Bachelor Thesis

Computer Science and Engineering 300 hp, Computer Engineer 180 hp

Sentiment Analysis of Nordic Languages

Thesis in Computer Science and Engineering 15hp

Halmstad 2019-06-18

C. Fredrik M. Mårtensson, Jesper A. Holmblad

HALMSTAD UNIVERSITY

Abstract

This thesis explores the possibility of applying sentiment analysis to extract tonality of user reviews on the Nordic languages. Data processing is performed in the form of preprocessing through tokenization and padding. A model is built in a framework called Keras. Models for classification and regression were built using LSTM and GRU architectures. The results showed how the dataset influences the end result and the correlation between observed and predicted values for classification and regression. The project shows that it is possible to implement NLP in the Nordic languages and how limitations in input and performance in hardware affected the result. Some questions that arose during the project consist of methods for improving the dataset and alternative solutions for managing information related to big data and GDPR.

Abstrakt

Denna avhandling undersöker möjligheten att tillämpa sentiment analys för att extrahera tonalitet av användarrecensioner på nordiska språk. Databehandling utförs i form av förprocessering genom tokenisering och padding. En modell är uppbyggd i en ramverkad Keras. Modeller för klassificering och regression byggdes med LSTM och GRU-arkitekturer. Resultaten visade hur datasetet påverkar slutresultatet och korrelationen mellan observerade och förutspådda värden för klassificering och regression. Projektet visar att det är möjligt att implementera NLP på de nordiska språken och hur begränsningar i input och prestanda i hårdvara påverkat resultatet. Några frågor som uppstod under projektet består av metoder för att förbättra datasetet och alternativa lösningar för hantering av information relaterad till stora data och GDPR.

Contents

1 Introduction

1.1 Background

Text analysis is often used to extract information from the growing internet of things (IoT). The more information that can be extracted through IoT, the more robust and efficient solutions are required. There are several cloud services that process and provide solutions to the problem in the larger languages. From the Google Cloud Natural Language API [1], English, French, German, Italian, Japanese and more are languages that are provided by their service. Since the individual Nordic language constitutes such a small proportion of the world's language there are few if any dedicated services, according to Quicksearch. It's possible to translate Nordic into English and then apply state of the art machine learning to obtain sentiment. In theory translating the input has a negative impact on the performance of the service. The translation problem can be caused by several reasons, but mainly the reason can be related to the amount of data needed to translate a language. A translation also in some cases has difficulty in handling misspellings without correcting the error.

A test for Swedish through Google translate, table 1, resulted in the sentence; "hor mår du?", english; "how are you?. The translation became "are you horror?" The result is far from accurate and alternative answers were given when the sentence changed a letter (H) into capital; "Hor mår du?" the translation became; "Do you feel hor?" and again the translation failed. According to an article from 2016, it was demonstrated how a translation of the language reduced the accuracy of the model [1]. This gives room for exploring dedicated solutions for sentiment analysis.

Attempt	Swedish	Expected	Observed
1	hor mår du?	how are you?	are you horror?
2	Hor mår du?	How are you?	Do you feel hor?

Table 1: Re-created test to show the problems of execution of translation, through google translate.

1.2 Aim

The Bachelor thesis aims to achieve the goals of creating a generalised model for machine learning. This includes providing knowledge about how to create

[1] `https://cloud.google.com/natural-language/`

and analyse a model to determine if it is good. The models should be analysed against each other and the development of the best model should be done. The project analyses algorithms to improve the model and minimise over-fitting.

1.3 Problem definition

The specifications consist of the requirements that expect to be solved in the result. The first goal includes analysing the given dataset and carrying out a preprocess if it is considered necessary. Thereafter a machine learning will be implemented to see if it is possible to create a model for the Nordic languages from the data set and present how they perform towards a user's feedback.

Which individual models and algorithms can be used to improve the result in the machine learning? Verification of models will set a standard for what model that will be considered sufficient and what goal the models should strive towards. The test is verified by the table in chapter 4.2. A target value for the project's measurement value is 68%. The value consists of recommendation by Quicksearch and is therefore used as the main goal. In order to get the model out to the market, an implementation must be carried out against the model in order to make it available to a web service. Finally, tests for taking out the topic from the stated reviews should also be carried out to see if information can be categorised for easier processing. This is no primary goal and will be implemented if time is sufficient. The chosen approach can be observed in figure 1.

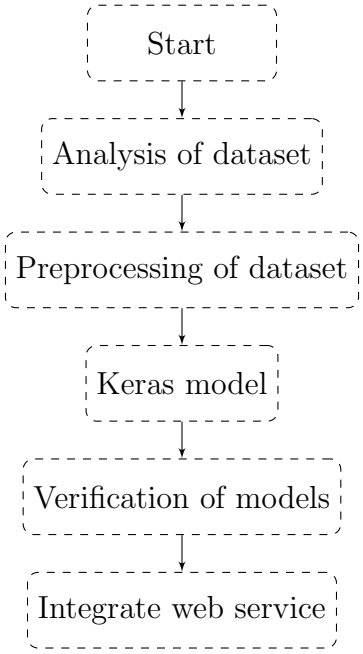


Figure 1: The planned workflow

1.4 Delimitation's

The delimitation's are based on problem definition. This includes that the project should only investigate whether it is possible to implement machine learning on the company's data and present the result. Nothing outside the creation of a machine learning is included. In summary, this consists of two points:

- No graphical user interface (GUI) should be presented in the result as it is not included for creating a machine learning.
- The database is obtained from the company to provide arbitrary information that can be used to create a model.

2 Related work for sentiment analysis

Methods used in machine learning are continuously developed to improve performance and efficiency. To find the sentiment of a text review a word embedding is needed. Previous studies have utilised word2vec, developed by Tomas Mikolov in 2013 at Google. Word2vec uses the shallow neural network that is implemented in both the Skip Gram and Common Bag Of Words (CBOW) [2, 3]. Previous studies show how sentiment analysis can be applied to products, feedback and social networks to find a pattern or predict trends [4, 5]. In a study at *Department of Computer Science, University of Illinois at Chicago* [6]. The objective was to provide a feature-based summary of a large number of customer reviews of a product sold online. A Chinese study uses attention based LSTM to show that a sentence can contain multiple sentiments directed at different topics [7]. Suggesting that topic and sentiment mustn't be extracted with two separate models or else the wrong sentiment could be related to the topic. The articles are related to this project with regard to the application area, what is excluded is created by an embedding or feature vector and classification and how this is applied. In comparison, this project uses regression and a smaller test of classification. There are different layers to use for the model including LSTM and GRU. Both have their respective advantages, but according to an article, GRU is considered better suited for sentiment analysis than LSTM [8].

3 Theories

This section covers theories related to the project. Each theory describes what the topic and what it is used for. It goes in deeper detail for the optimiser, loss function, and the architectures used during the project. The feature vector section explains the basic shape of machine learning input. Correlation analysis is explained since this is the main metric used to evaluate the results. Word embedding section shows classification input can be represented in a hyper dimension in order to find relationships between classes. Some light is shed on the hazards of data skew since this is present in the data used. Two means of regularisation for machine learning is covered. Lastly Natural language processing shows what can and can not be parsed by ordinary machine learning.

3.1 Feature vector

A feature vector is a vector of that commonly contain numeric presentation called features for a machine learning. Features can be extracted from the preprocessing phase with different methods [9]. A feature is an measurable property. It is important to choose informative and independent features for effective algorithms. Feature vectors is widely used in machine learning as: classification, regression and pattern recognition. It is combined with weights through a dot product in order to construct a linear predictor function that in turn is used to determine a result for a prediction.

The vector space that is associated with vectors is often called the feature space. It's is not uncommon to use some form of dimensional redundancy technique [10] to reduce the feature space.

3.2 Correlation analysis

Correlation analysis is a statistical evaluation method to explain a relationship between two numerical and continuous variables. If a deterministic change is discovered between two variables it means that there's a linear relationship between them [11]. Correlation is expressed on a scale from negative one to positive one. A 100% correlation means that $f(x) = g(x)$ and a -100% means $f(x) = -g(x)$. Zero correlation suggest there's no linear relationship between the two vectors. For instance: two perfectly random vectors are excepted to

have zero correlation. The correlation between two vectors X and Y is:

$$corr(X, Y) = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Where σ_X and σ_Y are the standard deviation respectively. μ_X and μ_Y are the mean values. E is the expected value.

3.3 Word Embedding

A non-linear item can be approximated by a N dimensions long vector. Take this vector, normalise it and arrange so that two similar items get similar values on each dimension. A matrix of such vectors is called an embedding. Table 2 shows an example word embedding.

Word	Fruit	Round	Orange
Apple	0.82	0.57	0.08
Orange	0.6	0.54	0.6
Pear	0.96	0.29	0.0
Turtle	0.58	0.58	0.58

Table 2: Example of three dimensional word embedding. Similar words have similar values.

The dimension labels and the values in table 2 are made up for readability, in reality embedding dimensions represent abstract properties. Note that there's no information on turtles since all dimensions have the same value. This indicates that the embedding could use more dimensions.

The main use of embeddings is to enable comparison of items. This is done with cosine similarity but since the vectors are normalised it reduces to dot product. We denote the similarity between two words j and I as $p(w_j|w_I)$.

Creation of an embedding is done by training it like a machine learning model. There are many different models for embedding but the one used for this thesis is word2vec continous skip-gram[12]. The skip-gram model takes one word I as input and a set of words C (as in context) as target. The model is based on the softmax function so that:

$$p(w_j|w_I) = \frac{\exp(u_{c,j})}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{1}$$

Where V is the training set, $j' \neq j$ and

$$u_{c,j} = u_j = {v'_{w_j}}^T \cdot {v'_{w_I}}^T \tag{2}$$

where v'_w is a vector from the embedding matrix. The loss function E is defined as:

$$E = -\log \prod_{c=1}^{C} p(w_{c,j}|w_I) \tag{3}$$

For each iteration the values in the embedding are updated one vector at a time with equation 4 [13]:

$${v'_{w_j}}^{[new]} = {v'_{w_j}}^{[old]} - \eta \sum_{c=1}^{C} \frac{\partial E}{\partial u_{c,j}} {v'_{w_I}}^T \tag{4}$$

3.4 Data skew

Data skew is a scenario when a dataset is unevenly distributed. There are two kind of data skews; negative and positive skewness.

In a normal distribution (symmetric bell curve), the mean, median, and mode are the same value. In comparison the skewed distribution, the mean, median,and mode have different values

The positively skewed distribution have a larger set of extreme values in the data. This cause the mean to be higher than the median. The negative skewed distribution is the opposite. The mean is lower than the median as the extreme values is a lot less in the data. In figure 2 an example of negative data skew is presented in a normal distributed graph. One way to handle data skew is transformation [14]; logarithms or square root. According to Pearson's moment coefficient of skewness the calculation of skewness for a random variable X, equation 5.

$$\gamma_1 = E[(\frac{(X - \mu)}{\sigma})^3] \tag{5}$$

Where μ is the mean, σ is the standard deviation, E is expectation operator. The formula a sample of n values, equation 6.

$$b_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}} \tag{6}$$

Where $\overline{x}$ is the sample mean.

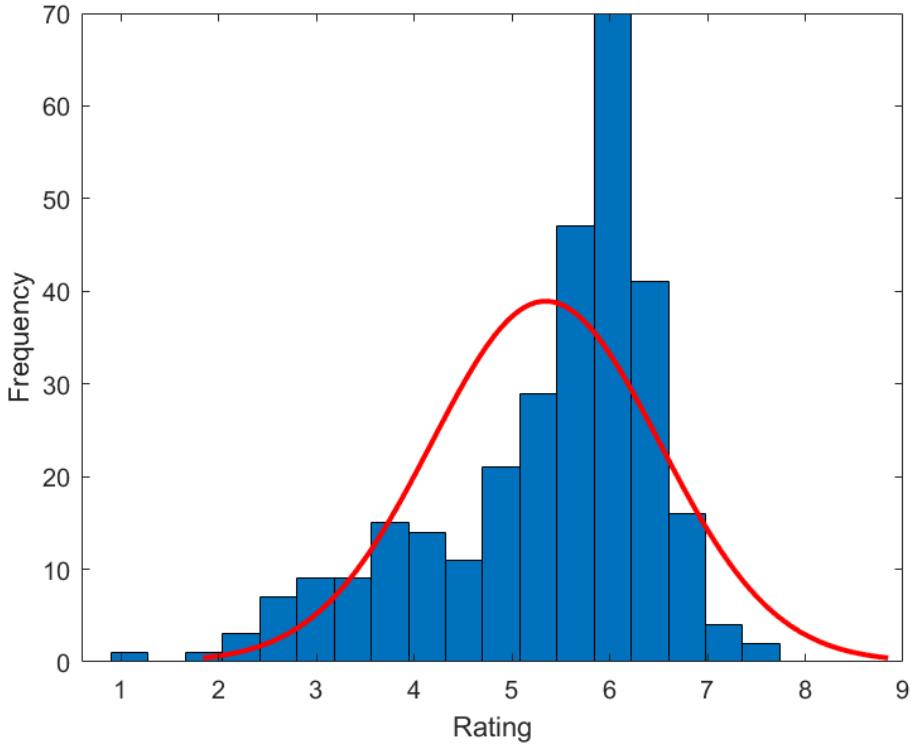


Figure 2: An example of negative data skew when the results contains a smaller set extreme values.

3.5 Regularisation

Overfit can occur when a ML model is over-trained [15]. It can be seen as the model memorised the training data rather than learning trends, resulting in negative impact on the validation. An example of overfit is displayed in figure 3.

Overfitting can be avoided by applying regularisation methods. A common method is elastic net regularisation, eq: 7 [16], however this method is not suitable for recurrent neural networks (RNN).

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|_1) \tag{7}$$

For RNN regularisation dropout [17] is a simple yet effective way to help the model generalise. Dropout means to omit the weights of a unit in the

network randomly at a given probability. The probability of a output unit being dropped can simply be calculated with $1 - \varphi$ were φ is the probability of drop.

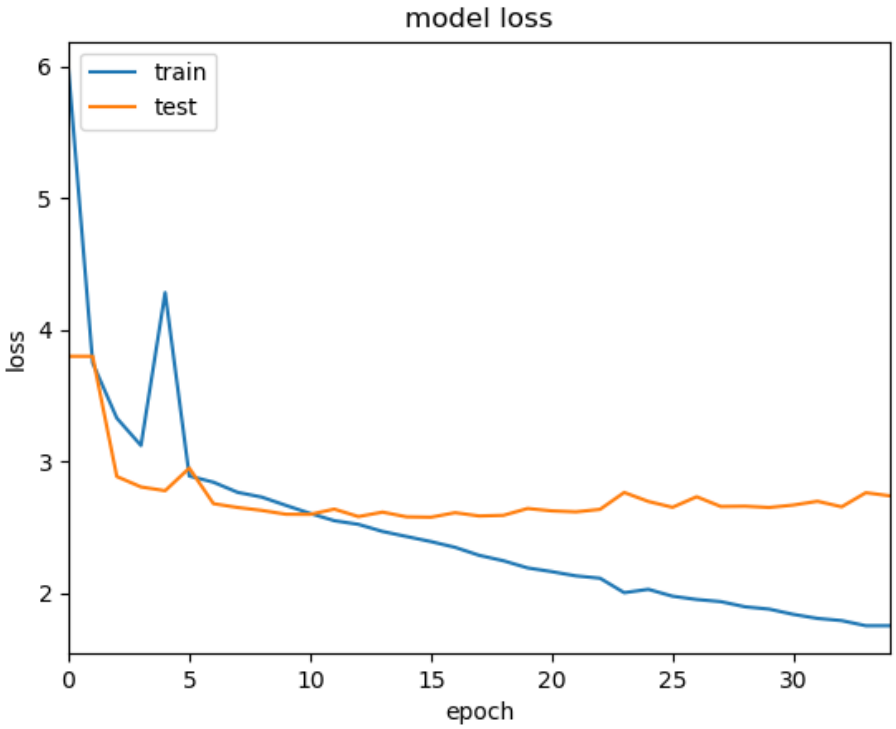


Figure 3: Result on overfit between the training set and test set

3.6 Natural language processing

Natural language processing (NLP) is a process to create an understanding between a human and a computer. This includes inputs from text and audio, real time or batch processing. The difference between the human language and computers is that the human language tend to be a bit more unstructured. In comparison to a computer language that operates on its binary structure is a lot faster than the human language.

Natural language processing [18] is a sub-field of Artificial Intelligence (AI) that focuses on giving a computer the ability and to analyse and process the human language. Computers is yet to understand the natural language to its full potential as languages contains a lot of data and sometimes the hidden meaning is not so clear without reading the whole picture. This can be referred

to as a idiom.

When working with natural language processing the problems mostly shows when working with the idiom. As an example: “Kill two birds with one stone” would for a human mean that someone has accomplished two things at the same time. The computer however is most likely saying that someone killed two birds with one stone. This is a unacceptable result from the machine learning and far from being a accurate. NLP is built upon utilising algorithms [19] and rules to analyse and categories data, in this case text, to associate meaning with each sentence and collect the important information from them. This is not 100% guaranteed that it will work and may produce vague results.

3.7 Classification & Regression

Classification and regression are branches for supervised machine learning. Regression is best applied to attempting to predict numerical or continuous changes in information while classification attempts to predict discrete or categorised output.

Both regression and classification use an input layer, hidden layer and output layer [20]. Input layer consists of information describing a set of data. An example of information that can be described in the input layer is an embedding. The hidden layers consist of the algorithms that are used, such as GRU and LSTM, to adapt and find connections between information. Usually composed of weighted inputs, a transformation function and activation function. Lastly, there are a output layer that produce the results created in earlier stages. One way to check the accuracy of the methods is to use Confusion matrix and Boxplot. Each method is implemented to determine how good the model performs against the test set.

3.8 Confusion matrix & Boxplot

Confusion matrix is also known as an error matrix. Confusion matrix is a table layout that allows one to visualise the performance of the result. It is often used for supervised and unsupervised machine learning. Each row presents a predicted class while each column presents the observed value. The accuracy of the confusion matrix can be explained in eq 8 were; true positive (TP): correct predicted, true negative (TN): correct predicted, false positive (FP):

false predicted, false negative (FN): Did not predict.

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

An alternative way of describing information is via the Boxplot. Boxplot is best suited to regression as it does not consist of categorised values in comparison to confusion matrix. Boxplot describes whether a distribution is skewed and whether there are potential unusual observations or outliners. By finding; The upper and lower quartile, the median and upper and lower extremes, it is possible to describe a set of information. As an example in results, page 27 where the median is on the upper quartile which is 10, which shows that the information is skewed.

3.9 Keras, LSTM & GRU

Keras is a framework for neural network (NN). Keras contains multiple-number implementations of commonly used neural-network blocks included; layers, activation functions and optimises to facilitate and create an easy and user-friendly framework.

Neural networks are made of neurons and connections. A connection between two neurons means the output of one becomes the input for the next. A neuron is a building block for networks the that perform an algorithm on the input it receives [21]. The simplest algorithm is a first order polynomial[22]:

$$y = \sum kx + m \tag{9}$$

RNN stands for recursive neural network, a subgroup of NN. It can be used on any causal series. Since time series are causal and common, use of RNN is also common. In this thesis RNN is used on reviews. The words in a review aren't necessarily written in order but they are read as a causal series allowing RNN to be used on it.

The principle of RNN is that each neuron feeds it's output back into itself making the last output a function of all previous. This way a full series can be represented with just one item. Jordans networks algorithm as example 10 :

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{10}$$

Where x_t is the input vector, h_t is the internal state, y is the output and σ are activation functions. W, U and b are the parameter matrices and vectors.

Within Keras there are two common architectures; LSTM and GRU . LSTM (Long short-term memory) is a RNN architecture that uses so called gates to control the data flow. LSTM consists of a cell, input gate, output gate and a forget gate. The cell remembers the value of arbitrary time intervals and the three gates regulate the flow of information in and out of the cell eq 11 .

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + W_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{11}$$

Where $\circ$ denotes element-wise product, f_t is the forget gate, i_t is the input gate, o_t is the output gate, c_t is the cell state vector and h_t is the output vector. The initial states of $c_o = 0$ and $h_o = 0$. The activation functions σ_g is a sigmoid function, σ_c a hyperbolic tangent function, σ_h a hyperbolic tangent function or $\sigma_h(x) = x$ in the peephole version of LSTM.

The other architecture is GRU (Gated recurrent units), that is using gating mechanisms within the RNN. It is very similar LSTM with forget gate. A difference is that GRU lacks an output gate, which means that it has fewer parameters than LSTM, eq 12. It is proven in previous studies that LSTM cells outperform GRU cells in larger datasets [23] but GRU provides a better performance on smaller sets of data [24]

$$\begin{aligned} z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\ h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \end{aligned} \tag{12}$$

Were z_t update gate vector, r_t reset gate vector.

3.10 Preprocessing

Preprocessing is a data mining technique that involves transforming raw data into an understandable format. Raw data sets is generally incomplete, this includes problems such as missing attribute values, missing certain attributes

of interest, or containing only aggregate data, noisy or inconsistent. Data preprocessing is a proven method of resolving such issues. Preprocessing consists of methods for handling poor or incorrect information to create a normalised data set. By transforming, reducing and removing impossible combinations, it is possible to reduce the information and create a data set that is more efficient to manage and improve the result [25]. The goal of the method is to improve the accuracy of predictive tasks and minimise time complexity. The worse the input, the harder it will be for a machine learning to find a pattern. It is possible to describe this as a contradiction, if a review claims to be positive and the same review occurs but negative. This can be referred to as a false positive. The model will adapt to this and the more errors that are present in the data set will result in a poorer performance result. There is no proper way to perform a preprocessing process, but it often involves analysing the data, removing missing values, reducing redundant data.

3.11 Optimiser & Loss function

RMSProp (Root Mean Square Propogation) is an unpublished, adaptive learning rate method proposed by G. Hilton [26]. It reduce the oscillations on a different approach compared to momentum. It removes the need to change the learning rate and adapt it automatically [27] . Each parameter in the RMSProp has its own learning rate by exponentially decaying average of squared gradients. β is suggested to be set towards 0.9 and υ_{dw} towards 0.001 as default value. ϵ is a small scalar to prevent division by zero, equation 13.

$$\begin{aligned} \upsilon_{dw} &= \beta \cdot \upsilon_{dw} + (1-\beta) \cdot dw^2 \\ \upsilon_{db} &= \beta \cdot \upsilon_{dw} + (1-\beta) \cdot db^2 \\ W &= W - \alpha \cdot \frac{dw}{\sqrt{\upsilon_{dw} + \epsilon}} \\ b &= b - \alpha \cdot \frac{db}{\sqrt{\upsilon_{db} + \epsilon}} \end{aligned} \tag{13}$$

Alternative to RMSProp is Adam (Adaptive Moment Estimation) that can be seen as a combination of RMSProp and Stochastic Gradient Descent with a momentum. The moment of a random variable can be described eq 14 were m is the momentum and X is the expected value of that variable to the power of n.

$$m_n = E[X^n] \tag{14}$$

Secondly the m and v are moving averages. The idea is to run the average of both the gradients and the second moments of the gradients are used, eq 15.

$$\begin{aligned} m_w^{(t+1)} &\leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^{(t)} \\ v_w^{(t+1)} &\leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)(\nabla_w L^{(t)})^2 \\ \hat{m}_w &= \frac{m_w^{(t+1)}}{1-(\beta_1)^{t+1}} \\ \hat{v}_w &= \frac{v_w^{(t+1)}}{1-(\beta_2)^{t+1}} \\ w^{(t+1)} &\leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon} \end{aligned} \tag{15}$$

The given parameters w^t and loss function; L^t, forgetting factors β_1, β_2 for gradients and second moments of gradient and ϵ is a scalar to prevent deviation by zero. The calculation of loss function utilises Mean Absolute Error (MAE) equation 16 which takes sum of absolute differences between two continuous variables, observed and predicted value. It measures average magnitude of errors in a set of predictions [28]. y_i is the observed, y_i^p is the predicted value and n is the number of samples.

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - y_i^p \mid}{n} \tag{16}$$

4 Methods

4.1 Method Description

One of the goals with machine learning is to produce high precision during testing without overfitting the model too much. Different frameworks are used to facilitate the creation of machine learning and data processing of which several are well documented and easy to use. Measurements for loss, validation loss and average duration are carried out to present the best model. In absence of Big Data, methods for generalisation were applied to help the model handle a lower dataset. Generalisation is further enforced by tokenizing the data into a third party word embedding.

4.1.1 Framework

The project was constructed with a number of frameworks and packages, including; Keras, gensim, pandas, zipfile and sklearn. The respective packages may in turn contain several imported packages, seen in table 12, which provide methods for importing, processing and analysing data from embedding and the dataset.

The core package is Keras. It's explained as: "A high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation."[2] Tensorflow is used as the backend in this case.

4.1.2 Analysis of dataset

The input data plays a major part in determining how good quality the data is and how the preprocessing should be carried out. The analysis consists of getting an overview of the dataset with boxplots and understanding the relationships between different reviews through manual processing. At a quick overview, it became clear that information was missing, several misspellings and wrong language existed. This means that this information must be filtered using preprocessing methods.

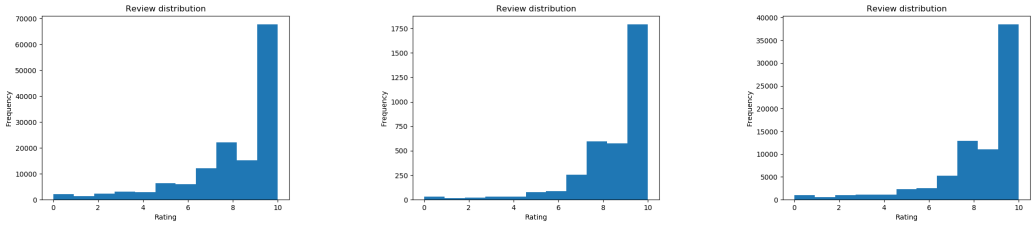
[2] `https://Keras.io/`

The data sets that are imported consist of two files; CSV and XMLX, both of which contain the information for Swedish, Danish and Norwegian. The data set is structured with two columns; NPS and comment. NPS stands for net promoter score which is used as a management tool to measure how loyal the costumers are to a company. Note that the values stored in the NPS column is not the NPS itself but the ratings used to calculate it. The ratings consists of a value between 0 - 10 where 0 is considered lowest and 10 is the highest rating a user can give. In addition to NPS there is a comment column that contains a review that users can add in addition to the rating. NPS is considered the label and comment the specified input.

The NPS is divided into three categories to facilitate the classification and be able to interpret the output from the regression model. User reviews between 0-7 are considered bad, 8-9 are neutral and 10 are good reviews. This is to be able to handle data skew that can be observed in figure 4.

In figure 4 the distribution of ratings in their respective language is displayed. Figure 4a shows label distribution for the Swedish dataset. The same distribution can be seen for Norwegian in figure 4b and for Danish in figure 4c.

The data is negatively skewed, this means that there's a disproportionate number of labels. Since the mid and lower ratings are underrepresented the models performance became much lower in this span.



(a) Swedish (b) Norwegian (c) Danish

Figure 4: Review distribution of each language

4.1.3 Preprocessing

In order to utilize an embedding and dataset, a preprocessing must be performed. This process is illustrated in figure 5. The first step is importing the embedding set into the Python project by calling the method ZipFile from the package zipfile, with the arguments; $repository + "69.zip", "r"$ where repository is the selected directory, $69.zip$ is the file name of the database and

"r" is a mode where the file should be seekable. Thereafter the archive is opened into a stream for the gensim package to import it to the embedding. Utilising $gensim.models.KeyedVectors.load\_word2vec\_format$ with the following arguments found in table 3 generates the embedding with the available database.

Argument	Input	Explaination
fname	stream	The datastream loaded from the archive
binary	False	Load the data as word2vec plaintext.
unicode_errors	'replace'	Replace data where unicode error appears

Table 3: Explanation of the settings for embedding import through gensim.

In addition to the embedding, the dataset should also be imported. The dataset consists of two different files of different formats, XMLX and CSV, were XMLX contains seven different sheets in three languages; three Danish, three Swedish and one Norwegian. The CSV file contains another dataset for Swedish

- The XMLX is imported using Panda's excel reader, $pd.io.excel.read\_excel$, to create a dictionary. The arguments include a path to the file and $dtype = \{'Comment' : str\}$ as the sheet consist of two columns; NPS (rating) and $Comment$ and $sheet\_name$ that is the selected sheets. They are thereafter fused into a list using Panda's concat with arguments; $ignore\_index = True$.
- The CSV is imported using Panda's csv reader, $pd.read\_csv$. The arguments include for the import is found in Table 4. As the CSV file used in this project have column named in dtype is not consistent with the XMLX they are replaced with $\{'aw' :' Comment',' asd' :' NPS'\}$.

Argument	Input	Explanation
	Path	The datastream loaded from the archive
sep	';'	Load the data as word2vec plaintext.
dtype	{'aw':str,'asd':float}	Set datatype of aw; string and asd; float
encoding	'utf_8'	Set encoding
axis	'columns'	Axis to target with mapper

Table 4: The arguments used for the CSV import

After XLSX and CSV were merged into a large unsorted list, all rows that lack values are dropped from the set. To help the model generalise, all non-

alphabetic characters are removed. This includes, question mark, point, exclamation mark, and more. At the same time, all capital letters are replaced with lower case ones.

The last cleaning step is tokenization which involves checking if and where all the words in the dataset exists in the embedding. By splitting the sentences into words, comparing the words to the vocabulary and adding the words index to a list if it exists. If the operation fails, a zero is added as a representation for any word out of the vocabulary. Each word has a unique number that represents their index in the embedding. The indexes in turn describe the frequency of the word where the most frequent word in the indexes starts at 1 for 0 to be reserved for out of vocabulary.

Before the information is sent to the created model, the dataset must be padded or truncated to a fixed length to keep the dimensions consistent. This is done since the Keras framework requires consistent dimensions.

Lastly, the dataset is split into four sets; $x\_train, x\_test, y\_train, y\_test$, using $train\_test\_split$ from sklearn, where each row is randomly selected with a ratio of 30% to testing and 70% to training. The ratio is based on maximising the amount of training data but at the same time enough test data to determine if the model is functioning. The testsets are saved for validation later on to check if the model can generalise. A larger testset means a more validated result but it costs training size which affects if the result is good or not.

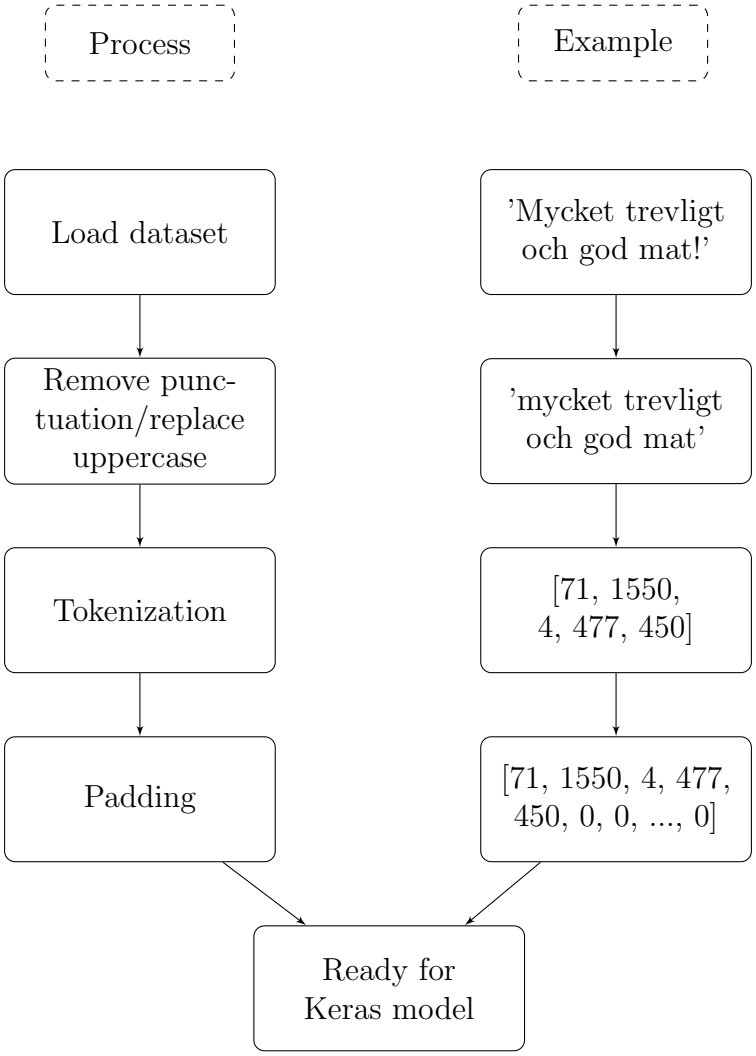


Figure 5: The planned preprocessing structure of the project.

4.1.4 Keras model

The Keras model consists of a sequence of layers. All layers take an input and produce an output which then gets passed into the next layer. The layers used in this model are; Embedding, LSTM, GRU and dense. The models in this project were composed of layers as shown in figure 6. Where BATCH is number of training examples utilised in one iteration, MAXLEN is the maximum length of input, 100 is the embedding dimension and the OUTPUT is the dimension of the final output. In this study the regression model will output one value for sentiment and the classification model will output three values: probability for good, neutral and bad sentiment. Were good is defined as a rating of 10, neutral as 8 to 9 and bad as 0 to 7.

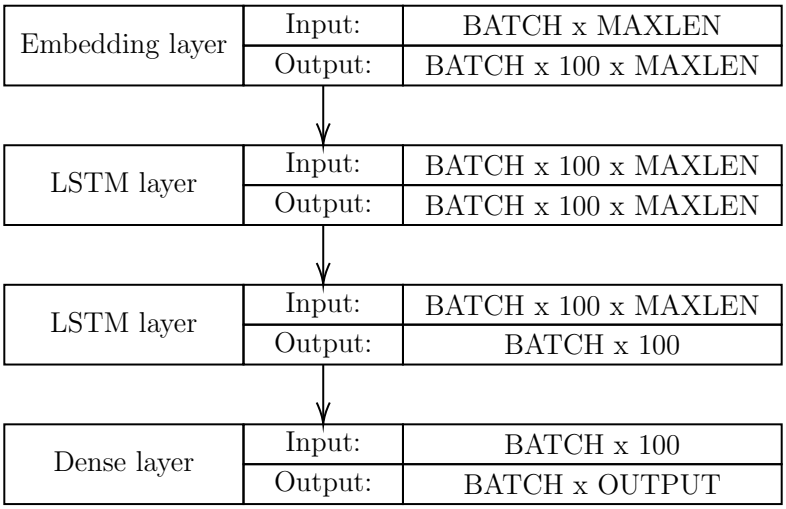


Figure 6: The planned Keras structure of the model.

The embedding used was pretrained. In the table 5 the arguments used to initiate the embedding as a pre-trained one are presented. Training of the embedding was disabled by setting $trainable = False$ to ensure it stays impartial to the training data.

Argument	Input	Explanation
input_dim	len(embedding.vocab)	Dimension length of the vocabulary
output_dim	embedding.vector_size	The vector size of the embedding
weights	[embedding.vectors]	The weighs from the embedding
input_length	MAXLEN	Maximum length of the training data
trainable	False	Set layer to be none trainable

Table 5: The Keras arguments for the embedding.

In order to create a model object the constructor, $Sequential()$, is called. The first layer added to the model needs to have input dimensions defined but the following layers can automatically infer dimensions from its previous layer. An exception to this is embedding layers that can't infer shape and thous they must be added to the model first of all.

Recurrent layers were added after the embedding. Recurrent layers are ideal when processing causal data series. A property shared between all is that they reduce the data flows dimension by one. In order to chain recurrent layers this property can be bypassed by setting $return\_sequence = True$. The two recurrent architectures explored are Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM).

Table 6 shows the arguments changed from their default values the recurrent layers were constructed. The return_sequence is $True$ for all recurrent layers except the last one. Recurrent_dropout and dropout was set to 0.5.

A final dense layer was added to collect all the information from the neural network and compile it into the desired shape.

Argument	Explanation
units	Positive integer, dimensional of the output space.
dropout	Fraction of input to drop [0-1] from the input
recurrent_dropout	Fraction of input to drop [0-1] from the recurrent state
return_sequences	Whether to return the last output in the output sequence, or the full sequence.

Table 6: Arguments being modified for Keras LSTM and GRU layer

To finalise the model its *compile* method was called with the arguments: $optimiser =' rmsprop', loss =' mae'$.

Once the model was finished the training procedure were started using the $model.fit$ with the given arguments in table 7.

Argument	Explanation
x	Array of training data
y	Array of training data
batch size	Size of the batch
epochs	Number of epochs to train the model
verbose	Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch.
validation_data	Tuple, (x_test, y_test), to evaluate the loss and any model metrics at the end of each epoch.

Table 7: Arguments modified for Keras model.fit

In order to be able to distribute the same amount of time on each model, the number of units was changed, presented in section 5 and thus treating the imbalance in speed between the corresponding layers.

4.2 Verification of models

Table 8 illustrates metrics used to validate the results. Main metric used for validation was Spearman correlation between true labels and predicted labels. For classification models the predicted label was derived from the label with the highest probability. Target value for this metric is 68%.

Secondary metric for the regression model was boxplots. For the classification model a confusion matrix was used instead. Ideally these metrics should show a bell curve distribution.

Priority	Metric	Target	Model
1	Spearman correlation	68%	Regression
			Classification
2	Boxplot	Bell curve distribution	Regression
	Confusion matrix		Classification

Table 8: Metrics used to validate the model

The tests that are carried out are a repetition of the known model with different parameters for GRU, LSTM, table 6, and model.fit, table 7. Each test is presented using boxplot, confusion matrix and a model loss, fig 3, section 3.5. The information shows how the model performs against predicted and observed values. When a model performs better through lower loss and higher accuracy, the previous model replaces and is used as a basis for newer measurements. To give a summary value, a correlation analysis is extracted to compare the values from the training data and the test data to see the percentage result of the model.

4.3 Web service integration

The integration into a website means that it must be possible to access the information from external sources and be able to extract a tonality from user reviews. By including (import package) and loading the models from keras model it is possible to predict an information of an input. The models are loaded in memory to quickly return information to the client.

5 Results

The correlation on LSTM and GRU shows the relation between the observed and predicted value of a testset. The lower the value, the worse the model can be considered. The ultimatum would be to converge with 100% but then it should be in consideration that the model could have overfitted during the training and memorised the testset.

5.1 Classification model

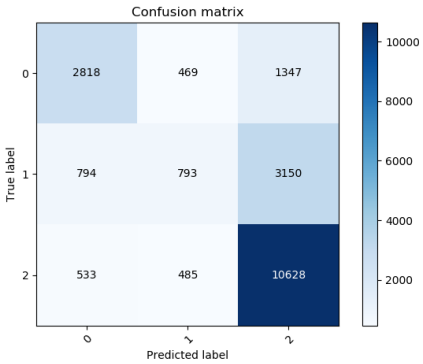
Architecture	Batch_size	Epochs	Units	Correlation
LSTM	40	15	80	55.0%
GRU	50	15	100	58%

Table 9: Correlation for the classification models.

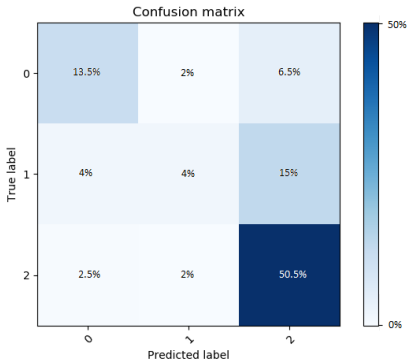
The GRU based model resulted in a 58% correlation with the distribution illustrated by figure 7a. For the LSTM build a correlation of 55% was obtained and a distribution as shown in figure 7c. How the labels translate into sentiment is shown in table 10. In order to balance the class distribution the bad ratings contain a wider span from 0 - 7 to compensate the low amount of negative review to deal with data skewness that can be observed in figure 4. The accuracy for GRU in figure 7a is 68.22% and 66.77% for LSTM in figure 7c.

Label	Sentiment	Rating
0	Bad	0-7
1	Neutral	8-9
2	Good	10

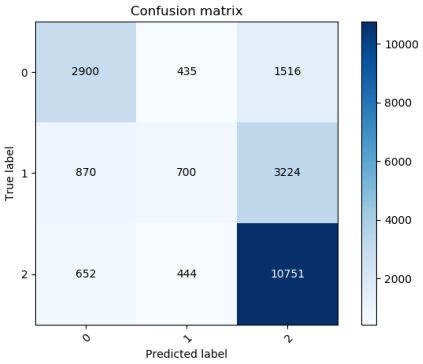
Table 10: Illustration on how the classification labels should be interpreted.



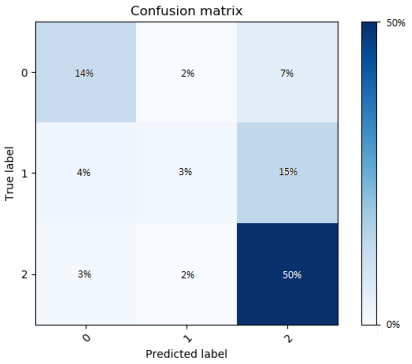
(a) GRU numeric



(b) GRU procent



(c) LSTM numeric



(d) LSTM procent

Figure 7: Confusion matrix for both classification models. Label 0 = bad, 1 = neutral and 2 = good rating. The GRU model resulted in 55.0% correlation and the LSTM one got 58% which means they fail the primary metric (see table 8). Note how both models struggle with label 1 which means they fail the secondary metric aswell. The total number of measuring points for 7a & 7b is 21017 and for 7c & 7d is 21492.

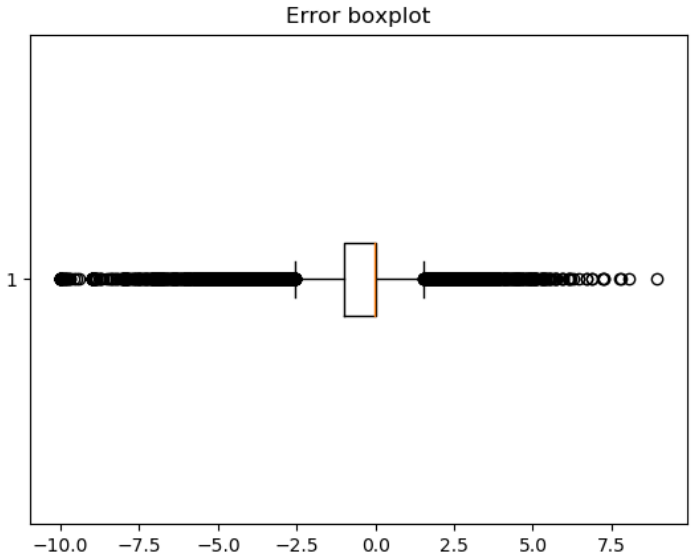
5.2 Regression model

Architecture	Batch_size	Epochs	Units	Correlation
LSTM	50	40	80	65.8%
GRU	50	15	100	61.7%

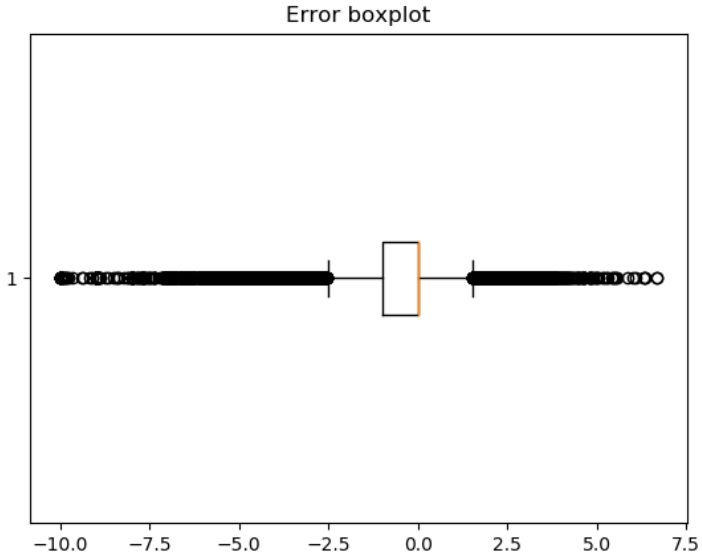
Table 11: Correlation for the regression models.

The GRU based model resulted in a 61.7% correlation with the distribution illustrated by figure 8a. For the LSTM build a correlation of 65.8% was obtained

and a distribution as shown in figure 8b.



(a) GRU



(b) LSTM

Figure 8: Boxplot of the regression models showing the distribution of error

6 Discussion

6.1 Analysis of dataset

The dataset is the most important part in a machine learning model. In this project a poor quality set was used, table 2, and as result the models performance suffered. Different companies had different input sanitation procedures and it's the worst one that sets the bar for the dataset. Half the Swedish data was of such poor quality the model struggled to converge with tests using mean square error as loss. Because of that a large portion of the data was discarded during the preprocessing. Examples of bad data found in the datasets are: wrong language, wrong label, multiple labels on one cell and empty cells. Even the best datasets used were of poor quality which sparked the question whether or not to refine the data. This was not done since according to the delimitation's set, this is not included in the project. If the processing of the dataset was included in the work, alternative or additional methods for preprocessing would be implemented. This would involve implementing methods mentioned in [25] to handle the problems of noisy data, missing values and data redundancy. A suggested remedy to the mislabelled data would be to use machine learning to find labels that deviate strongly and drop them from the dataset. The data cannot be explicitly presented due to confidentiality.

6.2 Analysis of result

A previous study delivered 86.32% accuracy using a GRU model for sentiment prediction [8]. This number as many others aren't comparable to the results in this thesis. This is because none of them use the same metric and the same dataset. The only source that can be used for comparison is the goal of 68% correlation mentioned by Quicksearch. The results are not protruding, considering the dataset and the false negative responses that lower the accuracy of the result. This implies the result is reasonable since the correlation resulted in 58% using classification with GRU and 65.8% using regression with LSTM. The classification did not reach the target value of 68% which means that it should be rejected. Regression was just on the edge and can be retained but should also be improved in the hope of passing the specified target. Both models use RMSProp as optimiser. The regression model use mean average error (MAE) as loss function while the correlation use categorical cross-entropy. In the end regression performed better. The LSTM model in regression also managed better than GRU. This was because the GRU model began to overfit. Note that if LSTM had been given more cpu time it would also have overfitted.

The method used shows that the dataset affects the result and a greater diversity of information is needed to get better results. Through the resulting classification, one can see that the model receives about $66 - 68\%$ accuracy, which says how much of the total set that the model guessed correct. The result is better than a guess but not high enough to be considered robust and safe for sensitive areas such as decision making. Regression peaked with a median of 10 which shows the skew of the information. Several measurement points of low values ended up within the lower quartile of the box plot. The model can to some extent predict bad reviews but this is assuming that the words are not of mixed sentiment.

The end result has been based on the assumption that the preprocessing is sufficient and that further modification of the dataset should not be done. This means that it is assumed that with enough information, both the classification and regression models can handle error margins from the input and label.

To be able to implement better models, more time and better hardware would be needed. During the tests, limited hardware was used for CPU, memory and hard drive, which delayed the creation of models. Idealy the machine learning process should be accelerated by a dedicated GPU. The more epochs added the longer time it will take to create a model. This makes it impossible to execute too many epochs without losing time that could be used to create more models.

6.3 Signal processing

One of the first experiments was carried out using signal processing. This was based on treating the input as a signal to make a more general model. Using cross-correlation between two signals to find a specific pattern as well as similar patterns. The idea was that anything that is a close enough match to a pattern would have similar sentiment. However the correlation of two words with different sentiment could just as well have greater similarity than a word containing a misspelling of the pattern. It was shown that only exact matches of the pattern would share sentiment which obsoleted the whole idea.

6.4 Social aspects

The project focus on several aspects for economical, integrity, security and efficiency. The project show how to improve manual data analysis. From an

economical perspective manual analysis of big data isn't viable. Instead one should rely on a data processing method such as NLP. Eliminating exposure of private information towards external sources which keeps the integrity towards GDPR and still allow companies to review the rating of their company. There are many perspective on machine learning and how the efficiency is improved in different areas including energy and time consumption. According to Google that reduced their energy consumption for cooling with 40% and 15% overall energy overhead[3] is one of many achievements proved by machine learning.

6.5 Future work

Refining the dataset would greatly improve the results of this model but there's also room for improvement on the model itself. Keeping periods during pre-processing but as a separate word is a practice that should be tested. Another practice missed is down-sampling and up-weighting to balance the dataset. Using an attention based model would allow longer sequences to be handled which would help generalise the model. Making the sentiment more fine grained by extracting one sentiment per topic would improve usefulness of the model. Support for multiple languages and automatic detection of languages would allow the model to handle the mixed languages present in the dataset.

[3] `https://sustainability.google/projects/machine-learning/`

7 Conclusion

The project presents a method for creating a singular model for three Nordic languages using deep neural network. The presented result shows two different models for classification and regression. In each model two different layers; LSTM and GRU were tested. The result with the highest correlation was 65.8% and was produced using regression model and two LSTM layers. The confusion matrices shows the model often guesses too high and the boxplot quartiles shows the same for the regression model. This suggests the dataset is unbalanced. Topic analysis was not available for arbitrary results and was therefore excluded from the project. With limited performance in hardware, additional fine-tuning was not possible. Future work could include refining the model and dataset. Add support for detection of language in mixed dataset and complete the topic analysis.

During the project, questions arose about methods for improving the dataset and alternative solutions in order to manage information related to big data and GDPR.

References

[1] Shalunts G, Backfried G, Commeignes N. The impact of machine translation on sentiment analysis. Data Analytics. 2016;63:51–56.

[2] Mikolov T, Kopecky J, Burget L, Glembek O, Cernocky J. Neural network based language models for highly inflective languages. 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. 2009;Available from: http://dx.doi.org/10.1109/ICASSP.2009.4960686.

[3] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed Representations of Words and Phrases and their Compositionality. Neural Information Processing Systems (NIPS); 2013.

[4] Liu B. Sentiment analysis and opinion mining. Synthesis lectures on human language technologies. 2012;5(1):1–167.

[5] Pak A, Paroubek P. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. vol. 10; 2010. .

[6] Hu M, Liu B. Mining and summarizing customer reviews. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2004. p. 168–177.

[7] Wang Y, Huang M, Zhao L, et al. Attention-based LSTM for aspect-level sentiment classification. In: Proceedings of the 2016 conference on empirical methods in natural language processing; 2016. p. 606–615.

[8] Yin W, Kann K, Yu M, Schütze H. Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:170201923. 2017;.

[9] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436.

[10] Sorzano COS, Vargas J, Montano AP. A survey of dimensionality reduction techniques. arXiv preprint arXiv:14032877. 2014;.

[11] Sun S. A survey of multi-view machine learning. Neural Computing and Applications. 2013 Dec;23(7):2031–2038. Available from: https://doi.org/10.1007/s00521-013-1362-6.

[12] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013;.

[13] Rong X. word2vec parameter learning explained. arXiv preprint arXiv:14112738. 2014;.

[14] Changyong F, Hongyue W, Naiji L, Tian C, Hua H, Ying L, et al. Log-transformation and its implications for data analysis. Shanghai archives of psychiatry. 2014;26(2):105.

[15] Provost F, Fawcett T. Data Science and its Relationship to Big Data and Data-Driven Decision Making. Big Data. 2013;1(1):51–59. PMID: 27447038. Available from: https://doi.org/10.1089/big.2013.1508.

[16] Zou H, Hastie T. Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology). 2005;67(2):301–320.

[17] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research. 2014;15(1):1929–1958.

[18] Hirschberg J, Manning CD. Advances in natural language processing. Science. 2015;349(6245):261–266.

[19] Jurafsky D, Martin JH. Speech and language processing. vol. 3. Pearson London; 2014.

[20] Gulli A, Pal S. Deep Learning with Keras. Packt Publishing Ltd; 2017.

[21] Dawson CW, Wilby R. An artificial neural network approach to rainfall-runoff modelling. Hydrological Sciences Journal. 1998;43(1):47–66.

[22] Jordan MI. Serial order: A parallel distributed processing approach. In: Advances in psychology. vol. 121. Elsevier; 1997. p. 471–495.

[23] Britz D, Goldie A, Luong MT, Le Q. Massive exploration of neural machine translation architectures. arXiv preprint arXiv:170303906. 2017;.

[24] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555. 2014;.

[25] García S, Luengo J, Herrera F. Data preprocessing in data mining. Springer; 2015.

[26] Hinton G, Srivastava N, Swersky K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on. 2012;14.

[27] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980. 2014;.

[28] Chai T, Draxler RR. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geoscientific model development. 2014;7(3):1247–1250.

8 Appendix

Package	Version
absl-py	0.7.1
astor	0.7.1
backcall	0.1.0
backports.functools-lru-cache	1.5
boto	2.49.0
boto3	1.9.139
botocore	1.12.139
certifi	2019.3.9
chardet	3.0.4
cheroot	6.5.5
CherryPy	18.1.1
colorama	0.4.1
cycler	0.10.0
decorator	4.4.0
docutils	0.14
gast	0.2.2
gensim	3.7.2
grpcio	1.19.0
h5py	2.9.0
idna	2.8
ipython	7.4.0
ipython-genutils	0.2.0
jaraco.functools	2.0
jedi	0.13.3
jmespath	0.9.4
Keras	2.2.4
Keras-Applications	1.0.7
Keras-Preprocessing	1.0.9
kiwisolver	1.0.1
llvmlite	0.28.0
Markdown	3.1
matplotlib	3.0.3
mock	2.0.0
more-itertools	7.0.0
numba	0.43.1
numpy	1.16.2
pandas	0.24.2
parso	0.4.0
pbr	5.1.3

pickleshare	0.7.5
pip	19.1.1
portend	2.4
prompt-toolkit	2.0.9
protobuf	3.7.1
Pygments	2.3.1
pyparsing	2.4.0
python-dateutil	2.8.0
pytz	2018.9
pywin32	224
PyYAML	5.1
requests	2.21.0
s3transfer	0.2.0
scikit-learn	0.20.3
scipy	1.2.1
setuptools	40.8.0
six	1.12.0
sklearn	0.0
smart-open	1.8.3
tempora	1.14.1
tensorboard	1.13.1
tensorflow	1.13.1
tensorflow-estimator	1.13.0
termcolor	1.1.0
traitlets	4.3.2
urllib3	1.24.2
wcwidth	0.1.7
Werkzeug	0.15.1
wheel	0.33.1
xlrd	1.2.0
zc.lockfile	1.4

Table 12: Packages used for the project

Jesper A. Holmblad, Computer Engineer 180 hp

C. Fredrik M. Mårtensson, Computer Science and Engineering 300 hp

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se