UNIVERSITÄT SIEGEN

# Exposé für die Masterarbeit: Clientseitiges Deep Learning durch Klassifizierung von Clickbaits

Ugur Tigu

Master-Thesis

zur Erlangung des akademischen Grades Master of Science (M.Sc.)

Studiengang Wirtschaftsinformatik

Fakultät IV - Institut für Wissensbasierte Systeme und Wissensmanagement

Universität Siegen

1. November 2020

Betreuer

Prof. Dr.-Ing. Madjid Fathi, Universität Siegen

Johannes Zenkert, Universität Siegen

**Tigu, Ugur**:
Exposé für die Masterarbeit: Clientseitiges Deep Learning durch Klassifizierung von Clickbaits / Ugur Tigu. –
Master-Thesis, Aachen: Universität Siegen, 2020. 9 Seiten.

**Tigu, Ugur**:
Exposé for the master's thesis / Ugur Tigu. –
Master Thesis, Aachen: University of Siegen, 2020. 9 pages.

## Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Universität Siegen öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Aachen, 1. November 2020

Ugur Tigu

# Abstract

***Exposé für die Masterarbeit: Clientseitiges Deep Learning durch Klassifizierung von Clickbaits***

***Exposé for the master's thesis***

# Inhaltsverzeichnis

## Kapitel 1

# Einletung

Die Programmiersprache Python dominiert, wenn es darum geht, Modelle zu entwickeln oder NLP-basierte Systeme zu entwickeln. Python bedankt seine Beliebtheit nicht nur seines relativ einfaches Syntax, sondern auch der großen Vielfalt an Bibliotheken für Deep Learning oder NLP. Klassisches Vorgehen ist dabei, zuerst ein Modell mit Daten zu trainieren und dieses in Form von Mikroservices dem Frontend anzubieten. Mobile Geräte und Browser haben aber nicht die Ressourcen, um die immer größer werdenden Modelle handzuhaben. Mit TensorFlow.js hat Google TensorFlow in den Browser und somit in die Welt von JavaScript hineingebracht. Es ist nun möglich, ein bereits trainiertes Modell in TensorFlow.js zu bringen oder ein Modell direkt in JavaScript zu entwickeln. TensorFlow.js ist die erste vollwertige Bibliothek in Industriequalität für die Erstellung neuronaler Netzwerke in JavaScript.

Sobald ein Modell für maschinelles Lernen trainiert wird, muss es irgendwo bereitgestellt werden, um Vorhersagen über reale Daten zu treffen (z. B.Klassifizierung von Bildern und Text, Erkennen von Ereignissen in Audio- oder Videostreams usw.). Ohne Bereitstellung ist das Training eines Modells nur eine Verschwendung von Rechenleistung. Es ist oft wünschenswert oder zwingend erforderlich, dass das „Irgendwo“ ein Web-Frontend ist.

Auf Desktops und Laptops ist der Webbrowser das dominierende Mittel, über die Benutzer auf Inhalte und Dienste im Internet zugreifen. Auf diese Weise verbringen Desktop- und Laptop-Benutzer die meiste Zeit mit diesen Geräten. Auch auf mobilen Geräten, können diese Seiten aufgerufen werden. Auf diese Weise erledigen Benutzer einen Großteil ihrer täglichen Arbeit, bleiben in Verbindung und unterhalten sich. Die breite Palette von Anwendungen, die im Webbrowser ausgeführt werden,

bietet umfangreiche Möglichkeiten für die Anwendung clientseitigen maschinellen Lernens. Für das mobile Frontend folgt der Webbrowser nativen mobilen Apps in Bezug auf Benutzereingriff und Zeitaufwand. Die browserbasierte Anwendung von Deep Learning bietet zusätzliche Vorteile: geringere Serverkosten, geringere Inferenzlatenz, Datenschutz, sofortige GPU Beschleunigung und sofortiger Zugriff.

TensorFlow.js oder die Modelle, die darin entwickelt werden, sind allerding kein vollwertiger Ersatz für ein „klassiches“ Modell. TensorFlow.js kann jedoch viele Probleme lösen und erreicht auch mit kleinen Modellen relativ gute und schelle Ergebnisse.

# Kapitel 2

# Vorstellung des Themas

## 2.1 Problemdefinition

Das Aufkommen von Online-Nachrichtenagenturen und die Explosion der Anzahl der Benutzer, die Nachrichten mit diesem Medium konsumieren, haben dazu geführt, dass mehrere Webseiten miteinander konkurrieren, um die Aufmerksamkeit der Benutzer zu erregen. Dies hat dazu geführt, dass Verkaufsstellen kreative Wege geschaffen haben, um Leser auf ihre Website zu locken. Eine der am häufigsten verwendeten Techniken ist die Verwendung von Clickbait-Überschriften. Diese Überschriften wurden speziell entwickelt, um das Interesse des Lesers an dem zu wecken, was versprochen wird, wenn auf den Artikel geklickt wird. Der Artikel liefert normalerweise nicht den Inhalt, den der Leser sucht.

## 2.2 Lösungsansatz

Textklassifizierung ist eine gängige Art, wie man „gute“ von „bösen“ Texten unterscheiden kann. Es ist allerdingt nicht praktisch ein großes Sprachmodell wie GPT-3 Browserkompatible zu machen. Erstens ist es völlig „overkill“ für ein solches Problem ein Sprachmodell zu benutzen und zweitens passen diese großen Modelle nicht in den Browser, da die Ladezeit unpraktisch groß ist.

Mit dieser Arbeit möchte ich ein Modell erstellen, um Clickbait Überschriften zu erkennen. Es wäre Hilfreich, wenn es ein Dienst gibt, welches eine Überschrift liest und dem Benutzer vorhersagt, ob es sich um Clickbait handelt oder nicht. So kann der Nutzer seine Zeit sparen und muss nicht auf die Seite gehen. Das Hauptprodukt

ist dabei dieser Dienst, welches ganz einfach in jede HTML-Seite importiert werden kann. Dieser Dienst muss klein, schnell und gute Ergebnisse liedern.

## 2.3 Methodologie

Der erste theoretische Teil ist mathematisch/statistsch geprägt. Der zweite theoretische Teil beschäftigt sich mit der Sprache. Es wird nicht scharf zwischen den beiden Kapiteln getrennt, da diese viele Überlappungen haben werden.

Ich werde ein Modell in TensorFlow.js entwickeln. Dieses Modell soll möglichst klein sein und schnell sein. Um mit TensorFlow.js arbeiten zu können muss der Text in Vektoren umgewandelt werden, z. B.mit einem CNN.

Es gibt bereits ein Datensatz [1] welches dafür verwendet werden kann. Das Kernproblem ist, dass Clickbaits mehr oder weniger ein Muster haben. Wichtig ist es, dieses zu erkennen und in das Modell einzuspeisen. Mit einem Versuchsaufbau soll das Modell getestet werden und analysiert werden.

Schließlich soll das Modell in die Browser-Umgebung gebracht und mittels eines minimalistischen React Frontends angeboten werden.

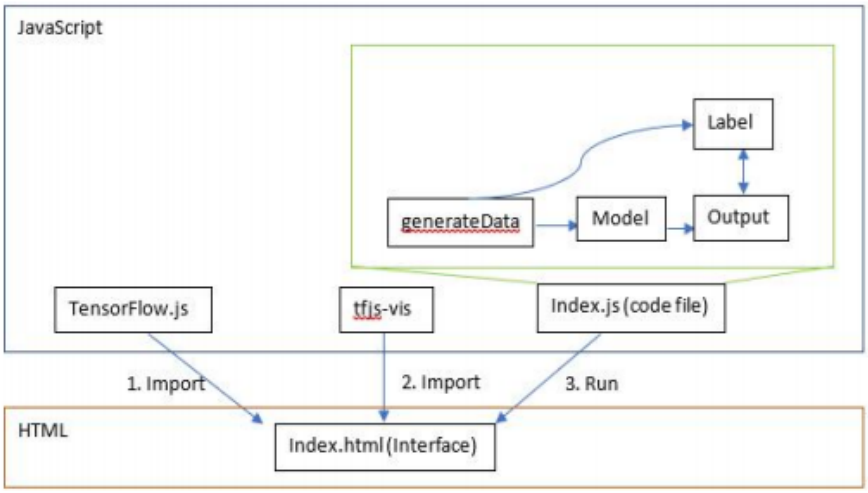


**Abbildung 2.1:** Beispiel einer Systemarchitektur entnommen aus Cho, Lee und Chung 2019. Der Client bekommt beim laden der Seite, neben dem HTML, CSS und dem JavaScript, welches für das Frontend nötig ist, ein weiteres Script. Dieser Script beantwortet die Anfragen bezüglich des ML-Modells. Die Berechnung findet also im Browser, beim Client statt.

[1] https://www.kaggle.com/atechnohazard/clickbait-classifier-using-lstm

## 2.4 Wissenschaftlicher Beitrag

- Den Stand der Technik in Bezug auf Deep Learning und Worteinbettungen zeigen
- Implementierung eines NLP-Problemes im Browser ohne zusätzliche Software oder Plugins, mit TensorFlow.js
- Optimierung des Browsers für Deep Learning
- Auswahl des optimalen Modells und der Trainingsmethode
- Produktion eines Dienstes, um Clickbait Nachrichten vorzubeugen

## 2.5 Einstiegsliteratur

- Kaur, P. Kumar und Kumaraguru 2020
- Vorakitphan, Leu und Fan 2018
- Anand, Chakraborty und Park 2017
- Gairola u. a. 2017
- V. Kumar u. a. 2018
- Glenski u. a. 2017
- Chawda u. a. 2019
- Seo o.D.
- Cho, Lee und Chung 2019
- Roberts, Hawthorne und Simon 2018
- Aggarwal und Zhai 2012
- Kowsari u. a. 2019
- Korde und Mahender 2012
- Altinel und Ganiz 2018
- Nordberg und Grandien 2020
- Raamkumar, Tan und Wee 2020
- Rivera 2020

- Nguyen 2020
- Zhang, Junbo Zhao und LeCun 2015
- Kiranyaz u. a. 2019
- Severyn und Moschitti 2015b
- Severyn und Moschitti 2015a
- Jianfeng Zhao, Mao und Chen 2019
- Eren, Ince und Kiranyaz 2019

## 2.6 Grobgliederung

1. Einleitung
   1.1. Motivation
   1.2. Wissenschaftlicher Beitrag
   1.3. Struktur der Arbeit
2. Neuronale Netze
   2.1. Einleitung
   2.2. Arten des Neuronalen Lernens
   2.3. Netzwerkparameter und Hyperparameter
   2.4. Aktivierungsfunktionen
   2.5. Verlustfunktion
   2.6. Optimizer
   2.7. Epochen
   2.8. TensorFlow.js
   2.9. Schluss
3. Die natürliche Sprache
   3.1. Einleitung
   3.2. Vektorisierung des Textes durch Encoding
   3.3. Worteinbettungen

## Kapitel 3

# Grobe Zeit- und Ressourcenplanung

- **November 2020 bis Dezember 2020:** Kapitel 2, 3 (ca. 40% der Arbeit)
- **Januar 2021:** Kapitel 4 (ca. 35% der Arbeit)
- **Februar 2021:** Kapitel 5 (ca. 10% der Arbeit)
- **März 2021:** Kapitel 1, 6, 7 (ca. 15% der Arbeit)

# Abkürzungsverzeichnis

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Listings

# Literatur


Aggarwal, Charu C und ChengXiang Zhai (2012). „A survey of text classification algorithms“. In: *Mining text data*. Springer, S. 163–222.

Altinel, Berna und Murat Can Ganiz (2018). „Semantic text classification: A survey of past and recent advances“. In: *Information Processing & Management* 54.6, S. 1129–1153.

Anand, Ankesh, Tanmoy Chakraborty und Noseong Park (2017). „We used neural networks to detect clickbaits: You won't believe what happened next!“ In: *European Conference on Information Retrieval*. Springer, S. 541–547.

Chawda, Sarjak u. a. (2019). „A Novel Approach for Clickbait Detection“. In: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, S. 1318–1321.

Cho, Jaeyoung, Sangwon Lee und Tai Myoung Chung (2019). „A shop recommendation learning with Tensorflow. js“. In: *Proceedings of the Korean Society of Computer Information Conference*. Korean Society of Computer Information, S. 267–270.

Eren, Levent, Turker Ince und Serkan Kiranyaz (2019). „A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier“. In: *Journal of Signal Processing Systems* 91.2, S. 179–189.

Gairola, Siddhartha u. a. (2017). „A neural clickbait detection engine“. In: *arXiv preprint arXiv:1710.01507*.

Glenski, Maria u. a. (2017). „Fishing for clickbaits in social images and texts with linguistically-infused neural network models“. In: *arXiv preprint arXiv:1710.06390*.

Kaur, Sawinder, Parteek Kumar und Ponnurangam Kumaraguru (2020). „Detecting clickbaits using two-phase hybrid CNN-LSTM biterm model“. In: *Expert Systems with Applications*, S. 113350.

Kiranyaz, Serkan u. a. (2019). „1D convolutional neural networks and applications: A survey“. In: *arXiv preprint arXiv:1905.03554*.

Korde, Vandana und C Namrata Mahender (2012). „Text classification and classifiers: A survey“. In: *International Journal of Artificial Intelligence & Applications* 3.2, S. 85.

Kowsari, Kamran u. a. (2019). „Text classification algorithms: A survey“. In: *Information* 10.4, S. 150.

Kumar, Vaibhav u. a. (2018). „Identifying clickbait: A multi-strategy approach using neural networks“. In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, S. 1225–1228.

Nguyen, Kha (2020). „Real-time fashion items classification using TensorflowJS and ZalandoMNIST dataset“. In:

Nordberg, Gustav und Jesper Grandien (2020). *The crucial parts of text classification with TensorFlow. js and categorisation of news articles*.

Raamkumar, Aravind Sesagiri, Soon Guan Tan und Hwee Lin Wee (2020). „Use of health belief model–based deep learning classifiers for covid-19 social media content to examine public perceptions of physical distancing: Model development and case study“. In: *JMIR public health and surveillance* 6.3, e20493.

Rivera, Juan De Dios Santos (2020). „Identifying toxic text from a Google Chrome Extension“. In: *Practical TensorFlow. js*. Springer, S. 151–162.

Roberts, Adam, Curtis Hawthorne und Ian Simon (2018). „Magenta. js: A javascript api for augmenting creativity with deep learning“. In:

Seo, Jae Duk (o.D.). „Predicting Keyword Popularity using Tensorflow JS“. In: ().

Severyn, Aliaksei und Alessandro Moschitti (2015a). „Twitter sentiment analysis with deep convolutional neural networks“. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, S. 959–962.

— (2015b). „Unitn: Training deep convolutional neural network for twitter sentiment classification“. In: *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, S. 464–469.

Vorakitphan, Vorakit, Fang-Yie Leu und Yao-Chung Fan (2018). „Clickbait detection based on word embedding models“. In: *International Conference on*

*Innovative Mobile and Internet Services in Ubiquitous Computing*. Springer, S. 557–564.

Zhang, Xiang, Junbo Zhao und Yann LeCun (2015). „Character-level convolutional networks for text classification“. In: *Advances in neural information processing systems*, S. 649–657.

Zhao, Jianfeng, Xia Mao und Lijiang Chen (2019). „Speech emotion recognition using deep 1D & 2D CNN LSTM networks“. In: *Biomedical Signal Processing and Control* 47, S. 312–323.