted and the posterior probabilities of each word occurring under the requirement of belonging to each class get calculated. If we want to transform a new post now, the probability of falling into one class is calculated by multiplying the class-probabilities of every word the post contains. Last step is to measure the degree of membership to one class by normalizing it with added probabilities of both classes. Finally, all the calculations channel into one value, which denotes the degree of impact a post had in the past and therefore should also have in the future.

## 4.2 Structure of LSTM models

**Article [YY17]** The first aspect to state about the article is the missing information about the actual architecture of the network they used as a whole. The authors only focused on the embedding as well as LSTM layer, no information is given about any possible number of layers, further layer types or amount of neurons in further layers. The authors experimented with all parameters in certain intervals and concluded what achieved the highest accuracy among all the parameters. The first dimension of the input is the embedding dimension which got evaluated in an interval of $50, 100, 200, 300$. A high embedding dimension can on one hand display more relations in its broader vector space, but at the same time, especially with too few training data, can start to perform worse due to a too complex vector space for too few information. The second dimension is the word- or sequence-length of the tweets. The authors tested this parameter with values of 20 and 30 without cutting any stop-words like 'the' or 'a'. Finally the last parameter is the number of LSTM neurons in the first layer of the network. The more neurons, the complexer the relationship in the data the network can model, until the complexity is too high and the model just learns the train data 'by heart' which is called *Overfitting*. The amount of neurons got evaluated out of the interval $[5, 32, 64]$.

**Article [Liu18]** To exactly display the effect of the new's influence factor on the prediction outcome the authors conduct experiments with three distinct models: In the first try the prediction is made solely on the daily closing stock price of the last ten days. For the second approach the input is broaden with more price feature like closing, opening, lowest, highest-price, price change, volume (amount of commodities traded at this day) and more. Finally, the third approach also contains additionally the news influence factor in the input. To measure the outcomes with different inputs offers the opportunity to distinctly judge the influence of every factor from the set of inputs.

The network structure itself consists of two GRU layers, more details on the amount of neurons for example is not given. For the training-process 530 trading days of data are available. To validate the model on unseen validation data the authors used cross-validation and divided the set into the first 100 days as training set and the remaining 430 days as validation set. The performance on both sets is measured by three different loss functions: *Root Mean Squared Error (RMSE)*, *Mean Absolute Error (MAE)* [CD14, WM05] and *Mean Absolute Percentage Error (MAPE)*

[MGGR16].

## 4.3 Performance Evaluation

**Article [YY17]** The authors of article [YY17] extract three major learnings from their experiments: The first outcome is that an embedding dimension of 200 outperforms an embedding dimension of 50 significantly, but at the same time an embedding dimension of 300 does not increase the performance further compared to 200 embedding dimensions. This fact is due to the size of the vector space: If we want to map a huge vocabulary onto a small feature space, like it is the case with 50 embedding dimensions, not all relations between words can be modeled as exact as in a bigger feature space, where it is possible to model word distances very exact. The stagnation in performance while increasing the embedding dimension over 200 is simply reasoned to enough possibilities in a feature space with 200 embedding dimensions. All information can be modeled in 200 dimensions, thus, 300 dimensions don't increase the performance anymore.

A second more trivial finding is that a tweet length of 30 words outperforms a tweet length of 20 words. If information is cut out of a tweet longer than 20 words to truncate the sentence, information is lost which the model can not learn from. Therefore, always the maximum amount of text should be used in order to provide the network with as many informations as possible.

Regarding the last parameter, the amount of LSTM neurons, the performance reached its peak when using a small number of 5 neurons, showing in a low in the loss function. As previously stated, a higher complexity model with more neurons is prone to over fitting when the task complexity is not accordingly high. This phenomena got recognized by the authors in their experiments, visible in the better performance of 5 neurons compared to 32 and 64 neurons.

Combining all the evaluated parameters together, the authors acquired the best model with a tweet length of 30, an embedding dimension of 200 and 5 LSTM cells.

**Article [Liu18]** The results of the three different models evaluated in article [Liu18] are shown in table 1.

Table 1: Results measured with three different methods, taken from [Liu18].

| Method | MAE | MAPE | RMSE |
|---|---|---|---|
| 1 | 0.752236602 | 11.66814942 | 0.953598732 |
| 2 | 0.728867794 | 10.70145408 | 0.915376292 |
| 3 | 0.625429771 | 9.381182145 | 0.803118357 |

The results clearly indicate that news influence stock prices. The results for the third approach, which included the news factor in the input, achieves better performance measured by all three loss-functions.

The predictions of the model based on news are presented in figure 7, where the red line represents the real price and the blue line the predicted price. On the first look at the two price curves, one can see that they are quite close to each other, indicating a good performance of the model overall. But on a closer look it is visible that it mostly follows the trend and in case of reversals in price (places where the trend changes for at least a small amount of time) the prediction of the reversal mostly was not in time, thus, the question arises, weather the model achieves higher accuracy than an often used trend indicator (an indicator is a statistical tool that gives an output if the trend is going up or down based on some calculation on past price data). Since no further measurements in terms of actual trades according to prediction are conducted, a statement about actual profitability can not be made.

# 5 Conclusion

In this work two approaches of predicting future stock prices from social media posts are evaluated. The main difference betweens those two was the fact that one was solely based on social media embeddings as input, while the second one learned relations based on past stock prices in addition to social media entries. The first model, which tries to predict the future stock price of the S&P 500 Stock Index from Trump's tweets, reached a peak of 48% accuracy in an three-class-classification problem (stocks rise, fall or stay the same). The authors state that this is 5% more than human accuracy, but no further reference on how human performance got measured is provided. The overall conclusion is that predicting stock prices just from Trump's tweet is fully possible. The results show that there is definitely some correlation which the model detects, but it is also logical that the market is influenced by a much wider range of news as just Trumps's tweet and in the end, it is still the banks and investors who move the market with their capital in the direction that they want, independent from what Trump is saying. Thus, the model

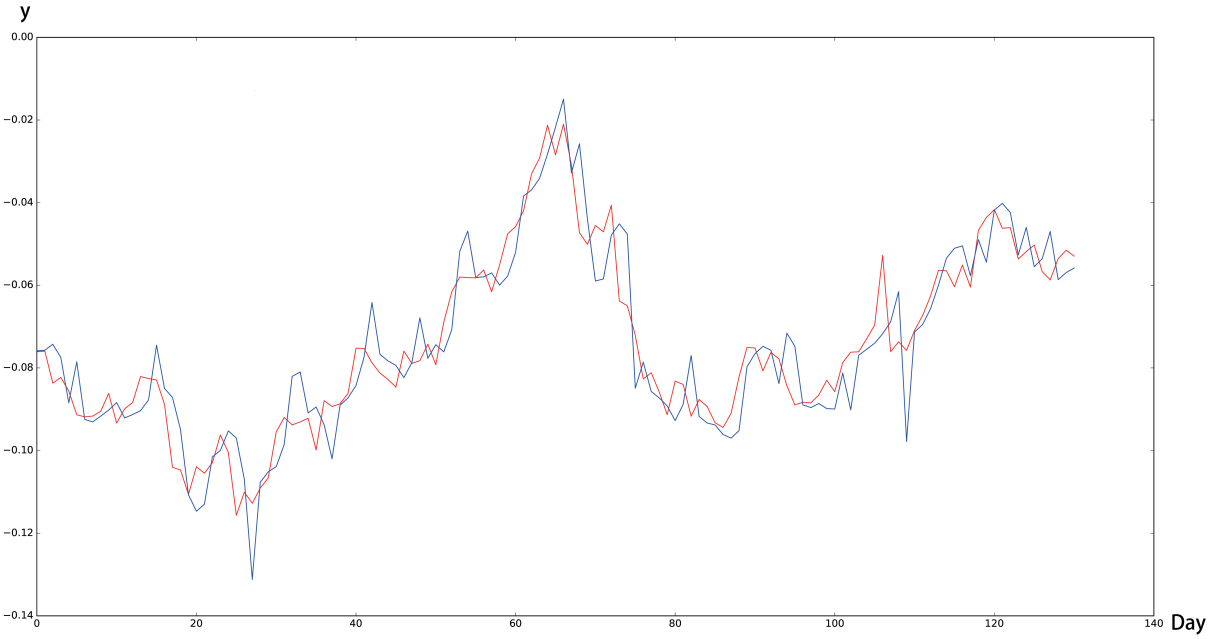



Figure 7: Stock prices over 140 days, the red line is the true price, the blue line the prediction, from [Liu18]

alone is not sufficient to extract any financial gains out of the market.

The same goes for the second model: the price prediction is still too inaccurate and does not predict major price reversals. Thus, extracting financial gains is not very likely, even though to test such a model in reality you would need to define an actual strategy of how to enter the market with which predicted price and when not.

Summarizing, even though the impact of news on future stock prices is proven in this work, the developed models are, despite recent advances in recurrent network technologies, not able to extract financial gains from the market yet. Still, in combination with advanced indicators it could be just one part or one indicator where price might go of a bigger and complexer system. Also the question, if the market is predictable and, thus, the efficient market hypothesis is invalid can only be answered with a clear *no* in this work.

# 6 Future Steps

As it can be seen already in the two described approaches, there is a broad range of methods on how to solve the stock market prediction problem based on social media news. Here we saw one model which was singularly build on trump's tweets,

while another used past stock data in addition to a news influence value as input. A plethora of researches are already using price indicators, like for example a moving average, instead or in addition to raw price data for prediction of price. For the future it would be a promising area of research to try to combine social media entries with different price indicator data. It can definitely be worth to investigate weather correlations exist between indicator events such as an indicated change of trend and major news postet on social media.

From a technical points of view, in the area of deep learning a rapid number of novel approaches or techniques get developed and presented each year. Techniques like *Attention-based Multi-Input LSTM* as it is presented in this article [LSZ18] specifically tackles the challenge of low correlation between inputs and outputs, which is arguably the case for future and past stick prices. The authors therein implement a so called *Attention Layer* into their recurrent network in order to filter the random noise and, therefore, focus on the relevant relations. While social media data is not part of this models features, it could be a huge improvement in the future to advance prediction of those more complex models with social media data.

# References


AgMK+18 Al-garadi, M. A., Mujtaba, G., Khan, M. S., Friday, N. H., Waqas, A. and Murtaza, G., Applications of big social media data analysis: An overview. *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, March 2018, pages 1–5.

Alc17 Alcantara, G., Empirical analysis of non-linear activation functions for deep neural networks in classification tasks. *arXiv:1710.11272v1*.

CD14 Chai, T. and Draxler, R., Root mean square error (rmse) or mean absolute error (mae)? arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7, pages 1247–1250.

CGCB14 Chung, J., Gulcehre, C., Cho, K. and Bengio, Y., Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS*.

Del05 Delashmit, W. H., Recent developments in multilayer perceptron neural networks. *Proceedings of the 7th Annual Memphis Area Engineering and Science Conference, MAESC*, ,pp 1-3.

Ell90 Ellacot, S. W., An analysis of the delta rule. *International Neural Network Conference*.

Hoc Hochreiter, S., The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*.

Jor86 Jordan, M. I., *Distributed Representations of Words and Phrases and their Compositionality*. 1986.

LBH15 LeCun, Y., Bengio, Y. and Hinton, G., Deep learning. *Nature*.

Liu18 Liu, H., Leveraging financial news for stock trend prediction with attention-based recurrent neural network. *arXiv:1811.06173v1*.

LSZ18 Li, H., Shen, Y. and Zhu, Y., Stock price prediction using attention-based multi-input lstm. 2018, pages 454–469.

Mba18 Mbadi, S., Predicting stock market movement using an enhanced naÃ¯ve bayes model for sentiment analysis classification., 02 2018.

McAM93 Mhaskar, H. N. and c. A. Micchelli, How to choose an activation function. *NIPS'93 Proceedings of the 6th International Conference on Neural Information Processing Systems.*

MGGR16 Myttenaere, A. D., Golden, B., Grand, B. L. and Rossi, F., Mean absolute percentage error for regression models. *Neurocomputing.*

MS10 Misra, J. and Saha, I., *Artificial neural networks in hardware: A survey of two decades of progress.* Neurocomputing, 2010.

MSC+14 Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J., Distributed representations of words and phrases and their compositionality, 2014.

MV17 Malaver-Vojvodic, M., Measuring the impact of president donald trumps tweets on the mexican peso/u.s. dollar exchange rate.

NPdO17 Nelson, D., Pereira, A. and de Oliveira, R., Stock market's price movement prediction with lstm neural networks. 05 2017, pages 1419–1426.

Pea89 Pearlmutter, B. A., Learning state space trajectories in recurrent neural networks. volume 2, 1989, pages 365–372.

PMB Pascanu, R., Mikolov, T. and Bengio, Y., On the difficulty of training recurrent neural networks. *arXiv:1211.5063.*

PSM14 Pennington, J., Socher, R. and Manning, C. D., Glove: Global vectors for word representation. *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pages 1532–1543, URL `http://www.aclweb.org/anthology/D14-1162`.

PXP00 Pham, D. L., Xu, C. and Prince, J. L., Current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2,1(2000), pages 315–337.

RN16 Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach.* Pearson Education, 2016.

Roj96 Rojas, R., *Neural Networks - A Systematic Introduction.* Springer, 1996.

RPV17 Roondiwala, M., Patel, H. and Varma, S., Predicting stock prices using lstm. *International Journal of Science and Research (IJSR)*, 6.

SS17 Seo, D.-W. and Shin, S.-Y., Methods using social media and search queries to predict infectious disease outbreaks. *Healthcare Informatics Research*, 23, page 343.

SVG+17 Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. and Soman, K., Stock price prediction using lstm, rnn and cnn-sliding window model. 09 2017, pages 1643–1647.

Wan14 Wang, Y., Stock price direction prediction by directly using prices data: an empirical study on the kospi and hsi. *Int. J. Business Intelligence and Data Mining*, 9.

Wer94 Werbos, P. J., The roots of backpropagation. *Ordered Derivatives to Neural Networks and Political Forecasting.*

WM05 Willmott, C. J. and Matsuura, K., Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30, pages 79–82.

YY17 Yang, T. and Yang, Y., Predict effect of trumps tweets on stock price milestone.