Bachelor Thesis
May 2020

The crucial parts of text classification with TensorFlow.js and categorisation of news articles

Gustav Nordberg,
Jesper Grandien

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering. The thesis is equivalent to 20 weeks of half-time studies.

Contact Information:
Author(s):
Gustav Nordberg
E-mail: guno17@student.bth.se

Jesper Grandien
E-mail: jegn17@student.bth.se

University advisor:
Michel Nass
E-mail: michel.nass@bth.se

Dept. Computer Science & Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Text classification is a subset of machine learning which is used to classify texts such as tweets, email, news headlines or articles, with tags or categories. As news publishing can have uncertainty in their categorisations, text classification could categorise articles autonomously and distinguish unclear categorisations. The library TensorFlow helps with operations and tools for the machine learning workflow.

This paper takes focus on the crucial parts of working with machine learning using TensorFlow.js and to what extent this model can categorise a news article. The authors evaluates different models to analyse how optimising the settings will affect the accuracy of the model.

Results of this paper was researched with a literature study of official documentations and peer reviewed reports. An empirical experiment where machine learning models were trained in TensorFlow.js was also performed.

The results showed that the model with the highest accuracy with 87.17% accuracy was trained with 1000 articles using Relu and Softmax activation functions and the Mean squared error loss function. While the model with lowest accuracy had 75.5% using Sigmoid activation functions and Categorical cross-entropy on the 5000 articles training set. Crucial parts for this development were: optimizer function, loss function, batch size, activation functions, training data and test data with labels, normalise function, shapes of layers and computing power.

There are several parts and functions to take in consideration when developing a machine learning model with text classification in TensorFlow.js. The training process needs to be performed multiple times as there are many parameters which has an affect on the model results. The model results can be improved by optimising and finding the best combination between different functions and parameters.

Keywords: TensorFlow.js, Machine learning, Text classification, JavaScript

Contents

Glossary

machine learning (ML) is the scientific study of creating computer programs with the ability to learn from experience [9].

API stands for application programming interface and is used to communicate between two applications. I.e software programs or libraries [21].

hyperparameter In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins. An example of a model hyperparameter is the learning rate or batch size [14].

dataset is a collection of data, I.e statistics gathered in the form of a database table [7].

Chapter 1

Introduction

1.1 Background

machine learning can be used for decreasing the amount of manual tasks like categorisation of articles or finding multiple articles covering the same event from a selection of articles. Machine learning models can as well strengthen medical diagnosis of x-ray results (image classification) and many other use cases, such as removal of background noise from microphones.

TensorFlow is a library made by Google that offers tools and an API for operations to be used when working with machine learning [12]. Text classification is a process used to categorise texts by using "Natural Language Processing" which analyses text and classifies tags or categories based on the input text data. Text classification can be used to classify texts like tweets, emails, news headlines and articles. Text classification is a subset of machine learning [11].

The focus of this paper was to identify the crucial parts of developing a machine learning model that can improve the categorisation of news articles. The focus in the experiment of this paper is to categorise articles for the specific categories: politics, wellness and entertainment.

1.1.1 Motives

The goal was to manage to get a positive accuracy measurement, such as over 50 percent which has at least half of the predictions correct. With the authors current knowledge it seemed possible to reach 50-75 percent correct predictions, given that there is enough training data to access. The other hypothesis was that the results would be able to show a pattern of what parameters and functions works better than others and what the crucial parts of a machine learning Text classification model are.

The motive behind finding the crucial parts of machine learning and TensorFlow.js is to provide knowledge around the basics of these concepts for developers. Together with the empirical results of this paper, this can then be used to create new experiments and have this paper as comparison.

1.1.2 Value

The idea was to use machine learning to pick out the relevant articles based on a category which can be used to personalise the experience for news websites. This could help users find specific content much faster, matching articles by the user needs and increase the user experience. It would be beneficial for companies as it would be easier to recommend personalised content for their users. The results of this paper can prove machine learning accuracy in analysing text contents. It will also bring guidelines for using TensorFlow.js in development purposes and describe necessary techniques used.

1.1.3 Technical challenges

One of many challenges with machine learning and text classification is accuracy. The classifier needs to have a high accuracy to be trusted and useful. The result of not enough training data could hypothetically generate low accuracy. Even with enough data, there are factors that can affect the accuracy like optimizer- or loss functions.

Some problems that might occur within the experiment could be handling difficult words and synonyms that could be hard to process into a readable data format for the machine learning model. The model does not know the similarities of "1" and "one" if it is not trained for it. This can be mitigated with normalise functions that for example converts "1" to "one" everywhere in the text. A normalise function can also remove special characters to improve the efficiency of learning. It could also be difficult to find enough amount of test/training data that is unique. Similar data can result in an inaccurate model that only works for similar written articles. Another problem could be to identify patterns of parameters that perform better than others, this can be mitigated with high scale testing where each test only changes one parameter per test.

1.1.4 Newsworthy

The research should result in more clarity in online news publishing. For example, an article about Donald Trump speaking about healthcare could be categorised as both wellness and politics. The machine learning model could specify which category that article fits best into. This research paper can also be helpful on another level of detailed categorisation autonomously, which could improve efficiency and

reduce cost for the media companies.

1.2 Research Questions

- **RQ1:** What are the crucial parts for developing a machine learning model with TensorFlow.js, which can be trained to categorise news articles?
- **RQ2:** At what accuracy can a news article be categorised using Text Classification and TensorFlow.js?
- **RQ3:** How much can a machine learning model improve its accuracy by optimising the model topology, hyperparameters and functions?

Chapter 2

Related Work

2.1 Previous study

There are already existing reports about text classification such as the research paper "News Article Text Classification in Indonesian Language". This paper focus on finding suitable algorithms for Text classification on Indonesian news articles, where the datasets used are gathered by using a web crawler on the site "www.cnnindonesia.com". The paper explains text processing by using Lemmatisation and removal of stop words in order to minimise noise in the document [24].

The paper "Deep Learning methods for Subject Text Classification of Articles" tests different approaches for feature vectors with a deep neural network. The focus is to test different approaches for subject classification using 7 Wikipedia subject categories. The paper concluded that their approach of representing a document in a sequence of words converted to word2vec vectors, performed better than a standard bag-of-words approach where documents are represented as frequency-of-words feature vectors [23].

A paper of Text classification algorithms were also found, where the paper evaluate suitable structures, architectures, and techniques for text classification and performed a survey on the subject [33].

The paper "Character-level Convolutional Networks for Text Classification" explores the use of character-level convolutional networks that uses text classification. The comparison is toward traditional usage for models such as bag of words, n-grams and deep learning models. The authors conclude that factors such as the size of the dataset, if the text is curated and choice of alphabet affects the model performance and that the character-level convolutional network model is effective as a method of text classification [34].

Chapter 3

Method

3.1 Literature study

The authors have performed a literature review based on RQ1, where the essential parts of TensorFlow.js and machine learning is presented. The goal is to inform the reader in ways to approach the broad concept of machine learning using text classification with TensorFlow.js. This will be informative around concepts for machine learning and the stages of the development process.

The authors have reviewed sources that mostly are based on technical reports, software documentation and peer review publications that includes information about the structure of TensorFlow.js and the components required for a functional machine learning model.

Keywords used in the search were: Machine learning, TensorFlow.js and text classification. The databases used in finding this information were mostly digital, such as Google scholar and Diva portal as well as documentation for software libraries like TensorFlow.

There might be limitations in the collected data as related works are mostly based on empirical data from other experiments, which can have different outcomes. Specially, the results of trained models can differ even though the same training methods are being used.

3.2 Empirical experiment

The authors have performed an experiment on machine learning models, based on datasets from news sources, that uses text classification to categorise an article in order to answer RQ1 & RQ2. For the scope of this paper, the categories used to be classified are politics, entertainment and wellness.

To be able to analyse different parameters and values, the experiment contains a lot of different machine learning models. This is done in order to evaluate the different values and their effect on the models ability to categorise articles.

Articles will be gathered to JSON files and sorted into categories for training purposes. This is done by our own written function in JavaScript that iterates through the original JSON file from the dataset and creates new JSON files for each category in the dataset. The datasets will be used as both training data and test data for the machine learning models. The training data and the test data should both be unique sets but from same source, to avoid misleading results.

The dataset used is from kaggle [19] which is a website for public datasets. This dataset includes enough training data to cover all the models that were performed in the experiment. The data is formatted as JSON including a headline and a labeled category for each article, which will be used during training.

The models trained on 2500 includes the same articles as the models trained with 1000 articles as well as 1500 more. Same method was used for the models trained on 5000 articles, 2500 articles were from the previous models. The articles from the models trained on 1000 and 2500 are therefore included in the training of the models with 5000.

Based on research for the project, the authors found TensorFlow which is a library for machine learning models. It has a big community with over 56 000 discussions on Stackoverflow, which includes guides and discussions that improves the learning curve for users a lot [28]. This report is focused on TensorFlow.js which is a library based on TensorFlow that will comply with JavaScript developers, performing the same functionality.

The evaluation of each model was measured by accuracy for correct as well as incorrect predictions made. The library itself calculates an accuracy based on the training data in the model and how often the algorithm makes the correct decision of classification. The authors measured accuracy based on additional tests, in the form of percent. A set of tests with new articles, from April 2020, collected from same source were performed, to get a view of the performance on new articles. The testing script is built to create predictions for one specific category and print out the results, which was then noted in the figures 5.1 - 5.6. This was performed with multiple tests of 100 articles for each respective category. The tests were both performed on the correct chosen category(politics) as well as the remaining categories, which was calculated as a score and compared to other models presented in the figures 5.1 - 5.6.

The models were tested with a testing suite of articles that was gathered from the following websites:

- The Huffington post, old articles (2018)

- The Huffington post (April 2020)
- New York Times (April 2020)

These were collected by both manually writing JSON objects with a headline and category field for each article and a scraping script written in Python.

This testing suite had a score between 0-100 to display the amount of correctness the model has on new data as well as similar data from the same dataset source as the training set. This score is based on the accuracy of 100 testing articles from each category. The experiment was performed to evaluate the best results for articles within the category of politics (high score means high accuracy) by correctness as well as predictions on wrong categories, such as wellness and entertainment. Low score on wrong categories means that the model is good at filtering these wrong categories in the prediction.

Chapter 4

Literature review

The authors have analysed reports and official documentation that describes the fundamental parts in developing a machine learning model for the task of text classification. It provides general knowledge around machine learning basics and the structure of TensorFlow. The fundamental parts are listed and described down below.

4.1 Tensor

According to the TensorFlow website, the definition of a tensor is as follows:

"A tensor is a generalization of vectors and matrices to potentially higher dimensions" [31]

TensorFlow uses tensors as the primary object in operations. It is defined by shape and datatype. The shape describes the amount of dimensions of the tensor and their sizes. The number of dimensions is called "Rank" but has other synonyms such as "n-dimension" or "degree". The elements of each dimension in the tensor is called the 'shape' of a tensor[31].

4.2 TensorFlow.js API

The Ops API is the API for low-level linear algebra operations that provide operations such as matrix multiplication and Tensor addition.

The layers API provides the building blocks and practices on a high level, focusing on neural networks. It is modeled after the TensorFlow Python namespace *tf.keras* which is found in the Keras API. The essence of Keras API is how the user can build a model with pre-defined layers containing default parameters [2].

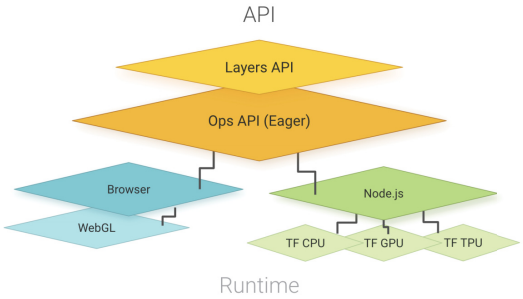

Figure 1. Overview of the TensorFlow.js architecture

4.3 Model definitions

Supervised learning is one type of supervision that is given to the model during training. You provide labels in the training which contain the wanted outcome. This is the typical learning task for classification.

Another definition of a machine learning model is the incremental learning and whether it is possible or not. Batch learning is an option for incremental, which needs to be trained with all available data which is usually done offline due to it's performance on computer resources. The procedure for Batch learning is training a model which is then deployed to production without further training. To train this model again requires training again from scratch. As the training could take hours, computing power could become costly when training on a regular basis and perhaps impossible with large amounts of data.

This would be more optimal with incremental learning. This is also called Online Learning which is efficient with continuous data flows by being adaptable to change and learn from new data continuously [9, Chapter 1].

4.4 Batch size

Batch size is the number of training examples that is used in one iteration of training a model [13]. Batch Gradient Descent is a learning algorithm where the batch size is equal to the amount of training examples. Stochastic Gradient Descent is when the batch size is defined as one training example. Mini-Batch Gradient Descent is between the two prior values, where some popular examples are 32, 64 and 128 [4].

4.5 Word embeddings

Working with text as input into a machine learning model requires a conversion from strings to numbers as the input are forms of vectors.

Word embedding represent words in a dense vector with floating values that are trainable rather than manually created. These vectors can have higher dimensions depending on the datasets of words. Similarity between words can be determined as the words have similar data structure of floating points [32].

4.6 Loss function

The definition of a loss function is described in TensorFlow.js API documentation as a function with the objective to minimise loss. The goal of the function is to give a number for "how wrong" the model's prediction was. The functions compute the loss for each batch and then updates its values to next batch. [29]

4.6.1 Mean squared error

"In statistics, the mean squared error (MSE) or mean squared deviation (MSD) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value." [17]. Figure 4.1 explains MSE in mathematical terms.

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y_i})^2$$

* n is the number of data points
* Y_i represents observed values
* $\hat{Y_i}$ represents predicted values

Figure 4.1: MSE explanation from study.com [5]

4.6.2 Categorical cross-entropy

"Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events." [3]. Cross-entropy is used widely among classification models. The model can estimate the probability for each class label, which can be used to calculate the difference of two probability distributions [3].

4.7 Optimizer

An optimizer in TensorFlow.js decides how much the model should change their parameters on each given model prediction. There are a lot of different types of predefined optimizers in TensorFlow.js, for example "sgd", "adamax" and "adam" [29]. In this report and in the scope for this experiment the "Adam"-optimizer was used by recommendation from a report by David Mack [16]. In figure 4.2 there is a visual explanation of how the Adam optimizer works with parameters in each iteration and how it selects their weights when modifying the model.

Figure 4.2: Adam explanation from Akira.ai [1]

4.8 Activation functions

Activation functions has two primary purposes, to help the model with interaction effects and help with non-linear effects. Interaction effects can be described as when variable A affects the prediction differently based on variable B. I.e a body weight having a positive weight for indicating diabetes on tall people but the opposite effect on short people, which means weight and height has an interaction effect. The effect of non-linear effects means that when placing a variable on the x-axis and the predictions on the y-axis, the graph will not display a straight line [6].

Sigmoid

A sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point [18]. Down below in figure 4.3 is the "Sigmoid" activation function graph, which is one of the activation functions that are used in the experiment.

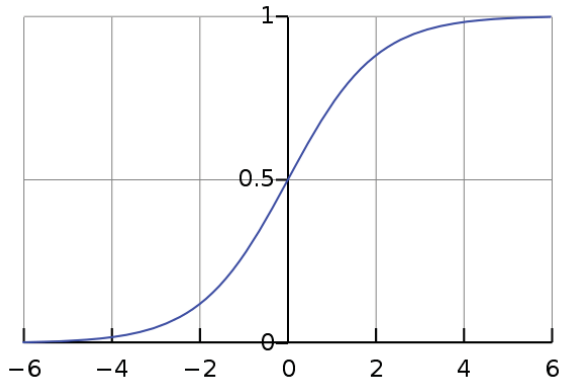

Figure 4.3: Sigmoid activation function [27]

Softmax

Softmax calculates the probabilities in decimal for each class where there are multiple classes, which must be summed up to 1 [22]. The softmax function is visually presented in figure 4.4 down below.

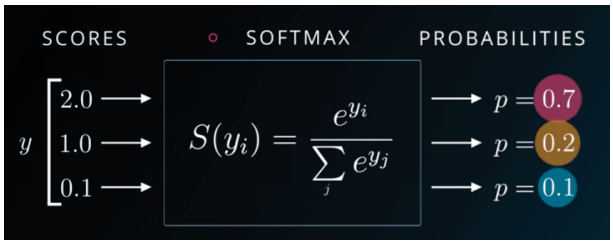

Figure 4.4: Softmax activation function [26]

Relu

Relu stands for Rectified Linear Unit, which is the most commonly used activation function for deep learning models. It returns 0 if the input is negative and returns any positive values back. Relu is widely used due to its good performance in most applications [6]. Relu is visually explained in the figure 4.5 down below.

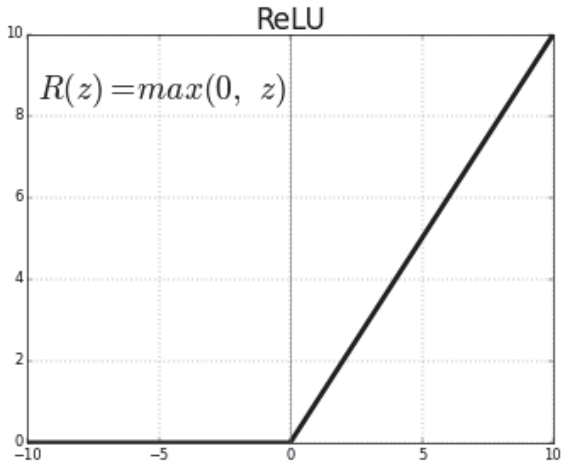

Figure 4.5: Relu activation function [25]

4.9 Sequential model

A sequential model is defined as a linear stack of layers. The first layer needs to have a defined input shape for the model to know what input shape to expect, where the other layers will do this automatically. To begin training, configuration of the learning process is required which includes an optimizer, a loss function, layers and their size and a list of metrics [10]. Metrics is the function to calculate the model performance, which can be one of the existing loss functions although the results will not affect the training [15].

According to TensorFlow, sequential model is the most common type for a model and is the type that is being used in this experiment [20].

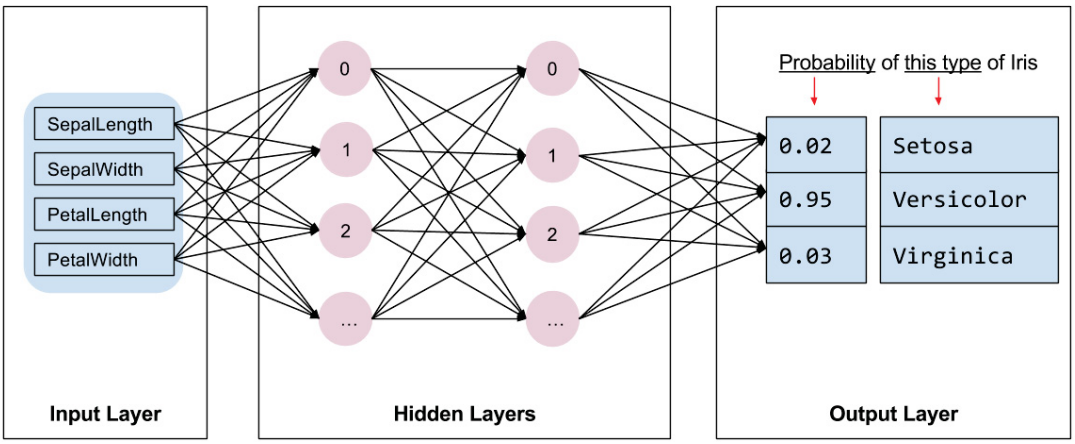

Figure 4.6: Image of custom network from TensorFlow guide [30]

4.9.1 Layer

The highest-level building block in deep learning is a layer. It works as a container that receives weighted inputs which is transformed with mostly non-linear functions. The values are then passed to the next layer as output. The layers between the input and output layer are called hidden layers [8].

Chapter 5

Results

The models are categorised in three groups by their respective amount of articles used in training. The groups are as follows: 1000(figure 5.1), 2500(figure 5.2) and 5000(figure 5.3) articles. Each table is tested with the two loss functions Categorical crossentropy and Mean squared error. The tests then have an incremental increase of units in the first layer. Units is the shape of the data that each layer outputs to the next layer. This range from 256, 128, 64 and 32 units with the exception of the first two rows in each table where the value for the loss function value was tested with 0.04 and 0.06. The remaining models are tested on the loss function value 0.02. The other main difference for each test are the different activation functions used in the layers which are Sigmoid, Relu and Softmax, as described in section 4.28.

All models are tested with 1000 epochs and 512 batch size. The strikeout rows are the models that were not able to create a reasonable prediction and the bold rows are the best models of each combined loss function and activation functions.

The measurement unit that is used for evaluation is the last column which subtract the average correctness with the average incorrectness of the predictions. This will be referred as percentage in future references.

5.1 Example of one row in the model sheets

One example of a model in the result look like this:

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
1000	1000	Adam 0.04	CatEnt	Sigmoid, Sigmoid, S	256, 9, 3	80	80	90	1	2	83,33	1,5	81,83

The first column displays the amount of articles used during training, which in this case is 1000. The settings of the model are then displayed in the five columns after in the order of: Amount of training iterations, aka epochs, optimizer function, loss

function, activation functions and lastly the units used for each respective layer. The following colored columns are the results from the testing. The first three colored columns are the results from sources in order as New York times, Huffington post and data from our training dataset. The two following columns are the incorrectness calculated on Wellness and Entertainment categories. The last three columns are calculated values based on previous inputs which include the average correctness, the average incorrectness and lastly the result of correctness minus the incorrectness.

5.2 Phase 1: Base testing models

The figures shown in this section are the base models for each amount of articles used that are trained with incremental numbers of units and different activation and loss functions.

5.2.1 1000 articles

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
~~1000~~	~~1000~~	~~Adam 0.06~~	~~CatEnt~~	~~Sigmoid, Sigmoid, S~~	~~256, 9, 3~~	0	0	0	0	0	0	0	0
1000	1000	Adam 0.04	CatEnt	Sigmoid, Sigmoid, S	256, 9, 3	80	80	90	1	2	83,33	1,5	81,83
1000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	256, 9, 3	80	82	92	2	2	84,67	2	82,67
1000	**1000**	**Adam 0.02**	**CatEnt**	**Sigmoid, Sigmoid,**	**128, 9, 3**	**80**	**80**	**94**	**1**	**2**	**84,67**	**1,5**	**83,17**
1000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	64, 9, 3	77	79	92	0	2	82,67	1	81,67
1000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	32, 9, 3	75	81	93	0	3	83	1,5	81,5
1000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	256, 9, 3	79	79	96	1	2	84,67	1,5	83,17
1000	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Softma**	**128, 9, 3**	**79**	**81**	**94**	**1**	**2**	**84,67**	**1,5**	**83,17**
1000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	64, 9, 3	76	77	93	2	2	82	2	80
1000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	32, 9, 3	74	78	90	3	2	80,67	2,5	78,17
1000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	256, 9, 3	77	75	90	0	3	80,67	1,5	79,17
1000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	128, 9, 3	71	72	93	1	2	78,67	1,5	77,17
1000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	64, 9, 3	73	79	94	2	2	82	2	80
1000	**1000**	**Adam 0.02**	**MSE**	**Sigmoid, Sigmoid,**	**32, 9, 3**	**76**	**77**	**95**	**0**	**2**	**82,67**	**1**	**81,67**
1000	**1000**	**Adam 0.02**	**MSE**	**Relu. Relu. Softma**	**256. 9. 3**	**85**	**82**	**93**	**1**	**4**	**86.67**	**2.5**	**84.17**
1000	1000	Adam 0.02	MSE	Relu, Relu, Softma	128, 9, 3	78	82	92	3	1	84	2	82
1000	1000	Adam 0.02	MSE	Relu, Relu, Softma	64, 9, 3	81	83	91	0	4	85	2	83
1000	1000	Adam 0.02	MSE	Relu, Relu, Softma	32, 9, 3	73	80	96	0	2	83	1	82

Figure 5.1: Table of models trained with 1000 articles

The results from figure 5.1 shows that Categorical cross-entropy had the higher average prediction percentage with 128 units for the two top models. However, MSE had one of the highest prediction percentage in the whole set of models, as 84.17. MSE performed better on low or high amount of units as the top 2 models for Sigmoid and Relu were 32 and 256 units respectively, while Categorical cross-entropy resulted in similar top models.

5.2.2 2500 articles

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
~~2500~~	~~1000~~	~~Adam 0.06~~	~~CatEnt~~	~~Sigmoid, Sigmoid, S~~	~~256, 9, 3~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
2500	1000	Adam 0.04	CatEnt	Sigmoid, Sigmoid, S	256, 9, 3	81	82	95	2	2	86	2	84
2500	**1000**	**Adam 0.02**	**CatEnt**	**Sigmoid, Sigmoid,**	**256, 9, 3**	**85**	**85**	**92**	**1**	**2**	**87,33**	**1,5**	**85,83**
2500	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	128, 9, 3	81	79	93	1	4	84,33	2,5	81,83
2500	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	64, 9, 3	77	77	90	2	3	81,33	2,5	78,83
2500	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	32, 9, 3	74	75	92	1	2	80,33	1,5	78,83
2500	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	256, 9, 3	78	79	94	0	2	83,67	1	82,67
2500	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Softma**	**128, 9, 3**	**81**	**83**	**94**	**1**	**2**	**86**	**1,5**	**84,5**
2500	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	64, 9, 3	80	78	91	0	2	83	1	82
2500	2500	Adam 0.02	CatEnt	Relu, Relu, Softma	32,9, 3	81	80	93	1	3	84,67	2	82,67
2500	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	256, 9, 3	79	82	93	1	3	84,67	2	82,67
2500	**1000**	**Adam 0.02**	**MSE**	**Sigmoid, Sigmoid,**	**128, 9, 3**	**82**	**80**	**94**	**0**	**2**	**85,33**	**1**	**84,33**
2500	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	64, 9, 3	76	80	95	1	3	83,67	2	81,67
2500	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	32, 9, 3	74	76	95	1	3	81,67	2	79,67
2500	1000	Adam 0.02	MSE	Relu, Relu, Softma	256, 9, 3	77	77	92	0	3	82	1,5	80,5
2500	1000	Adam 0.02	MSE	Relu, Relu, Softma	128, 9, 3	78	75	95	1	1	82,67	1	81,67
2500	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Softma**	**64, 9, 3**	**81**	**82**	**91**	**2**	**1**	**84,67**	**1,5**	**83,17**
2500	1000	Adam 0.02	MSE	Relu, Relu, Softma	32, 9, 3	72	80	93	0	3	81,67	1,5	80,17

Figure 5.2: Table of models trained with 2500 articles

The models shown on figure 5.2 shows that Sigmoid top models performs better with both Categorical cross-entropy and MSE with double the units as Relu top models. Which shows that Sigmoid performed better with higher units specified while Relu performed better on low units, although only approximately 1 percentage unit away from the Sigmoid competitors.

5.2.3 5000 articles

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
~~5000~~	~~1000~~	~~Adam 0.06~~	~~CatEnt~~	~~Sigmoid, Sigmoid, S~~	~~256, 9, 3~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
~~5000~~	~~1000~~	~~Adam 0.04~~	~~CatEnt~~	~~Sigmoid, Sigmoid, S~~	~~256, 9, 3~~	~~100~~	~~100~~	~~100~~	~~100~~	~~100~~	~~100~~	~~100~~	~~0~~
5000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	256, 9, 3	75	73	90	2	2	79,33	2	77,33
5000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	128, 9, 3	76	74	87	3	2	79	2,5	76,5
5000	**1000**	**Adam 0.02**	**CatEnt**	**Sigmoid, Sigmoid,**	**64, 9, 3**	**82**	**75**	**88**	**4**	**1**	**81,67**	**2,5**	**79,17**
5000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	32, 9, 3	79	74	87	6	3	80	4,5	75,5
5000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	256, 9, 3	77	81	93	4	3	83,67	3,5	80,17
5000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	128, 9, 3	80	83	85	4	2	82,67	3	79,67
5000	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Softma**	**64, 9, 3**	**80**	**78**	**90**	**3**	**2**	**82,67**	**2,5**	**80,17**
5000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	32,9, 3	79	78	88	4	3	81,67	3,5	78,17
5000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	256, 9, 3	77	79	85	2	2	80,33	2	78,33
5000	**1000**	**Adam 0.02**	**MSE**	**Sigmoid, Sigmoid,**	**128, 9, 3**	**84**	**84**	**91**	**2**	**4**	**86,33**	**3**	**83,33**
5000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	64, 9, 3	81	79	91	3	2	83,67	2,5	81,17
5000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	32, 9, 3	79	76	92	3	1	82,33	2	80,33
5000	1000	Adam 0.02	MSE	Relu, Relu, Softma	256, 9, 3	79	80	87	2	2	82	2	80
5000	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Softma**	**128, 9, 3**	**79**	**82**	**93**	**3**	**1**	**84,67**	**2**	**82,67**
5000	1000	Adam 0.02	MSE	Relu, Relu, Softma	64, 9, 3	80	79	92	7	3	83,67	5	78,67
5000	1000	Adam 0.02	MSE	Relu, Relu, Softma	32, 9, 3	78	74	89	2	0	80,33	1	79,33

Figure 5.3: Table of models trained with 5000 articles

The figure 5.3 shows that the top two models for Categorical cross-entropy both had 64 units for Sigmoid and Relu, while top models for MSE had 128 units respectively. The majority of models displayed a high percentage of incorrectness with values from 2-5 percentage, which resulted in lower average correctness percentage ranging from lowest as 79.17 to the highest as 83.33. This set shows

the model with the lowest accuracy of the whole training phase, which had an accuracy of 75.5%. This model was trained with Sigmoid as activation function and Categorical cross-entropy as loss function. There were also two models that were unable to make a reasonable prediction.

5.3 Phase 2: adjustment of top models

This section presents the models that are founded on the base models in section 5.2 and with the addition of two different approaches for these models. The first approach is to add an additional layer and the second approach increases the first layer to 512 units.

This approach was something that the authors decided was important to see different results with the best models from previous base tests and extend these with more layers as well as a multiplicative increase of units. This was done in order to see if changing these settings would have an impact on the model accuracy.

5.3.1 Top 12 models for Sigmoid and Relu

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
Best 1k models	***Top 2 Sig and 2 relu for MSE and CatEnt***												
1000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	128, 9, 3	80	80	94	1	2	84,67	1,5	83,17
1000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	128, 9, 3	79	81	94	1	2	84,67	1,5	83,17
1000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid, S	32, 9, 3	76	77	95	0	2	82,67	1	81,67
1000	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Softma**	**256, 9, 3**	**85**	**82**	**93**	**1**	**4**	**86,67**	**2,5**	**84,17**
Best 2.5k models													
2500	**1000**	**Adam 0.02**	**CatEnt**	**Sigmoid, Sigmoid,**	**256, 9, 3**	**85**	**85**	**92**	**1**	**2**	**87,33**	**1,5**	**85,83**
2500	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	128, 9, 3	81	83	94	1	2	86	1,5	84,5
2500	**1000**	**Adam 0.02**	**MSE**	**Sigmoid, Sigmoid,**	**128, 9, 3**	**82**	**80**	**94**	**0**	**2**	**85,33**	**1**	**84,33**
2500	1000	Adam 0.02	MSE	Relu, Relu, Softma	64, 9, 3	81	82	91	2	1	84,67	1,5	83,17
Best 5k models													
5000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid, S	64, 9, 3	82	75	88	4	1	81,67	2,5	79,17
5000	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Softma**	**64, 9, 3**	**80**	**78**	**90**	**3**	**2**	**82,67**	**2,5**	**80,17**
5000	**1000**	**Adam 0.02**	**MSE**	**Sigmoid, Sigmoid,**	**128, 9, 3**	**84**	**84**	**91**	**2**	**4**	**86,33**	**3**	**83,33**
5000	1000	Adam 0.02	MSE	Relu, Relu, Softma	128, 9, 3	79	82	93	3	1	84,67	2	82,67

Figure 5.4: Table of top 12 models for each training set

Figure 5.4 shows the 12 models that performed top Sigmoid and Relu models based on loss functions, collected from the 1000, 2500 and 5000 article tables respectively. The results show that the 2500 articles set has the model with the highest prediction percentage compared to the other two training sets, as well as the highest average prediction percentage.

5.3.2 Top 12 models with additional layer

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
1k models	**1 additional layer**												
1000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid,	256,128, 9,	74	75	91	1	2	**80**	**1,5**	**78,5**
1000	**1000**	**Adam 0.02**	**CatEnt**	**Relu relu, relu, so**	**256, 128, 9,**	**78**	**77**	**92**	**2**	**2**	**82,33**	**2**	**80,33**
1000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid,	64, 32, 9, 3	78	85	92	3	7	**85**	**5**	**80**
1000	**1000**	**Adam 0.02**	**MSE**	**Relu relu, relu, so**	**256, 128, 9,**	**84**	**86**	**96**	**1**	**2**	**88,67**	**1,5**	**87,17**
2.5k													
2500	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid,	256, 128, 9,	77	80	93	1	2	**83,33**	**1,5**	**81,83**
2500	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Relu, S**	**256, 128, 9**	**79**	**76**	**92**	**0**	**3**	**82,33**	**1,5**	**80,83**
2500	1000	Adam 0.02	MSE	Sigmoid, Sigmoid,	256, 128, 9,	82	82	93	0	3	85,67	1,5	84,17
2500	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Relu S**	**128, 64, 9, 3**	**80**	**86**	**95**	**2**	**1**	**87**	**1,5**	**85,5**
5k													
5000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid,	128, 64, 9, 3	80	77	91	3	2	82,67	2,5	80,17
5000	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Relu S**	**128, 64, 9, 3**	**81**	**81**	**89**	**5**	**1**	**83,67**	**3**	**80,67**
5000	1000	Adam 0.02	MSE	Sigmoid, Sigmoid,	256, 128, 9,	80	79	90	0	2	83	1	82
5000	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Relu S**	**256, 128, 9,**	**80**	**88**	**97**	**7**	**4**	**88,33**	**5,5**	**82,83**

Figure 5.5: Table of top 12 models with an additional layer

Figure 5.5 is the next step from figure 5.4, adding an additional layer at the start with the same parameters as the previous models. The model that stands out here is the model from the 1000 articles set with the prediction result of 87.17 percentage, which is the highest prediction achieved in the whole testing phase. The correctness of all models in this table are all above 80 percent although the incorrect predictions ranged from 1 to 5.5 percent, which resulted in a lower final prediction percentage for some of the models.

5.3.3 Top 12 models with first layer as 512 units

Articles per cate	Epochs	Optimizer	Loss functi	Activation functio	Units	NYT	HP	Politics	Wellness	Entertainment	AVG politics	AVG incorrect	Correct-in correct
3 layers, increasing L1 to 512													
Best 1k models	***Top 2 Sig and 2 relu for MSE and CatEnt***												
1000	1000	Adam 0.02	CatEnt	Sigmoid, Sigmoid,	512, 128, 3	77	80	91	0	3	82,67	1,5	81,17
1000	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Softma**	**512, 128, 3**	**79**	**79**	**96**	**3**	**2**	**84,67**	**2,5**	**82,17**
~~1000~~	~~1000~~	~~Adam 0.02~~	~~MSE~~	~~Sigmoid, Sigmoid,~~	~~512, 32, 3~~	~~100~~	~~100~~	~~100~~	~~100~~	~~100~~	~~100~~	~~100~~	~~0~~
1000	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Softma**	**512, 256, 3**	**84**	**86**	**94**	**2**	**4**	**88**	**3**	**85**
Best 2.5k models													
~~2500~~	~~1000~~	~~Adam 0.02~~	~~CatEnt~~	~~Sigmoid, Sigmoid,~~	~~512, 256, 3~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
2500	**1000**	**Adam 0.02**	**CatEnt**	**Relu, Relu, Softma**	**512, 128, 3**	**79**	**76**	**92**	**0**	**2**	**82,33**	**1**	**81,33**
2500	**1000**	**Adam 0.02**	**MSE**	**Sigmoid, Sigmoid,**	**512, 128, 3**	**81**	**86**	**94**	**1**	**3**	**87**	**2**	**85**
2500	1000	Adam 0.02	MSE	Relu, Relu, Softma	512, 64, 3	81	83	90	1	1	84,67	1	83,67
Best 5k models													
5000	**1000**	**Adam 0.02**	**CatEnt**	**Sigmoid, Sigmoid,**	**512, 64, 3**	**81**	**79**	**89**	**5**	**3**	**83**	**4**	**79**
5000	1000	Adam 0.02	CatEnt	Relu, Relu, Softma	512, 64, 3	79	71	87	2	1	79	1,5	77,5
~~5000~~	~~1000~~	~~Adam 0.02~~	~~MSE~~	~~Sigmoid, Sigmoid,~~	~~512, 128, 3~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
5000	**1000**	**Adam 0.02**	**MSE**	**Relu, Relu, Softma**	**512,128, 3**	**93**	**95**	**99**	**8**	**12**	**95,67**	**10**	**85,67**

Figure 5.6: Table of top 12 models with 512 units as first layer

Figure 5.6 has increased the first layer's units to 512, which resulted in one model in each set of models unable to make a prediction. This also had an effect on the incorrect percentage, which ranged from 1.5 to 10 percent.

Chapter 6

Analysis

6.1 RQ1

Based on the authors experiences from the experiment and the results of the literature review, the crucial parts in order to train a text classification model are the following:

- Optimizer function
- Loss function
- Batch size
- Activation functions
- Training data and test data with labels
- Normalise function
- Shapes of layers
- Computing power

The first stage of developing a text classification model using TensorFlow.js is the collection of data and adjusting this data into a Tensor so that the model can take this as an input. The authors used the Universal Sentence Encoder which is a TensorFlow model that works as word embedding but it is trained on word sequences rather than each individual word representation.

The result from the experiment shows that collecting available datasets online is an efficient way to access enough training and test data. Although it requires the data to be verified or tested beforehand. A risk is that the collected data contains undesirable content such as text from adverts or pay walls. This risk applies for web scraping as well but with an already existing online dataset, this risk could be lower due to already verified and tested data.

Incidentally, the collected dataset for the experiment was difficult to handle as input. This required more work with the preparations for the data by separating categories into individual files as well as functionality to automate the collection and shuffling of this data before training.

The experiment used a normalise function to remove capitalised letters that could decrease the accuracy for the model. E.g the model can think that "democracy" and "Democracy" are different words if the data is not normalised.

The model used in the experiment is built as a sequential model (See section 4.2.5), which is a linear layout for the model.

Configuring the model as described in section 4.9 includes the following settings: optimizer, loss function, layers and their sizes and the metrics used during training. The metric used in the experiment was accuracy, which displayed a simple percentage for each class that was predicted. As this value did not correspond to the testing described in section 5.1, The authors recommends to not evaluate the model solely by the accuracy during training.

6.2 RQ2

To categorise articles with text classification in this project, the authors limited the scope to two different loss functions, Mean squared error and Categorical cross-entropy (See section 4.2.6).

The accuracy from the testing results range from 75.5% to 87.17% which takes into account happy path predictions as well as wrong predictions as described in section 5.1. The accuracy result from the tests against NYT and HP sources shows an average on 80% and 79% correct predictions respectively. While the results from the same dataset as the training data returned an average accuracy of 92. This shows that the testing results have higher accuracy on the data from the dataset as the training batch.

The highest model with 87.17% accuracy had an increase of 3 percent points from the equivalent base model. Which showed that the addition of a layer for these specific settings had a positive outcome.

6.3 RQ3

When the authors performed tests of models with different amount of articles, results did not show as big of a difference as expected, one hypothesis is that quantity does not matter as much as a bigger variety of different articles. A lot of articles with the same category, from the same journalists could possibly have generated less accuracy for different test data.

This is hard to actually test in a big scale without a lot of data from different sources and a lot of computing power. Two things that makes this especially hard for this thesis is that there is a deadline and there is a limited time and amount of computing power. One hypothesis is that the system have reached a threshold where increasing the amount of articles do not increase the accuracy as much as in the beginning compared to lower amounts. This curve could be exponential but is time consuming to evaluate, which could be researched in future work.

Evaluating the models can be difficult as reproducing a model with same structure will result in different scores in our evaluation metrics. This generates a kind of randomness in the creation of models and finding the perfect model.

The Universal Sentence Encoder was used for the transformation of input text to vector format, which only accepts smaller paragraphs. The hypothesis is that having a large amount of iterations trained on with short texts will give a good complexity that can probably predict well on texts with longer length. The idea for iterations with smaller texts is similar to the usage of smaller batch sizes as this will incrementally increase the learning rate while larger texts could create an inconsistent learning rate and result in overfitting much faster.

6.4 Amount of data

The experiment shows that in general the models based on 2500 articles performed better than models trained with 1000 or 5000 articles. The average score for the models based on 1000 articles (see section 5.1.1) were rounded to 81.44 and the models based on 2500 articles (see section 5.1.2) have an average score that were rounded to 82.06. On the other hand, models based on 5000 articles (see section 5.1.3) average score were rounded to 79.41 which shows an indication that the amount of articles does not automatically improve the accuracy of the model. In this case there is a clear path that the models actually gets worse results with too many articles.

6.5 Loss functions

A comparison between Categorical cross-entropy and Mean squared error in the block of models based on both 1000 and 2500 articles (see section 5.1.1 and 5.1.2) shows that Categorical cross-entropy got a better average score in 4 of 8 comparisons each. Therefore no general benefit can be found. Worth to take into consideration is that there is only a few percentage in difference and could possibly change with more up scaled testing. A significant difference in the comparison between models in the 5000 articles block (see section 5.1.3) were noticed. Only 2 of 8 comparisons were better than Mean squared error.

6.6 Activation functions

The models were trained in the first phase with two sets of activation functions, one with three layers of the sigmoid activation function and one set of two relu and one softmax activation function. The two sets of layers are displayed below.

[sigmoid sigmoid, sigmoid] and [relu, relu, softmax]

6.6.1 Results of comparsions with Categorical crossentropy

In the block of 1000 articles (see section 5.1.1) the models using sigmoid got better average correctness in 2 of 3 comparisons and in the fourth comparison it was a draw, take in consideration that the results are rounded.

The results in the block of 2500 (see section 5.1.2) shows that in only 1 of 4 comparisons the sigmoid function won over the relu, relu, softmax models. This shows that the sigmoid function was less effective with more amounts of articles.

The results of the comparisons of models with 5000 articles (see section 5.1.3) shows that the models using sigmoid did not perform better in any of the 4 comparisons.

All these comparisons together shows that the sigmoid layers gets less effective with more articles in the training set.

6.6.2 Results of comparisons with Mean squared error

The comparison between the two sets of layers shows that sigmoid is in this case more effective with higher amount of articles together with Mean squared error. This is the opposite of what happened with Categorical crossentropy. Sigmoid performed better in 0 of 4 comparisons with 1000 articles and 2 of 4 times with 2500 articles. Finally, sigmoid performed better in 3 of 4 comparisons with 5000 articles.

Chapter 7

Validity threats

In some cases when the authors did overwrite a model by mistake and had to train a new model with same parameters, the result were different. This means that result can be different from time to time when training a machine learning model, even with exactly the same parameters, data and options.

Due to the global pandemic of COVID-19 this experiment took an unexpected turn. The testing data gathered in April 2020 mostly contain articles about COVID-19 and this could probably have an effect on the results of the correctness for those tests. Specially because the training data was collected and written before the outbreak of COVID-19. Results show a clear difference between the new articles and the old ones that were written before COVID-19. Therefore an assumption can be made that the model can not handle extraordinary situations like this as good as "normal" news. The results are still quite good and impressed the authors during the experiment. To validate the models accuracy it would be of interest to have another testing set with new articles done when the COVID-19 pandemic is over.

Chapter 8

Conclusion

There are multiple parameters and mathematical functions behind a working machine learning model and it requires a lot of research in order to determine which algorithm or mathematical function that fits the current case the best. To achieve a good result it is preferable to train a large amount of different models to compare with. The crucial parts in building a machine learning model that can classify articles in TensorFlow.js are the following: optimizer function, loss function, batch size, activation functions, training and test data, normalise function, shapes of layers and computing power. These results can help developers that have no prior experience in machine learning or TensorFlow. As the paper describes basics in both areas and supplies testing examples, other developers can start by using similar settings or try different settings and compare to the empirical result of this paper.

The results of the experiment in RQ2 were better than the expected but should be analysed with care due to the difference of data with and without COVID-19 articles. The final results show that the best performing model was trained with 1000 articles and got an accuracy of 87.17 percentage.

The value of optimising the parameters and options for a machine learning model are crucial for improving the accuracy of the model and can vary a lot. In this experiment the results shows that the accuracy can differ between 75.5 to 87.17 and it is possible that the model can be more accurate with more optimising as well.

Chapter 9

Future Work

More data in general could possibly increase the amount of categories that the model is able to predict and also create diversity and complexity in the area that the model can predict.

In the future it would be interesting to test and investigate different normalise functions and if that would decrease or increase the accuracy of the models. In this experiment a simple normalise function that convert all characters to lower case were used. In future work it would be interesting to try normalise functions that includes the following:

- Removing Stop words
- Standardisation
- Removing Acronyms
- Removing Contractions
- Rewrite abbreviations

The gathering of "new" articles was performed during the global pandemic COVID-19. The testing datasets contained a majority of articles about COVID-19. The training dataset contained data written before the outbreak of COVID-19 and have possibly affected the test results. Future work would be to reproduce the same tests with new data after the pandemic is over or include COVID-19 data in the training datasets.

As previously said in the analysis, the models has different results when it is retrained from start. This could be interesting to test by evaluating models based on repeated tests performed on the same model.

References

[1] "Adam optimization". In: *Akira AI* (2020). URL: https://www.akira.ai/glossary/adam-optimization/ (visited on 05/06/2020).

[2] Smilkov et al. "TensorFlow.JS: MACHINE LEARNING FOR THE WEB AND BEYOND". In: *Cornell University* (2019). URL: https://arxiv.org/pdf/1901.05350.pdf (visited on 03/19/2020). (last update: 28/02/2019).

[3] Jason Brownlee. "A Gentle Introduction to Cross-Entropy for Machine Learning". In: *Machine Learning Mastery* (2019). URL: https://machinelearningmastery.com/cross-entropy-for-machine-learning/ (visited on 05/07/2020). (last update: 20/12/2019).

[4] Jason Brownlee. "Difference Between a Batch and an Epoch in a Neural Network". In: *Machine learning mastery* (2019). URL: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ (visited on 04/22/2020). (last update: 26/10/2019).

[5] Bob Bruner. "Mean Squared Error: Deifinition examples". In: *Study.com* (2019). URL: https://study.com/academy/lesson/estimation-of-r-squared-variance-of-epsilon-definition-examples.html (visited on 05/07/2020). (published: 11/7/2019).

[6] DanB. "Rectified Linear Units (ReLU) in Deep Learning". In: *Kaggle* (2018). URL: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning (visited on 05/06/2020). (published: 2018).

[7] "Data set". In: *Wikipedia* (2019). URL: https://en.wikipedia.org/wiki/Data_set (visited on 03/26/2020). (last update: 08/12/2019).

[8] Tim Dettmers. "Deep Learning in a Nutshell: Core Concepts". In: *Nvidia Developer Blog* (2015). URL: https://devblogs.nvidia.com/deep-learning-nutshell-core-concepts/ (visited on 04/24/2020). (published: 3/11/2015).

[9] Aurélien Géron. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition". In: *Oreilly* (2020). URL: https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch01.html#landscape_chapter (visited on 03/26/2020).

[10] "Getting started with the Keras Sequential model". In: *Keras* (2020). URL: https://keras.io/getting-started/sequential-model-guide/ (visited on 04/24/2020). (last update: 12/04/2020).

[11] Google. *Machine learning - text classification guide*. 2018. URL: https://developers.google.com/machine-learning/guides/text-classification/ (visited on 02/28/2020). (last update: 01/10/2018).

[12] Google. *TensorFlow official documentation*. URL: https://www.TensorFlow.org/ (visited on 02/28/2020).

[13] google. *Machine Learning Glossary*. 2020. URL: https://developers.google.com/machine-learning/glossary#batch_size (visited on 04/22/2020). (last update: 18/04/2020).

[14] "Hyperparameter". In: *Wikipedia* (2020). URL: https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning) (visited on 04/17/2020). (last update: 06/04/2020).

[15] keras. "Usage of metrics". In: *Keras* (2020). URL: https://keras.io/metrics/ (visited on 04/24/2020).

[16] David Mack. "How to pick the best learning rate for your machine learning project". In: *Medium* (2018). URL: https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2 (visited on 05/14/2020). (published: 9/4/2018).

[17] "Mean squared error". In: *Wikipedia* (2020). URL: https://en.wikipedia.org/wiki/Mean_squared_error (visited on 05/07/2020). (last update: 1/5/2020).

[18] Francisco Mira J. (Jose); Sandoval. "From natural to artificial neural computation : International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995 : proceedings". In: *Internet Archive* (1995). URL: https://archive.org/details/fromnaturaltoart1995inte/page/235 (visited on 05/06/2020). (published: 1995).

[19] Rishabh Misra. "Kaggle dataset". In: *Kaggle* (2018). URL: https://www.kaggle.com/rmisra/news-category-dataset (visited on 02/28/2020). (last update: 02/12/2018).

[20] "Models and layers". In: *TensorFlow* (2020). URL: https://www.tensorflow.org/js/guide/models_and_layers (visited on 04/24/2020). (last update: 13/4/2020).

[21] Mulesoft. "What is an API?" In: *Mulesoft* (2020). URL: https://www.mulesoft.com/resources/api/what-is-an-api (visited on 03/26/2020).

[22] "Multi-Class Neural Networks: Softmax". In: *Google* (2020). URL: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax (visited on 05/06/2020). (last update: 17/3/2020).

[23] Henryk Maciejewski Piotr Semberecki. "Deep Learning methods for Subject Text Classification of Articles". In: *IEEE* (2017). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8104565&casa_token=g00Lie2BbzQAAAAA:vvD6bbLEz8dc6131Ynvw46eoc3hfjv5YNic2dA-58sQn5YtFpiQxGxmVEgyORPWonATHS-pA5zg (visited on 05/13/2020). (published: 13/11/2017).

[24] R.Wongso et al. "News Article Text Classification in Indonesian Language". In: *ScienceDirect* (2017). URL: https://www.sciencedirect.com/science/article/pii/S1877050917320872 (visited on 02/28/2020). (published: 14/10/2017).

[25] Kanchan Sarkar. "ReLU : Not a Differentiable Function: Why used in Gradient Based Optimization? and Other Generalizations of ReLU." In: *Medium* (2018). URL: https://medium.com/@kanchansarkar/relu-not-a-differentiable-function-why-used-in-gradient-based-optimization-7fef3a4cecec (visited on 05/07/2020). (published: 31/5/2018).

[26] Sefik Ilkin Serengil. "Softmax as a Neural Networks Activation Function". In: *Sefik Ilkin Serengil* (2017). URL: https://sefiks.com/2017/11/08/softmax-as-a-neural-networks-activation-function/ (visited on 05/07/2020). (published: 8/11/2017).

[27] "Sigmoid function". In: *Wikipedia* (2020). URL: https://en.wikipedia.org/wiki/Sigmoid_function (visited on 05/07/2020). (last update: 4/5/2020).

[28] Stackoverflow. *Questions tagged tensorflow*. 2020. URL: https://stackoverflow.com/questions/tagged/tensorflow (visited on 05/14/2020).

[29] TensorFlow. *Loss function*. 2020. URL: https://www.TensorFlow.org/js/guide/train_models#optimizer_loss_and_metric (visited on 03/13/2020). (last update: 21/02/2020).

[30] TensorFlow. *Picture of custom network*. URL: https://www.tensorflow.org/tutorials/customization/custom_training_walkthrough.

[31] TensorFlow. *TensorFlow tensor*. 2020. URL: https://www.TensorFlow.org/guide/tensor (visited on 03/03/2020). (last update: 24/02/2020).

[32] TensorFlow. *Word embeddings*. 2020. URL: https://www.TensorFlow.org/tutorials/text/word_embeddings (visited on 03/04/2020). (last update: 21/02/2020).

[33] "Text Classification Algorithms: A survey". In: *Cornell University* (2019). URL: https://arxiv.org/pdf/1904.08067.pdf (visited on 06/01/2020). (published: 23/04/2019).

[34] Yann LeCun Xiang Zhang Junbo Zhao. "Character-level Convolutional Networks for Text Classification". In: *NIPS Proceedings* (2015). URL: https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf (visited on 06/01/2020).

Appendix A

Project setup

To get started with a simple project in TensorFlow it is required to have Node.js installed. (Tested with v12.9.1)
To get the project up and running you just need to install the dependencies by running ‘npm i‘.

The structure of the GitHub project is as follows:

- /**my-app** has the web GUI created in React which loads in the TensorFlow model and takes one input as text.
- /**dataset** contains the Kaggle dataset which is our source of training data[19]. As well as our test datasets which the authors gathered from `www.nytimes.com` and `www.huffpost.com`.
- /**scraper** the webscraper that was used to gather test datasets.
- /**newModels** contains all the tested models that were created during the empirical experiment.

When all dependencies are installed the index.js and predict.js file will be ready to be run by "node <file>" and can train or predict headlines by changing the train/test data inputs in the file. This can also be done in the GUI by starting it with "npm start".