TECHNISCHE UNIVERSITÄT MÜNCHEN

MASTERARBEIT

Berge versetzen
-
Text Mining in der öffentlichen Verwaltung

Autor:
Raphael SCHÖNBALL

Betreuung:
Dr. Nicola JENTZSCH
Prof. Simon HEGELICH

Trägerschaft:
Kirsten RULF, BUNDESKANZLERAMT
Alexander DÜRNAGEL, BUNDESMINISTERIUM für VERKEHR und DIGITALE INFRASTRUKTUR

Eine Arbeit für die Erfüllung der Anforderungen für den Abschluss

Master of Science

im Bereich

Political Data Science
Hochschule für Politik, TUM

14. Januar 2021

Eidesstattliche Erklärung

Ich, Raphael SCHÖNBALL, erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit „Berge versetzen - Text Mining in der öffentlichen Verwaltung“ selbständig angefertigt habe. Die aus fremden Quellen direkt und indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Ich weiß, dass die Arbeit in digitalisierter Form daraufhin überprüft werden kann, ob unerlaubte Hilfsmittel verwendet wurden und ob es sich – insgesamt oder in Teilen – um ein Plagiat handelt. Zum Vergleich meiner Arbeit mit existierenden Quellen darf sie in eine Datenbank eingestellt werden und nach der Überprüfung zum Vergleich mit künftig eingehenden Arbeiten dort verbleiben. Weitere Vervielfältigungs- und Verwertungsrechte werden dadurch nicht eingeräumt.

Unterschrift:

Datum:

TECHNISCHE UNIVERSITÄT MÜNCHEN

Zusammenfassung

Hochschule für Politik, TUM

Master of Science

Berge versetzen
-
Text Mining in der
öffentlichen Verwaltung

von Raphael SCHÖNBALL

Moderne Verwaltungen werden heutzutage mit Informationsmassen geflutet. Der Bedarf nach einer Vorstrukturierung und Einordnung ist sehr groß. Ein Beispiel für einen dieser Text-Korpora sind die Stellungnahmen, die bezüglich der Daten-Strategie der Bundesregierung im Jahr 2020 an das Bundeskanzleramt gesendet wurden.
Diese Arbeit untersucht anhand dieses Korpus die Machbarkeit, *Natural-Language-Processing*-Algorithmen u.a. aus dem *Deep-Learning*-Bereich für die Strukturierung und Visualisierung von Text-Daten einzusetzen und diskutiert deren Anwendbarkeit in der politischen und bürokratischen Praxis.
Für die Vorstrukturierung von Texten wurden drei Prototypen entwickelt: Ein Trendgebirge, welches die wichtigsten Themen und deren semantische Nähe zueinander anschaulich als ein Bergpanorama darstellt, ein *Co-Occurrence-Network*, welches die gemeinsamen Vorkommnisse von zentralen Begriffen in Form eines Netzwerks darstellt sowie eine Klassifikation von Texten aufgrund ihres Inhalts zu ihren Adressaten.

Inhaltsverzeichnis

Eidesstattliche Erklärung **iii**

Zusammenfassung **v**

1 Einführung **1**
1.1 Wissenschaftlicher Kontext 2

2 Methoden **5**
2.1 Latent Dirichlet Allocation (LDA) 5
2.2 *Elbow*-Methode 7
2.3 Neuronale Netze 8
2.3.1 Nichtlinearitäten 8
2.3.2 Optimierung, Kostenfunktion und *Backpropagation* . . . 9
2.3.3 Biologische Motivation 11
2.3.4 Aktivierungsfunktionen 12
2.3.5 Aufbau eines neuronalen Netzes 13
2.3.6 Festlegen der Größe des Netzwerks 14
2.3.7 Vorstellung ausgewählter *Layers* 15
Embedding-Layer 15
1D-Convolution 16
Max-Pooling 16
2.4 *Word2Vec* 16
2.5 *Keygraph-Keyword-Extractor* 17
2.6 *Convolution* 19
2.7 *Support-Vector-Machines* anhand eines Mobilitätsbeispiels . . . 21
2.8 *Decision-Tree* 25
2.8.1 Entropie 25
2.8.2 Die Methode 26
2.9 K-*Nearest-Neigbor* 26

3 Entwicklung der Prototypen **29**
3.1 Trendgebirge 29

3.1.1 *Preprocessing* 29
3.1.2 Analyse 31
Ort eines Berges auf der Ebene 31
Höhe eines Berges 31
Generierung des Trendgebirges 31
3.1.3 Ergebnis 36
3.1.4 Diskussion 38
3.2 *Co-Occurrence-Network* 40
3.2.1 Algorithmus 40
3.2.2 Ergebnis 41
3.2.3 Diskussion 43
3.3 Klassifizierung von Dokumenten 44
3.3.1 Einleitung 45
3.3.2 Datensatz 45
3.3.3 Klassifizierung mit herkömmlichen Methoden des maschinellen Lernens 48
Algorithmus 48
Resultate 49
3.3.4 Klassifizierung mittels *Deep-Learning* 50
Algorithmus 50
Resultate 52
3.3.5 Diskussion 52
3.3.6 Fazit 53

4 Fazit und Ausblick **55**

Literatur **59**

Abbildungsverzeichnis

2.1 Schematische Darstellung eines biologischen Neurons (Li, 2020a) 11
2.2 Darstellung eines künstlichen Neurons (Li, 2020a) 11
2.3 Die ReLU-Funktion (Li, 2020a) 13
2.4 Die Sigmoid-Funktion: $\sigma(x) = 1/(1 + e - x)$ (Li, 2020a) 13
2.5 Darstellung der Schichten eines neuronalen Netzes (Li, 2020a) 14
2.6 Je komplexer das neuronale Netz ist, desto genauer kann es Klassifikationsprobleme lösen. (Li, 2020a) 15
2.7 Verschiedene mögliche Trenn-Linien linear-trennbarer Daten mittels SVM. Verändert nach (Hucker, 2020) 22
2.8 Die Hyper-Ebene und ihr Rand. Verändert nach (Hucker, 2020) 23
2.9 Nicht-linear-trennbare Daten (Hucker, 2020) 24
2.10 Der *Kernel-Trick* (Hucker, 2020) 24

3.1 Aus dem Entstehungsprozess des Trendgebirges: Hier ragen nur die einzelnen Punkte der Worte aus der Ebene hervor. Nun wird im Folgenden dargelegt, wie daraus ein Gebirge entsteht. 32
3.2 Aus dieser Vorlage wurden verschiedene Bergsamples generiert. 33
3.3 Ein Bergsample. Es ist zu bemerken, wie sich die linke Seite dieses Berges mit der des Volcano-Berges ähneln. 33
3.4 Das zweite Bergsample 33
3.5 Das dritte Bergsample 33
3.6 Gewichtungsfaktor bei Überschneidungen 35
3.7 Trendgebirge: Beispielplot 1 37
3.8 Trendgebirge: Beispielplot 2 38
3.9 Trendgebirge mit *Box-Blur*-Matrix 38
3.10 Im Vergleich: Trendgebirge ohne *Box-Blur*-Matrix; mit im vorherigen Kapitel genannter *Convolution*-Matrix 39
3.11 Cooccurrence-Network 42
3.12 Cooccurrence-Network: Größter Komponent 43
3.13 PDF-Version einer kleinen Anfrage. Verändert nach (OpenKnowledgeFoundation, 2019) 46
3.14 Auszug aus dem Datensatz der kleinen Anfragen 46

3.15 Die Top-10-Ministerien nach erhaltenen kleinen Anfragen . . 47
3.16 *Embedding* einzelner Anfragen 49

Abkürzungsverzeichnis

KI	**K**ünstliche **I**ntelligenz
LDA	**L**atent **D**irichlet **A**llocation
NLP	**N**atural **L**anguage **P**rocessing
CBOW	**C**ontinuous **B**ag of **W**ords

Kapitel 1

Einführung

> *Es war eine Zeit, in der noch so viele mathematische Probleme unbearbeitet waren; man musste sich nur bücken, eines greifen und lösen.*
> - nach Carl Runge, Begründer des Runge-Kutta-Verfahrens, über Forschungsmöglichkeiten am Ende des 19. Jahrhunderts

Das Ende des 19. Jahrhunderts war eine Blütezeit für die Mathematik in Deutschland. Die Chancen, wie Carl Runge sie beschreibt, ein eigenes mathematisches Theorem oder Verfahren zu erforschen, waren aufgrund des Aufkommens vieler neuer mathematischer Theorien sehr groß.
Diese Chancen ergeben sich heute im Forschungsgebiet des *Natural Language Processing* (NLP). Unsere öffentlichen Verwaltungen stellen einen Bereich dar, in welchem derzeit noch viele Probleme mit existierenden digitalen Lösungsmöglichkeiten unbearbeitet brach liegen.

Regierungen sind heutzutage mit der schnellen Veränderung unseres Alltags durch die Digitalisierung konfrontiert. Verwaltungen müssen mit sehr großen Mengen an Informationen, wissenschaftlichen Ausarbeitungen, Stellungnahmen und Online-Meinungsäußerungen zurechtkommen, die zumindest wahrgenommen werden müssen. Bei knappem Personal und zeitlichen Limitationen wird der Ruf nach einer automatisierten Vorstrukturierung der Datenmassen laut.
Dies wurde im Rahmen dieser Arbeit für die Erstellung der Daten-Strategie der Bundesregierung als ein Pilotprojekt durchgeführt.
Die Erstellung der Daten-Strategie ist zentral beim Bundeskanzleramt in der Abteilung 6 'Digitalpolitik, Strategische IT-Steuerung' unter der Leitung von Eva Christiansen angesiedelt.
Zur Daten-Strategie haben viele Stellen des gesellschaftlichen Lebens, Verbände, Unternehmen, Wissenschaftler:innen, Institute, NGOs und Bürger:innen Stellungnahmen abgegeben, die bei der Erstellung der Daten-Strategie als Position wahrgenommen werden mussten. Mithilfe dieser Masterarbeit soll

eine algorithmische Vorstrukturierung dieser Datenmassen erfolgen.

Ziel der Arbeit ist es, die Machbarkeit der Anwendung von NLP-Algorithmen zur Analyse dieses Korpus anhand verschiedener Prototypen in drei ausgewählten Beispielen zu demonstrieren und deren praktische Anwendbarkeit in alltäglichen bürokratischen und politischen Prozessen zu diskutieren.

1.1 Wissenschaftlicher Kontext

Verschiedene Initiativen sind im öffentlichen Sektor im Bereich NLP in der öffentlichen Verwaltung im Entstehungsprozess; einige Ansätze sind schon aktiv daran, das Feld zu erschließen. Die angegliederten wissenschaftlichen Disziplinen wiederum waren und sind in diesem Bereich bereits sehr aktiv.

Aus dem *E-Government*- und dem *Open-Government*-Bereich gibt es schon länger Forschungsarbeiten, die sich mit der algorithmischen Analyse von Texten beschäftigen: Stylios et al. (Stylios et al., 2010) haben schon 2010 die Anwendung von *Text-Mining* für die Miteinbeziehung von Bürgerinnen und Bürgern in Regierungsentscheidungen inklusive einer Klassifizierung von positiven und negativen Einstellungen gegenüber Regierungsmaßnahmen in Online-Foren erforscht. Rao und Dey (Rao & Dey, 2011) haben 2011 Zusammenhänge zwischen politischen Themen und Diskussionen in Online-Foren algorithmisch ermittelt.

Iriberri und Leroy (Iriberri & Leroy, 2007) verwendeten NLP-Methoden im Rahmen eines internetbasierten, öffentlichen Kriminalitätsmeldesystems, indem sie zentral Verbrechenseigenschaften aus den Erzählungen von Zeugen herausgelesen haben und auf dieser Basis zusätzliche Fragen stellten.

Quinn et al. (Quinn et al., 2010) trainierten ein Modell, um ca. 118.000 politische Reden aus dem US-Kongress-Archiv aufgrund ihrer Wortwahl zu thematischen Klassen zuzuordnen. Es werden Schlüsselwörter identifiziert und es wird eine hierarchische Verschachtelung in Themen und Unterthemen durchgeführt. Ihr Modell ermöglicht es, thematisch ähnliche Reden zu verknüpfen und die Dynamik der politischen Agenda zu analysieren.

Kwon et al.(Kwon et al., 2006) klassifizierten öffentliche Kommentare zu politischen Themen in neun allgemeinem Themenklassen: Wirtschaft, Umwelt, Regierungsverantwortung, Gesundheit, Recht, Politik, Verschmutzung, Wissenschaft und Technologie. Für die Klassifikation der Kommentare wurden *Support-Vector-Machines* verwendet.

Im Bereich Daten-Visualisierung forscht Samuel Bohman im *Paper* 'Data Visualization: An Untapped Potential for Political Participation and Civic Engagement' (Bohman, 2015) daran, welche Macht Daten-Visualisierungen haben können. Er untersucht insbesondere neue Möglichkeiten für die Bürgerbeteiligungsforschung: *Data-Storytelling*, *Infographics*, *Data Physicalization*, und *The Quantified Self*.

O'Neill und Wheintal (O'Neill et al., 2017) wenden Visualisierungsalgorithmen in der Klimaforschung an. Nach einer Übersicht über die Geschichte und neuesten Entwicklungen der Datenvisualisierung werden neuere Tools mithilfe der Plattform *Tableau* auf Daten von Abfall-Verwertungs-Projekten in einigen Entwicklungsländern angewendet.

Zeitgenössische Forschung fokussiert sich ebenfalls auf das Thema *E-Government*. Abu et al. (Abu-Shanab & Harb, 2019) verwendeten *Text-Mining*-Methoden, um schließlich ein gesteigertes Interesse an ebenjenem Thema in der Forschungslandschaft festzustellen. Die Schlagworte *Open-Government*, *Smart-Cities* und *Analytics* würden unter Forschern in diesem Bereich zuletzt immer mehr Aufmerksamkeit genießen. Die Themen *Governance*, *E-Government-Adoption*, *E-Participation*, *E-Democracy*, *Administration and Procurement* wären ebenfalls zentral im wissenschaftlichen Fokus. *E-Government*

Eine Forschungsarbeit von Papadopoulos et al. (Papadopoulos & Charalabidis, 2020) fällt thematisch in das gleiche Gebiet der vorliegenden Arbeit. Die Forscher sammelten Dokumente nationaler Künstlicher-Intelligenz-Strategien (KI-Strategien), extrahierten Schlüsselwörter, führten eine LDA aus und verglichen ähnlich klassifizierte Strategien. Sie identifizierten zwei Gruppen an Regierungen: Eine Gruppe, der u.a. China und Japan angehören, hat einen starken wirtschaftlichen Fokus. Die andere Gruppe, der u.a. Deutschland angehört, untersucht in ihren KI-Strategien zusätzlich, wie die

Regierungsarbeit durch KI profitieren kann und wie ethische und sicherheitsbezogenen Themen in die weitere Entwicklung von KI integriert werden können.

Im gleichen Bereich ist die Fallstudie von Hagen et. al. (Hagen et al., 2015) Hier wurde die Anwendbarkeit von *Topic-Modeling* auf *E-Petitions* mit Beispieldaten des Petitionssystems WtP, *'We, the People '*, der Obama Administration untersucht. Die Konklusion der Studie ist die Anwendbarkeit von *Topic-Modeling* auf Petitionstexten, um kohärente Themen zu identifizieren.

Kapitel 2

Methoden

In dieser Masterarbeit wurde auf verschiedene theoretische Konzepte und Methoden zurückgegriffen. Diese werden im Nachfolgenden einzeln intuitiv vorgestellt und näher erläutert.

2.1 Latent Dirichlet Allocation (LDA)

LDA ist ein Algorithmus, welcher Themen in Textkorpora identifiziert. Er stellt fest, zu welchen Dokumenten Themen zu welchem Anteil zuordenbar sind und gibt zusätzlich an, welche wichtigen Schlagworte zu den jeweiligen Themen gehören. Der Algorithmus wurde 2003 von Blei, Ng und Jordan eingeführt. (Blei et al., 2003)

Die grundsätzliche Idee hierbei ist, dass Dokumente eine Verteilung über Themen sind, wohingegen jedes Thema durch Verteilung über Wörter repräsentiert wird. Die grundlegenden Annahmen sind, dass, wenn mehrere Dokumente über ein ähnliches Set an Wörtern verfügen, dass diese ähnlichen Sets dann auch von vergleichbaren Themen handeln. Zwei Wahrscheinlichkeitsverteilungen sind zentral: Zum einen die Wahrscheinlichkeitsverteilung eines Dokuments über verschiedenen Themen, also welchen Anteil ein Thema an einem Dokument hat; zum anderen die Wahrscheinlichkeitsverteilung eines Themas über verschiedenen Wörtern, also, welchen Anteil ein Wort in einem Thema an der Gesamtheit der Wörter dieses Themas hat.(Blei et al., 2003)

Der Algorithmus Zunächst wird jedem Wort willkürlich einem Thema zugewiesen. Dann wird immer wieder über alle Dokumente und in jedem Dokument über alle Wörter wie folgt iteriert: Es wird davon ausgegangen, dass die bisherigen Themen-Zuweisungen aller Wörter bis auf das aktuelle Wort w stimmen. Dann werden folgende Wahrscheinlichkeiten für jedes Thema t berechnet:

- Die Wahrscheinlichkeit des Themas t in diesem Dokument d : $P(t|d)$ also der Anteil der Wörter, die Thema t zugewiesen worden sind, durch die Anzahl der Wörter im Dokument.
- Die Wahrscheinlichkeit, dass ein Wort w zu einem Thema t gehört : $P(w|t)$ also wie häufig es im gesamten Textkorpus vorkommt, dass Wort w Thema t zugeordnet wird durch die Anzahl aller Wörter in Thema t.

Diese beiden Wahrscheinlichkeiten werden multipliziert:

$$Q = P(w|t) * P(t|d) \tag{2.1}$$

Das t, welches diese Wahrscheinlichkeit Q maximiert, bekommt das aktuelle Wort w zugeordnet. Dies wird so lange iteriert, bis sich der Algorithmus die Zuordnungen nicht mehr oder nur noch kaum bei einer weiteren Iteration ändert. Dieser Fixpunkt (mit der einmaligen Ausführung des Algorithmus' über den gesamten Korpus als Abbildung) beinhaltet die finalen Zuweisungen der Wörter zu Themen und damit auch, welche Themen je nach Wortbasis in einem Dokument vorkommen. (Blei et al., 2003)

Die zufälligen Zuweisungen von Themen zu Wörtern am Anfang vor Anwendung des Algorithmus geschieht nach der Dirichlet-Verteilung. Hierbei spielen α und β als Priors für das Resultat des Algorithmus' eine große Rolle: Die Wahl des Priors und damit die Wahl des Startpunkts der Optimierungsiteration kann, wie bei allen Optimierungsproblemen, einen großen Einfluss auf das Ergebnis haben. Je nachdem, ob es viele lokale Extremstellen in der Zielfunktion gibt, endet die Iteration in einem der lokalen Extremstellen in der Nähe des Startpunkts. Aufgrund dieser Wichtigkeit der Wahl der Priors α und β muss Nachfolgendes darüber gesagt sein: α setzt den Prior für die Wahrscheinlichkeitsverteilung Dokumente nach Themen. Ein hohes α trifft die Annahme, dass in einem Dokument eher viele verschiedene Themen vorkommen, wohingegen ein niedriges α annimmt, dass ein Dokument sehr wenige Themen enthält. β ist der Prior für die Verteilung Thema nach Wörtern. Analog wie bei α trifft ein hohes β die Annahme, dass ein Thema sehr viele verschiedene Wörter des Korpus beinhalten wird, wohingegen ein kleines β bedeutet, dass von nur wenigen, speziellen Wörtern für die Beschreibung eines Themas ausgegangen wird. (Blei et al., 2003)

Abschließend ist festzuhalten, dass dieser Algorithmus also nicht nur zur Identifizierung von Themen in Korpora verwendet werden kann, sondern

ebenfalls als Verfahren, um Dokumente zu finden, die gleiche Themen behandeln.

2.2 *Elbow*-Methode

Die Bestimmung der Anzahl an Themen beim LDA-Algorithmus ist eine entscheidende für dessen Resultat, da für ein untersuchtes Wort immer über alle Themen Wahrscheinlichkeiten ausgerechnet werden müssen.

Zur Bestimmung dieses wichtigen Parameters kann auch alternativ der Ansatz verfolgt werden, einfach die Anzahl der Themen inkrementell zu erhöhen, sodass bei jeder weiteren Erhöhung eine semantische händische Analyse erfolgen kann. So kann qualitativ die beste Anzahl an Themen identifiziert werden.

Eine quantitative und statistisch fundierte Methode ist die *Elbow*-Methode (Dewi & Thiel, 2017; Thorndike, 1953). Hierbei werden zunächst Dokument-*Embeddings* benötigt. Dafür eignet sich z.B. eine *Document-Term-Matrix*, in welcher jedes Dokument durch einen Punkt im *Feature-Space* aller vorkommenden Wörter dargestellt wird (natürlich bietet es sich aus *Performance*-Gründen an, nur die wichtigsten Wörter des Korpus für den *Feature-Space* zu berücksichtigen).(Dewi & Thiel, 2017)
Nun wird die *k-Means-Cluster*-Methode (Bishop, 2006) angewendet, um im *Feature-Space* die Dokumente zu *clustern*. Auf Basis der vorkommenden Wörtern sind ähnliche Dokumente in diesem *Feature-Space* nicht weit voneinander entfernt; hierfür wird nun die optimale Anzahl an *Clustern* gesucht, um die Punkte der Dokumente bestmöglich zu *Clustern* gruppieren zu können. Da auch der *k-Means*--Methode eine Anzahl an *Clustern* vorgegeben werden muss, gehen wir wie folgt vor:

Wir berechnen für jedes k, welches angemessen erscheint (z.B. von 1 bis 15) nach der *k-Means*--Methode den SSE (*Sum of Squared Errors*). Dieser wird für jedes k geplottet. Im resultierenden Plot nimmt der SSE-Wert mit jedem weiteren Inkrementieren von k ab. Interessant ist, wenn dieser Abfall aber einen Knick, einen *Elbow*, aufweist. Wenn der SSE-Wert plötzlich langsamer abfällt als zuvor, kann davon ausgegangen werden, dass ein dem Datensatz gut entsprechendes *Clustering* durch ein weiter inkrementiertes *k* nur weiter unterteilt wird, obwohl das der Datengrundlage nicht entspricht. Daher wird der *Elbow* im Plot identifiziert.

Die Stelle, an welcher sich der *Elbow* befindet, ist das gesuchte k, was als Anzahl an *Clustern* für das Identifizieren von Themen mit Hilfe des LDA-Algorithmus' herangenommen wird.(Dewi & Thiel, 2017)

2.3 Neuronale Netze

Neuronale Netzwerke bilden die Klasse von Algorithmen, die am ehesten als Künstliche Intelligenz beschrieben werden können. Sie sind aufgrund ihrer herausragenden Performanz in verschiedensten Gebieten auch für eine Anwendung bei politischen Fragestellungen interessant. Neuronale Netze finden sich auch in der Biologie als Systeme von verbundenen Neuronen, die hier diskutierten künstlichen neuronalen Netze als algorithmisches Modell, welches mittels Optimierungen in meist hoch-dimensionalen Räumen u.a. Klassifizierungs- oder Regressionsprobleme löst, werden im Folgenden dennoch aus praktischen Gründen als neuronale Netze aufgeführt. In diesem Unterkapitel sollen neuronale Netzwerke kurz nach (Li, 2020a, 2020b, 2020c) eingeführt werden.

2.3.1 Nichtlinearitäten

Ein zentrales Element von neuronalen Netzen sind neben vielen linearen die nicht-linearen Funktionen. Diese seien zunächst eingeführt:
Eine lineare Klassifizierung $s = Wx$ ermöglicht es, mithilfe eine Gewichtungsmatrix W eine Klassifizierung der Daten in x durchzuführen. Das Ergebnis der Klassifizierung ist in s enthalten.

Ein neuronales Netzwerk hingegen würde eine Klassifizierung so vornehmen:

$$s = W_2 max(0, W_1 x) \tag{2.2}$$

W_1 und W_2 sind Gewichtungsmatrizen. W_1 ist hierbei eine Zwischenmatrix, die die Datenpunkte zunächst in Zwischenwerte transformiert. Die Funktion max(0,-) ist eine Nichtlinearität, die pro Vektoreintrag von $W_1 x$ elementweise angewendet wird. Dies setzt alle Aktivierungen, also jeden einzelnen Eintrag des Ergebnisses von $W_1 x$, welche unter Null liegen, auf Null. Erneute Multiplikation mit W_2 ergibt einen Vektor, in dem Klassenwerte enthalten sind; d.h. die Information, welcher Klasse dieser Eintrag in Wahrscheinlichkeit zugeordnet wird. Wenn die Nichtlinearität weggelassen würde, könnten die

beiden Matrizen zu einer einzigen Matrix verrechnet werden und daher wären die vorhergesagten Klassenwerte wieder eine lineare Funktion der Eingabe.(Li, 2020a)

Ein dreischichtiges neuronales Netz könnte analog zu diesem grundlegenden Modell so aussehen:

$$s = W_3 max(0, W_2 max(0, W_1 x)) \tag{2.3}$$

Hier sind die Einträge von W_3, W_2, W_1 die zu lernenden Parameter. Nach Anwendung der Nichtlinearität entstehen jeweils Zwischenergebnis-Vektoren. Da diese nach Anwendung der Nichlinearität der mittleren Schichten, auch versteckte *Layers* genannt, entstehen, werden sie auch versteckte Vektoren bezeichnet. Die Anzahl an Dimensionen dieser versteckten Vektoren werden von Hyperparametern festgelegt, welche später näher erläutert werden.

2.3.2 Optimierung, Kostenfunktion und *Backpropagation*

Im Nachfolgenden wird, angelehnt an (Li, 2020b), erklärt, wie die Kostenfunktion und *Backpropagation* dafür verwendet werden, um das Model zu optimieren.

Die Zielfunktion des Optimierungsverfahrens ist die Kostenfunktion (auch *Loss-Function*). Diese gibt intuitiv an, wie viele falsche Ergebnisse eine Durchführung der *Forward-Propagation*, also eine Anwendung des Modells, erzeugt hat. Eine Kostenfunktion für ein binäres Klassifikationsproblem ist die *Binary-Crossentropy* H:

$$H = -\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{2.4}$$

y_i sind die Zielwerte des Modells, $\hat{y}_i$ sind die errechneten Werte des Modells und N ist die Anzahl der Datenpunkte.

Für die spätere Anwendung einer Klassifizierung mit mehreren Klassen wird die *Categorical-Crossentropy-Loss-Function* verwendet:

$$H = -\sum_{i=1}^{N} \sum_{j=1}^{C} \mathbb{I}_{j=y_i} log(\hat{y}_i) \tag{2.5}$$

y_i ist nicht, wie normalerweise üblich, ein Vektor, bestehend aus 0 und 1 (*One-Hot-Encoding*). Vielmehr ist y_i einfach eine Zahl, die die Klasse angibt. C ist die Anzahl an Klassen. $\hat{y}_i$ ist dagegen das Ergebnis der Anwendung des Modells und daher ein Vektor, bestehend aus Wahrscheinlichkeiten dafür, dass der Datenpunkt zu der jeweiligen Klasse gehört. $\mathbb{I}$ ist die Indexfunktion.

Das generelle Ziel der Optimierung ist die Minimierung der Kostenfunktion. Die Parameter W_2 und W_1 werden mit der Methode des stochastischen Gradienten-Abstiegs erlernt. Das Bilden der Ableitung nach allen Parametern in W_1 und W_2 gibt die Richtung an, in welche sich die Parameter verändern müssen, damit die Kostenfunktion weiter verkleinert wird. Das Ableiten der Kostenfunktion bedarf der Kettenregel. w sei ein Vektor aller Parameter. $\hat{y}_i$ ist dabei das Ergebnis vieler komplexer Berechnungen des Modell. Diese seien beispielhaft gezeigt: (Li, 2020b)

$$\hat{y}_i = f(W_k(f(W_{k-1}x_{k-1}) \tag{2.6}$$

$f(x)$ ist hier eine Aktivierungsfunktion, W_k die Gewichtsmatrix des vorletzten zum letzten *Layer*. x_k und W_{k-1} Gewichtsmatrizen. x_{k-1} sei der Ergebnisvektor des 3. letzten *Layers*. Dann gilt:

$$H'(w) = \frac{\partial H}{\partial \hat{y}_i} * \frac{\partial f(W_k x_k)}{\partial W_k x_k} * \frac{\partial W_k x_k}{\partial x_k} * \frac{\partial f(W_{k-1}x_{k-1})}{\partial W_{k-1}x_{k-1}} * \frac{\partial W_{k-1}x_{k-1}}{\partial x_{k-1}} * ... \tag{2.7}$$

Hier ist zu sehen, wie die Kostenfunktion wegen der iterativen Anwendung der Kettenregel nach und nach einzeln abgeleitet wird. Dieser Prozess, sich schrittweise von vorne nach hinten vorzuarbeiten, wird *Backpropagation* genannt. Das konträre Konzept ist die *Forward-Propagation*, also das Anwenden des Status-Quo des Modells auf einen Datenpunkt, sodass das Modell ein Ergebnis erzielt. Durch die *Forward-Propagation* sind alle Werte, die in die Ableitungen eingesetzt werden, schon errechnet worden und daher kann die Errechnung der Ableitung nach jedem Parameter direkt im Back-Propagation-Prozess vorgenommen werden. Für intuitive Erklärungen hierzu können Links in den Fußnoten für das Gradientenverfahren[1] und für die *Backpropagation*[2] gefunden werden.

[1]https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31)

[2]https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd)

2.3.3 Biologische Motivation

Das menschliche Gehirn besteht aus Neuronen. Man könnte auch sagen, dass sie die grundlegenden Berechnungseinheiten des Gehirns darstellen. Im menschlichen Nervensystem gibt es ca. 86 Milliarden Neuronen, die mit ca. 10^14 - 10^15 Synapsen verbunden sind (Li, 2020a). Abbildung 2.1 zeigt eine Cartoon-Zeichnung eines biologischen Neurons und Abbildung 2.2 im Vergleich dazu ein gängiges mathematisches Neuronen-Modell. (Li, 2020a)

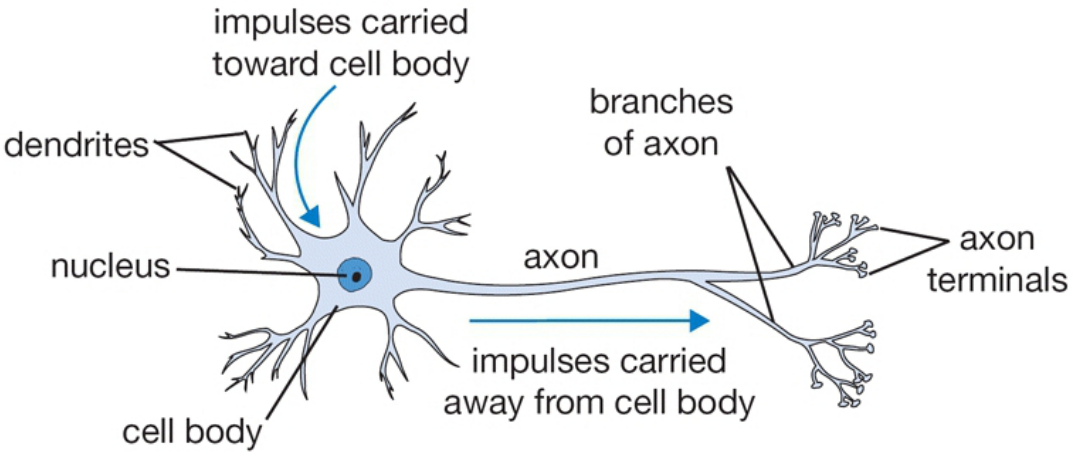

ABBILDUNG 2.1: Schematische Darstellung eines biologischen Neurons (Li, 2020a)

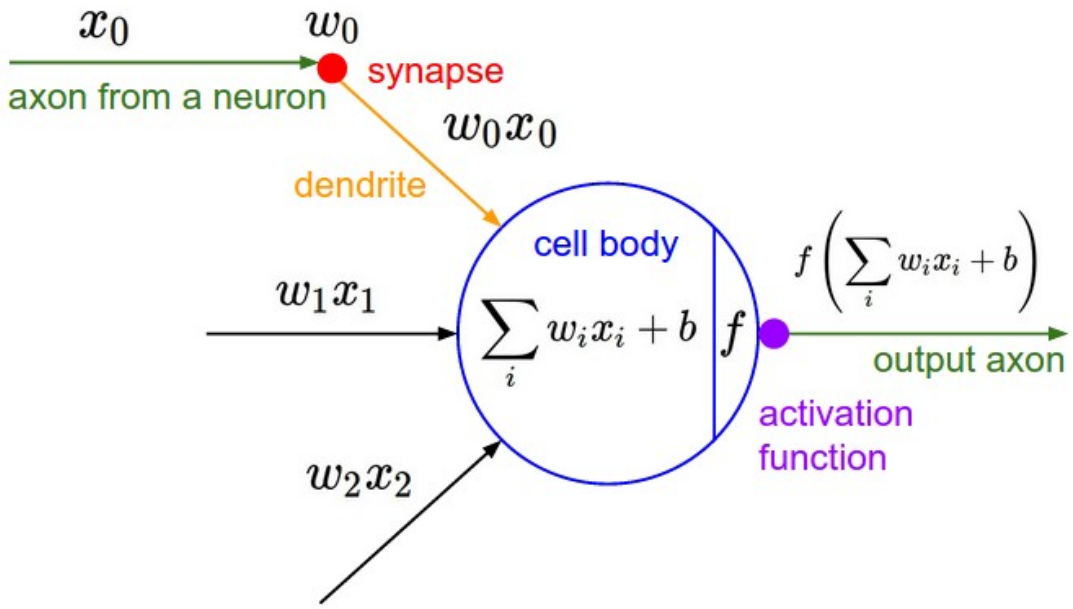

ABBILDUNG 2.2: Darstellung eines künstlichen Neurons (Li, 2020a)

Ein Neuron empfängt Eingangssignale über seine Dendriten, verrechnet die Signale im Zellkörper und leitet Ausgangssignale über sein (einziges)

Axon weiter. Über Synapsen an den Verzweigungen des Axons kommuniziert das Neuron mit Dendriten anderer Neuronen.
Im Rechenmodell eines Neurons werden die Signale x_0, die entlang des Axons und seiner Verzweigungen wandern, basierend auf dem Gewicht der Synapse mit dem Dendrit eines anderen Neurons w_0 multipliziert:

$$w_0 x_0 \tag{2.8}$$

Die Gewichte der Synapsen w sind erlernbar und lenken die Signale insofern, als dass ein Signal zu einem zweiten Neuron verstärkt wird (positives Gewicht) und zu einem anderen gehemmt wird (negatives Gewicht). So können sich Wege mit positiven Gewichten durch das neuronale Netz bilden, auf welchen ein Signal aufgrund eines bestimmten *Patterns* positiv weitergegeben wird.

Wenn die Summe aller $w_0 x_0$ innerhalb eines Zellkörpers über einem bestimmten Schwellwert liegt, welcher von der Aktivierungsfunktion kontrolliert werden kann, kann das Neuron feuern und sendet ein Signal entlang seines Axons. Jedes Neuron summiert also über alle Dimensionen i $w_i x_i$ auf, addiert einen Bias (in Abbildung 2.2 ist dieser b) und wendet die Aktivierungsfunktion an (in Abbildung 2.2 ist dies f). Eine Aktivierungsfunktion kann z.B. die Nichtlinearität von oben sein, die ReLU-Funktion (*Rectified Linear Unit*). Wie oben schon erwähnt, wird der Prozess des Weitergebens von Signalen durch das Netzwerk auch *Forward-Propagation* genannt. (Li, 2020a)

Diese Darstellung eines biologischen Neurons ist sehr ungenau: Nicht nur die Diversität unterschiedlicher Neuronenarten wird unterschlagen, sondern auch die Komplexität des nichtlinearen Systems eines Neurons.

2.3.4 Aktivierungsfunktionen

Aktivierungsfunktionen sind meist Funktionen von $\mathbb{R} \to \mathbb{R}$.

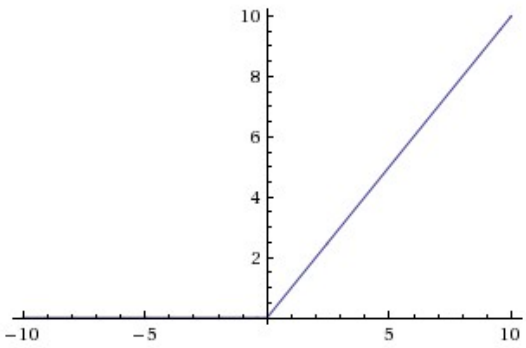

ABBILDUNG 2.3: Die ReLU-Funktion (Li, 2020a)

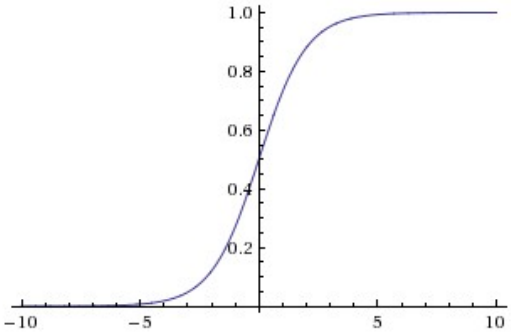

ABBILDUNG 2.4: Die Sigmoid-Funktion: $\sigma(x) = 1/(1 + e - x)$ (Li, 2020a)

Aktivierungsfunktionen sind zentral für den Trainingsprozess. Bei der Optimierung der Kostenfunktion wird der Gradient der Funktion, also die Ableitung in allen Dimensionen, gebildet. Aufgrund der Kettenregel, angewendet auf Formeln wie 2.3, beinhaltet die Ableitung der Gesamtfunktion auch die der Aktivierungsfunktion als Faktor. Daher ist der Gradient schnell 0, wenn die Ableitung der Aktivierungsfunktion sehr klein ist. Das ist bei der Sigmoid-Funktion in 2.4 z.B. dann der Fall, wenn besonders hohe oder tiefe Werte als x der Funktion übergeben werden. Daher ist eine zentrale Aufgabe des Umgangs mit neuronalen Netzen, auf die richtige Wahl der Aktivierungsfunktion zu achten. (Li, 2020a)

2.3.5 Aufbau eines neuronalen Netzes

Neuronale Netzwerke werden meist mit Netzwerken von Neuronen dargestellt. In diesem Graph sind Zyklen nicht erlaubt, da dies nicht terminierende Prozesse verursachen würde. Die Neuronen sind in verschiedenen Schichten, auch *Layers*, angeordnet. Populär sind die vollständig verbundene *Layers*, bei welchen jedes Neuron eines *Layers* mit allen Neuronen des benachbarten *Layers* verbunden ist.(Li, 2020a)

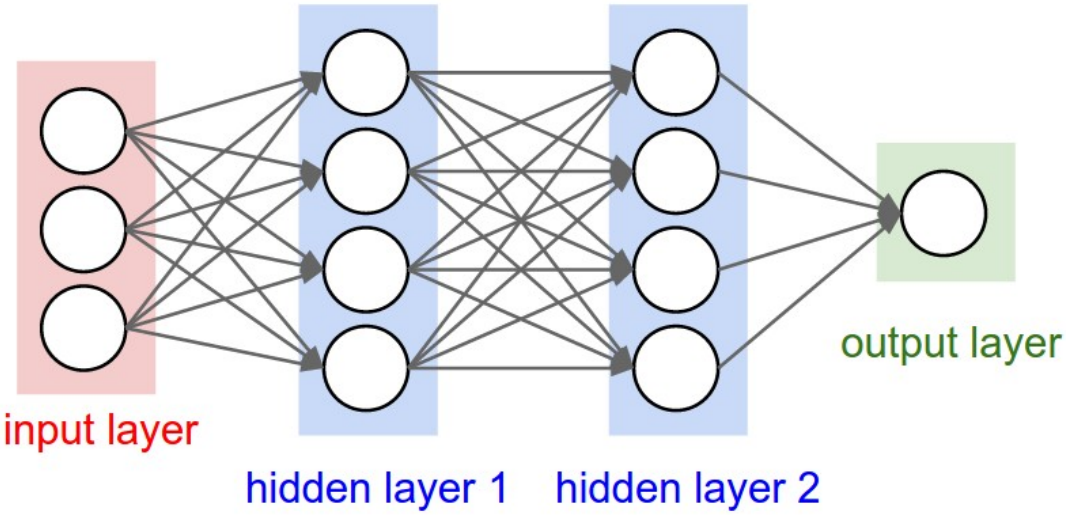

ABBILDUNG 2.5: Darstellung der Schichten eines neuronalen Netzes (Li, 2020a)

In Abbildung 2.5 wird beispielhaft ein 3-schichtiges neuronales Netz mit drei Eingängen, zwei versteckten Schichten mit je 4 Neuronen und einer Ausgabeschicht gezeigt.

Wie viele Parameter sind nun trainierbar?
Das Netzwerk in 2.5 hat $4 + 4 + 1 = 9$ Neuronen (die Eingabe-Neuronen werden nicht mitgezählt)
$[3x4] + [4x4] + [4x1] = 12 + 16 + 4 = 32$ Gewichte und $4 + 4 + 1 = 9$ biases w, also insgesamt 41 lernbare Parameter.(Li, 2020a)

2.3.6 Festlegen der Größe des Netzwerks

Je größer ein neuronales Netz ist, desto besser kann es Funktionen approximieren. Beispiel: Es geht um ein binäres Klassifizierungsproblem in zwei Dimensionen. Wir könnten drei separate neuronale Netze trainieren, jedes mit einer versteckten Schicht einer eigenen Größe und erhielten die folgenden Klassifizierer:

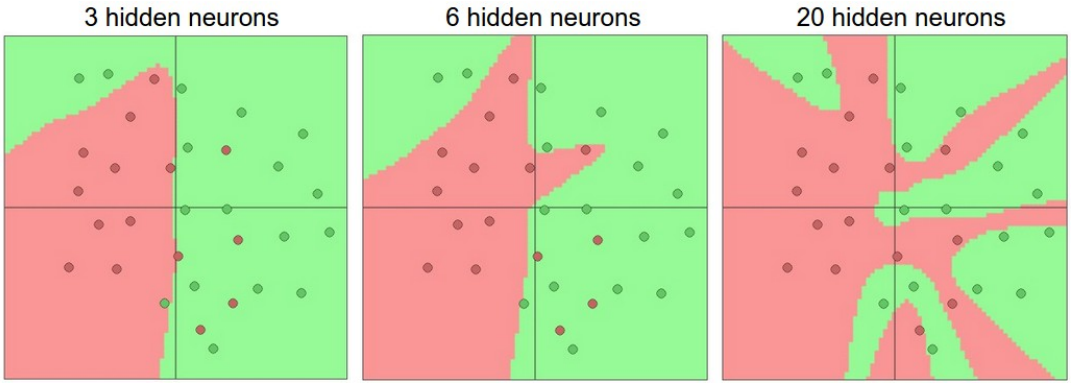

ABBILDUNG 2.6: Je komplexer das neuronale Netz ist, desto genauer kann es Klassifikationsprobleme lösen. (Li, 2020a)

Wie in Abbildung 2.6 zu sehen ist, kann das neuronale Netz mit mehr Neuronen in versteckten *Layers* die grünen Punkte besser zur grünen Fläche zuordnen.

Grundsätzlich sind größere Netze also leistungsfähiger als kleinere. Doch ist Vorsicht angebracht: Passt sich das Modell zu exakt an Daten an, führt dies zu *Overfitting*. Das Rauschen in einem Datensatz wird modelliert. Dies ist zu vermeiden, weil es darum geht, größere Zusammenhänge und Trends festzustellen, um ein allgemeingültiges Modell zu erzeugen, welches auf andere Datensätze übertragbar ist. Zum Beispiel erfasst das Modell mit 20 versteckten Neuronen alle Trainingsdaten korrekt, aber auf Kosten einer unklaren und zersplitterten Segmentierung des Raums in viele einzelne rote und grüne Entscheidungsbereiche.(Li, 2020a) Für die Verhinderung von *Overfitting* in Neuronalen Netzen gibt es verschiedene Methoden, z.B. L2-Regularisierung[3] oder *Dropout*[4].

2.3.7 Vorstellung ausgewählter *Layers*

Embedding-Layer

Embeddings dienen meist dazu, Wörter in Vektorräumen zu platzieren. Ein bekanntes Verfahren, *Word2Vec*, wird in 2.4 erklärt. Im *Deep-Learning*-Bereich wird ein *Embedding-Layer* verwendet, um die Dimensionalität eines Raumes zu reduzieren. Die Anzahl der Eingabe-Neuronen entspricht, gerade in

[3]https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c
[4]https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

sprachbasierten Trainingsprozessen, häufig der Anzahl der Wörter in einem Vokabular. Um eine große Zahl an Dimensionen und damit sehr vielen Gewichten zu vermeiden, erfolgt ein *Embedding*. Wörter werden in einem Vektorraum mit geringerer Dimension an unterschiedlichen Stellen platziert. Wie alle anderen Gewichte gehören die Gewichte der Kanten, die das vorherige *Layer* mit dem *Embedding*-Layer verbinden, zum neuronalen Netz und werden im Laufe des Verfahrens optimiert. (Kiela & Bottou, 2014; Koehrsen, 2019)

1D-Convolution

Zur allgemeinen Funktionsweise von *Convolutions* siehe Kapitel 2.6.
Die *Convolution* wird über einem Fenster, einer Datenmenge, mit zu spezifizierenden Dimensionen auf alle Datenpunkte im Fenster angewendet. Nach der Anwendung der Matrix auf die Eingabedaten wird der von der *Convolution* zu generierende Wert gebildet. Die *Kernel-Size* gibt an, wie viele Eingabedaten gleichzeitig analysiert werden. Die *Strides* geben an, um wie viele Schritte das Fenster, also die Anwendung der Matrix, bei jeder neuen Anwendungen der *Convolution* verschoben wird. Nach der Anwendung der Matrix wird zusätzlich eine Aktivierungsfunktion auf jeden Wert angewendet. Die *Padding*-Methode gibt an, ob die Dimensionen durch die Operation beibehalten werden sollen oder nicht. (Li, 2020c)

Max-Pooling

Max-Pooling ist eine weitere Methode, um Dimensionen zu reduzieren. Wenn Eingabedaten in einem n-dimensionalen *Array* vorliegen, nimmt sich das *Max-Pooling-Layer* einen verbundenen Teil des *Arrays* heraus und ersetzt ihn allein durch das Maximum aller Werte.
Eine Abwandlung des *Max-Pooling* ist das *Global-Max-Pooling*. Hierbei wird auf die Fenstergröße verzichtet und es wird das Maximum der untersuchten Dateneinheit herangezogen. (Li, 2020c)

2.4 *Word2Vec*

Word2Vec ist ein *Embedding*-Algorithmus, mit dem es möglich ist, Wörter eines Textes oder eines Textcorpus‘ in semantischer Weise in einem n- dimensionalen Vektorraum anzuordnen. Um ein Wort in einem Vektorraum zu platzieren, gibt es mehrere Methoden. (Mikolov et al., 2013)

Da im Rahmen dieser Arbeit der *Continuous-Bag-of-Words-Model*-Ansatz (CBOW) verwendet wurde, wird in den nachfolgenden Erläuterungen der Fokus auf diesen Ansatz gelegt. Er besagt, dass der Algorithmus sich jedes Wortes eines Textes annimmt. Das Wort, welches anhand von probabilistischen Modellen vorhergesagt werden soll, ist das untersuchte Wort. Um dieses untersuchte Wort wird ein *Window* gelegt mit einer *Window-Size* (von z.B. 5). Dies bedeutet in diesem Beispiel, dass sich im *Window* die 5 Wörter befinden, die vor dem untersuchten Wort stehen und die 5 Wörter, die nach dem untersuchten Wort stehen. Im CBOW-Ansatz wird nun anhand der *Window*-Wörter versucht, das untersuchte Wort vorherzusagen.

Dies geschieht mithilfe eines *Feedforwand-Neural-Net-Language-Models* (NN-LM). Dies besteht aus 4 *Layers*:
Einem *Input*-, einem *Projection*-, einem *Hidden*- und einem *Output-Layer*. Das *Input-Layer* kodiert jedes Wort im Korpus mit 1 oder 0, je nachdem, ob es im *Window* des untersuchten Wortes ist oder nicht. Dieses *Layer* wird auf ein *Projection-Layer* projiziert, was, weil es sich je nach *Window-Size* nur um wenige aktivierte Wörter handelt, sich als günstig bezüglich der Berechnungskosten herausstellt (nur wenige Gewichte müssen pro Update geändert werden). Danach folgt ein *Hidden-Layer* und das *Output-Layer*, welche das wahrscheinlichste Wort für das untersuchte Wort aus allen Wörtern im Korpus im Trainingsprozess bestimmen.(Mikolov et al., 2013)

2.5 *Keygraph-Keyword-Extractor*

Diese Methode ist geeignet, um Schlüsselbegriffe aus Texten zu extrahieren, kommt ohne die Anwendung von KI-Algorithmen aus und vermeidet damit den *Black-Box*-Effekt, welcher besagt, dass das Ergebnis eines Algorithmus nicht einfach erklärbar ist.(Ohsawa et al., 1998)
Zunächst wird eine gewisse Anzahl von Wörtern auf der Grundlage ihrer Häufigkeit ausgewählt und als Anfangsknoten dem Graphen hinzugefügt. Das Kantengewicht zwischen jedem dieser Wörter wird dann wie folgt berechnet:

$$assoc(w_1, w_2) = \sum_{s \in D} min(|w_1|_s, |w_2|_s), \tag{2.9}$$

s steht für einen Satz im Dokument D. w_1, w_2 stehe für zwei unterschiedliche Wörter, $|w|_s$ steht für die Häufigkeit des Terms w in Satz s. Z.B. kommen 2 Wörter in nur 3 Sätzen miteinander vor. Dann ist $min(|w_1|_s, |w_2|_s) = 0$ für

alle anderen Sätze und für die 3 Sätze z.B. $min(|w_1|_s, |w_2|_s) = 1$. Insgesamt wäre dann z.B. das Kantengewicht $assoc(w_1, w_2) = 3$.
Die (Anzahl an Knoten im Graphen) -1 Kanten mit dem größten Gewicht werden in den Graphen eingefügt.
Wenn eine Kante zwischen zwei Termen der einzige Pfad ist, über den die beiden verbunden werden, wird diese vom Graphen entfernt. Das bedeutet, dass Kanten im Graphen Teil von mind. einem Zyklus sein müssen. Dadurch entstehen meist auch Knoten, die zunächst gar nicht mit anderen Knoten verbunden sind. (Ohsawa et al., 1998)
Dadurch entstehende zusammenhängende Subgraphen werden als 'Konzept-*Cluster*' bezeichnet. Sie beinhalten paarweise miteinander in Sätzen vorkommende Wörter.
Dann werden alle Wörter werden nach ihrer *Key-Score* sortiert. Die *Key-Score* eines Terms wird wie folgt berechnet:

$$key(w) = 1 - \prod_{g \in G} (1 - \frac{based(w, g)}{neighbors(g)}) \tag{2.10}$$

mit g als *Cluster* in der Menge aller *Cluster* G und w dem Wort. Die Menge aller Sätze s ergibt das Dokument D. Die Funktion $based(w, g)$ gibt als intuitive Erklärung an, wie häufig Wörter aus g mit w zusammen in einem Satz s vorkommen. Die Funktion $neighbors(g)$ gibt an, wenn ein Satz überhaupt Wörter aus g enthält, wie viele Wörter er dann hat, außer der Wörter aus g. $|w|_s$ ist die Häufigkeit von Wort w in Satz s, $|g|_s$ wie häufig Wörter aus *Cluster* g im Satz s vorkommen. (Ohsawa et al., 1998)

$$based(w, g) = \sum_{s \in D} |w|_s |g - w|_s \tag{2.11}$$

$$neighbors(g) = \sum_{s \in D} \sum_{w \in s} |w|_s |g - w|_s \tag{2.12}$$

$$|g - w|_s = \begin{cases} |g|_s - |w|_s & \text{if } w \in g, \\ |g|_s & \text{if } w \notin g \end{cases} \tag{2.13}$$

Letztendlich wird also mit dem Bruch in Gleichung 2.10 für ein Wort w und einen *Cluster* g im gesamten Dokument berechnet, wie groß der Anteil von Wort w an allen Wörtern in Sätzen, in denen Wörter aus g vorkommen, ist. Es gibt Wörter w, die häufiger als einmal in einem Satz vorkommen. Es kann aber auch sein, dass w kein Mal in einem Satz vorkommt, in welchem aber Wörter aus *Cluster* g vorkommen. Dementsprechend ändert sich der Wert

des Bruchs $\frac{based(w,g)}{neighbors(g)}$. Es wird die Gegenprobe gebildet und dann über alle *Cluster* in G aufmultipliziert. Wörter w, die einen großen Anteil an allen Wörtern in Sätzen haben, in welchen auch Wörter aus g vorkommen, verkleinern dieses Produkt stärker als Wörter mit geringerem Anteil. Weil nochmal die Gegenprobe gebildet wird, ist der resultierende *Key-Score* für das Wort w dann größer.(Ohsawa et al., 1998)
Grundsätzlich gilt: Wenn w häufig mit Wörtern aus allen *Clustern* in Sätzen des Dokuments vorkommt, ist der *Key-Score* größer.

Die Wörter mit den größten *Key-Scores* werden in der Menge *High-Key-Terms* zusammengefasst. Dann werden diese ausgewählten Wörter als Knoten in G eingefügt (sofern sie nicht schon in G enthalten sind).

Wie werden die neu hinzugefügten Knoten nun mit dem Graphen verbunden?
Es werden alle möglichen Kanten zu einem *High-Key-Term* w_i nach $assoc(w_i, w_j)$ mit w_j als allen schon existierenden Knoten in G berechnet. Die Kanten mit den höchsten Werten, die den Wort w_i mit zwei oder mehr existierenden *Clustern* g verbinden, werden als neue Kanten für den Graphen G ausgewählt.(Ohsawa et al., 1998)

Zuletzt werden alle Wörter im nun existierenden Graphen nach ihrem Kantengewicht bewertet: Für alle Kanten eines Knotens w_i zu den benachbarten Knoten w_j wird $assoc(w_i, w_j)$ aufsummiert. Die Knoten mit den größten Werten werden als Ergebnis des *Keygraph-Keyword-Extractor* ausgegeben. (Ohsawa et al., 1998)

2.6 *Convolution*

Das Konzept einer *Convolution* ist ein gängiges Werkzeug in der Bildverarbeitung. Es kann zur Schärfung oder Weichzeichnung eines Bildes oder dafür verwendet werden, um Kanten und Ecken in einem Bild zu finden etc. Im *Deep-Learning*-Bereich sind *Convolutions* ein gängiges Mittel, um weitere Informationen aus Bildern zu generieren, um mit diesen zusätzlichen Information Trainingsprozesse des maschinellen Lernens zu unterstützen. (Szegedy et al., 2015)

Eine *Convolution* wird durch seinen *Kernel* definiert. Dieser ist eine Matrix K, die mit jedem möglichen Teil eines Bildes multipliziert wird (an den

Rändern ist dies mitunter nicht möglich). Die Bildpixel sind in Matrix B, der *Kernel* in der Matrix K dargestellt.
Dann gilt:

$$B' = K * B \tag{2.14}$$

wobei keine Matrix-Multiplikation gemeint ist, sondern eine elementweise Multplizierung der sich an selber Position befindenden Werte innerhalb der Matrizen. B' ist das modifizierte Bild.
Die Art des *Kernels* ist hierbei entscheidend für das Resultat, z.B. ob eine Weichzeichnung oder eine Schärfung etc. vorgenommen werden soll.
In dieser Arbeit sollten folgende *Kernel* vorgestellt werden: (Ludwig, 2020)

- Schärfung:
  Hier wird sich auf das zentrale Pixel konzentriert, wobei Nachbarpixel mit hohen Werten und damit z.B. mit intensiveren Farben den finalen Wert des behandelten Pixels erniedrigen können. Dies bewirkt, dass Intensitätsdifferenzen zwischen benachbarten Pixeln verstärkt werden und dass das Bild schärfer wird.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{2.15}$$

- Ecken- und Kanten-Identifizierung:
  Eine Erweiterung des Prinzips einer Schärfung; Erläuterung analog zum vorherigen Punkt.

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \tag{2.16}$$

- box blur:
  Hierbei wird jedes Pixel durch den Durchschnitt seiner Nachbarn ersetzt. Dies bewirkt, dass alle Formen ineinander verschwimmen (dies wird im Nachfolgenden demonstriert).

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.17}$$

Convolutions können auch im NLP-Bereich eingesetzt werden. Dort werden sie nicht auf Pixel-Daten angewendet, sondern z.B. auf Daten, die aus

den *Embeddings* verschiedener Wörter stammen, wobei jedes Wort durch einen Vektor an Signalen dargestellt wird.

2.7 *Support-Vector-Machines* anhand eines Mobilitätsbeispiels

Support-Vector-Machines (SVM) sind eine elegante Methode, um Klassifizierungs - oder Regressions-Aufgaben zu bewältigen. Zusammengefasst wird ein Vektorraum mit Hyperplanes so unterteilt, sodass anhand dieser Einteilung Datenpunkte in Klassen einsortiert werden können. Zunächst wird die binäre Klassifikation mit SVM vorgestellt.(Hucker, 2020)

Die Methode wird zuerst an linear trennbaren Daten erklärt:
Im Allgemeinen besteht die Idee der SVM darin, eine Grenzlinie zu identifizieren, die Datenpunkte in Klassen einteilt. Zu diesem Zweck betrachten wir das nachfolgende fiktive Beispiel:

Beispiel aus dem Mobilitätsbereich *E-Scooter* bieten als elektrische Roller Personen die Möglichkeit, schneller als zu Fuß von A nach B zu kommen. Mithilfe von Datenanalysen soll mehr über die Personengruppe herausgefunden werden, die einen selbst-verschuldeten Unfall mit einem *E-Scooter* gehabt hat. So können weitere Benutzer von *E-Scootern* in eine unfallgefährdete und nicht-unfallgefährdete Gruppe eingeteilt werden.

Die roten Punkte seien Daten von *E-Scooter*-Benutzer:innen mit selbstverschuldetem Verkehrsunfall und die blauen von Benutzer:innen ohne Unfall. Diese sind nach den Variablen Verkehrserfahrung in Jahren (z.B. ab dem Zeitpunkt einer bestandenen Fahrprüfung) und Sicherheit des Fahrstils, was beispielsweise anhand der Unfälle mit anderen Verkehrsmitteln mit Mitschuld gemessen werden könnte, angeordnet.
Das Ziel der Analyse ist es, eine Trennlinie, auch Hyperebene genannt, zu finden, welche die beiden Gruppen am besten trennt. Damit können neue Datenpunkte anhand dieser Linie zu einer Klasse zugeordnet werden.

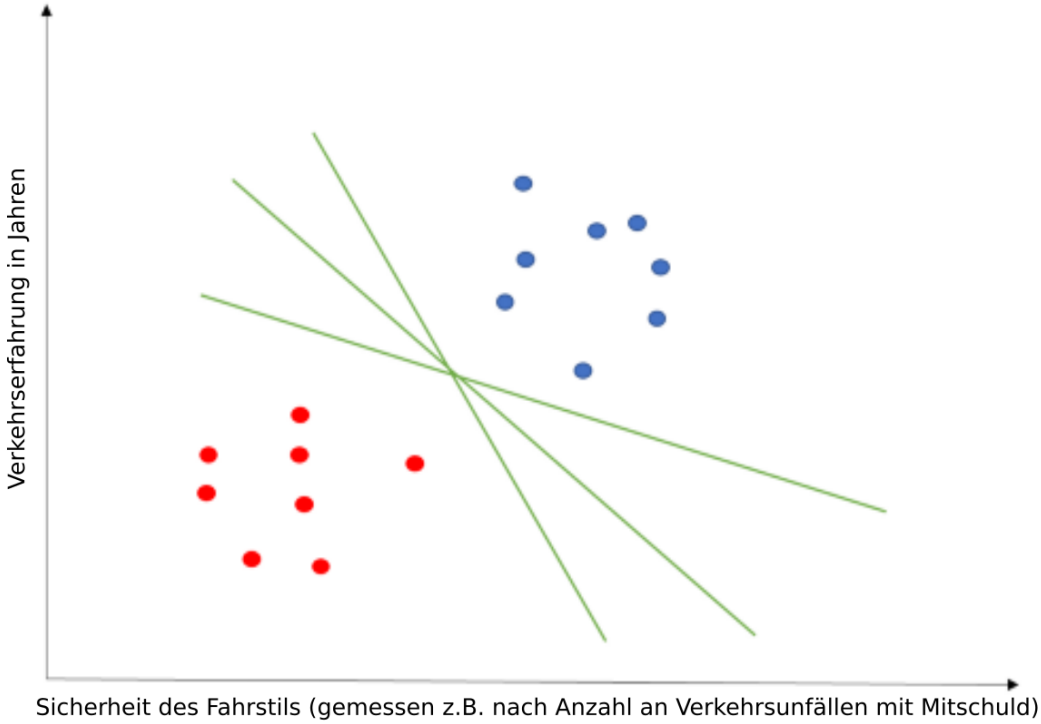

ABBILDUNG 2.7: Verschiedene mögliche Trenn-Linien linear-trennbarer Daten mittels SVM. Verändert nach (Hucker, 2020)

Der SVM-Algorithmus versucht durch Testen, die beste Trennlinie zu finden.

Die beste Hypereben bildet den größten Bereich zwischen den beiden Datengruppen, in welchem neue Datenpunkte am ehesten vom bisherigen Muster abweichen können, ohne die Gültigkeit des Modells zu untergraben. Die SVM suchen daher nach einer Grenze, die maximal weit von jedem Datenpunkt entfernt ist. Dieser Abstand von dieser Entscheidungsgrenze zu den nächstgelegenen Datenpunkten bestimmt den Rand des *Classifiers*.

Die beste Hyperebene ist diejenige, die den Rand maximiert, unter der Einschränkung, dass jeder Datenpunkt korrekt erfasst wird. (Hucker, 2020)

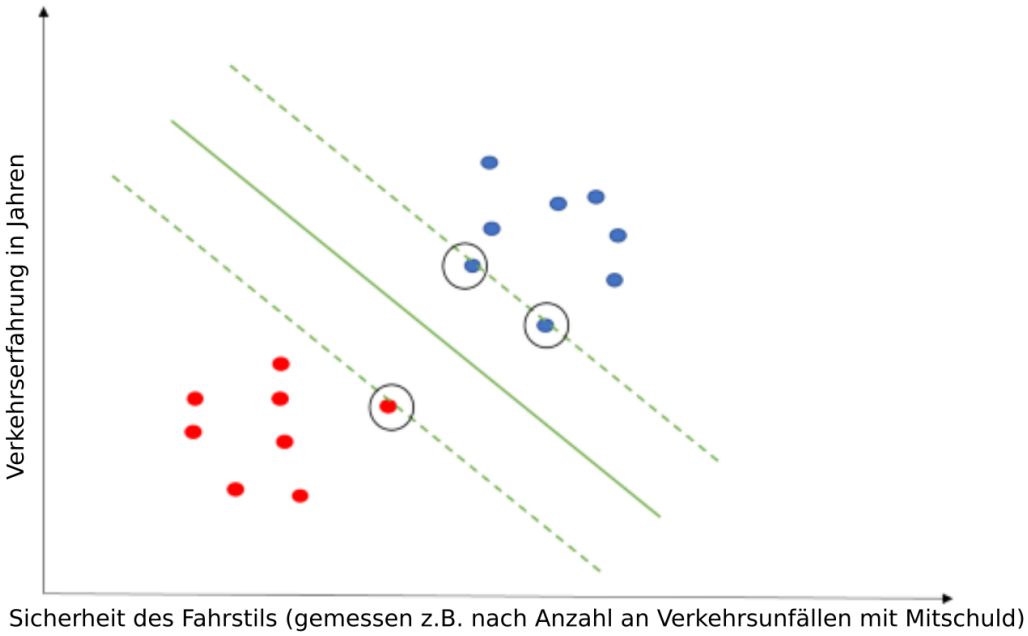

ABBILDUNG 2.8: Die Hyper-Ebene und ihr Rand. Verändert nach (Hucker, 2020)

Die eingekreisten Datenpunkte sind für die Modellbildung entscheidend. Diese Datenpunkte, die die Spitzen der Stützvektoren darstellen, sind letztendlich am Findungsprozess des optimalen Stützvektors beteiligt. Stützvektoren sind Vektoren, die senkrecht von der Ebene ausgehen.

Um den optimalen Stützvektor zu bilden, muss die Norm des Normalenvektors der Grenz-Hyperplane unter der Voraussetzung, dass die Hyperplane die zu trennenden Punkte voneinander abgrenzt, minimiert werden. Für die Lösung dieses quadratischen Optimierungsproblems wird die Langrange-Multiplier-Method mithilfe der Dualform und dem Kuhn-Tucker-Theorem verwendet. (Berwick, 2008)

Nun werden nicht linear-trennbare Daten behandelt:

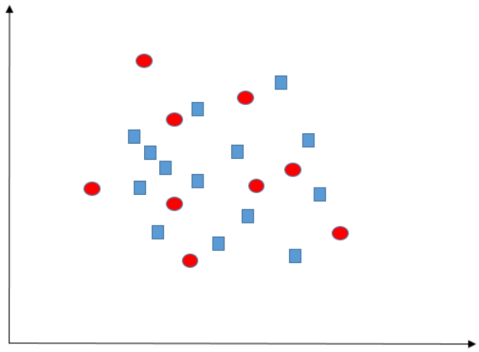

ABBILDUNG 2.9: Nicht-linear-trennbare Daten (Hucker, 2020)

Das Ziehen einer geraden Trennlinie ist nun nicht möglich. Um eine Trennung durchführen zu können, wird der *Kernel-Trick* angewendet. Alle Datenpunkte werden mithilfe einer *Kernel*-Funktion $\phi(x)$ in einen *Feature-Space* abgebildet, in welchem die Berechnung der optimalen Hyperplane möglich und einfacher sein kann.(Hucker, 2020)

Wenn die Kreise als Teil einer Ebene intuitiv gesprochen 'angehoben' werden, können mit Hilfe einer gegebenen Transformationsfunktion $\phi(x)$ die Datenpunkte besser getrennt werden:

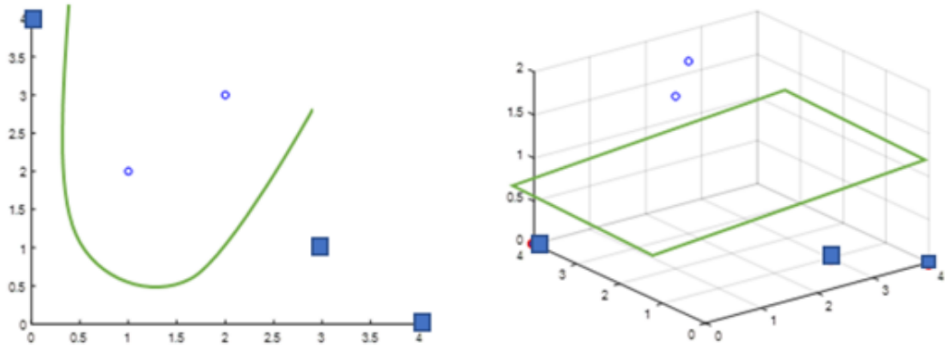

ABBILDUNG 2.10: Der *Kernel-Trick* (Hucker, 2020)

Die bekannteste der *Kernel*-Funktionen ist wohl die Radiale Basisfunktion (RBF) oder Gaußsche Kernfunktion, die nun beispielhaft angegeben wird:

$$K(x,y) = \exp^{-\frac{(|x-y|^2)}{2\sigma^2}} \tag{2.18}$$

wobei x und y zwei Vektoren von Features sind, die zwei verschiedene Datenpunkte repräsentieren. σ ist ein frei bestimmbarer Parameter.(Hucker,

2020)

Dank dieses Tricks ist es also möglich, mit nicht-linearen Datensätzen so umzugehen, als seien sie linear trennbar. (Alto, 2019)

Die Erweiterung für eine Klassifizierung mit mehreren Klassen erfolgt z.B. mit der *One-vs-Rest*-Methode:
Jede Datenpunktklasse wird mithilfe einer binären Klassifikation von allen andern getrennt. Die Vereinigung all dieser Grenzen ergibt dann den *Multi-Class-Classifier*. (Hucker, 2020)

2.8 *Decision-Tree*

2.8.1 Entropie

Um *Decision-Trees* vorzustellen, ist zunächst ein Blick auf das Konzept der Entropie wichtig. Eine Definition der Entropie ist

$$E(S) = \sum_{i=1}^{c} -p_i log_2 p_i \tag{2.19}$$

(T, 2019) Dabei ist p_i in dem behandelten Fall die Wahrscheinlichkeit, wie häufig ein Element / eine Klasse i in den Daten auftritt und S das betrachtete System.
Entropie wird auch als das (numerisch invertierte) Maß für die Reinheit einer Klasse an Datenpunkten bezeichnet.

Um die Verkleinerung der Reinheit, also die Steigerung der Entropie, bei einer Veränderung des Datensatzes zu messen, wird ein Maß benötigt. Dieser Informationsgewinn ist derartig definiert:

$$D(Y, X) = E(Y) - E(Y|X) \tag{2.20}$$

X sind zusätzliche Informationen, mit welcher die Entropie von Y verringert werden kann. $D(X, Y)$ ist also der Gewinn an 'Reinheit'. Je größer die Verringerung dieser Unordnung durch X ist, desto mehr Information wird aus X über Y gewonnen.

Die Variable, die die Entropie am meisten verringert, wird nun herangezogen, um anhand ihr die nächste Aufteilung des Datensatzes im Erstellungsprozess eines *Decision-Tree* durchzuführen. (T, 2019)

Neben der Entropie gibt es noch andere Maße für die Bestimmung der besten *Split*-Variablen, wie den {emphGini-Index oder die *Classification Error Rate*. (Hegelich, 2016)

2.8.2 Die Methode

Im Nachfolgenden wird dargelegt, wie ein Entscheidungsbaum-Algorithmus mit Hilfe der Konzepte Entropie und Informationsgewinn entscheidet, welches Attribut zuerst unterteilt wird und welche Variable mehr Informationen für die Zielvariable beinhaltet.

Es werden alle Differenzen in der Varianz aller noch nicht aufgesplitteten Variabeln errechnet. Die Variable, welche die Entropie am meisten verringert, wird vom Entscheidungsbaum aufgeteilt.

Dieses Verfahren wird iterativ fortgesetzt, bis eine vorgegebene Tiefe erreicht wurde oder bis es keinen weiteren Split gibt, welcher einen Informationsgewinn über einem Mindestgewinn (für eine Terminierung der Iteration) verursacht. Dabei ist zu beachten, dass der Informationsgewinn für jede verbleibende Variable bei jedem Iterationsschritt neu berechnet werden muss. Final kann dann mit dem erstellten Modell ein jeder unklassifizierter Datenpunkt einer Klasse nach den definierten *Split*-Eigenschaften zugeordnet werden. (T, 2019)

Random-Forest Eine Fortführung von *Decision-Trees* ist die Methode *Random-Forest*. Um *Overfitting*, eine zu detaillierte Approximation der Daten, zu vermeiden und mit dem Problem, dass leicht andere Daten zu einem vollkommen anderen *Decision-Tree* führen kann, werden mittels *Bagging* viele verschiedene *Decision-Trees* auf unterschiedlichen Teilmengen der verfügbaren Variablen trainiert. Am Ende entscheidet eine Mehrheitswahl unter allen *Decision-Trees* über die Klassifizierung eines Datenpunkts. Dieses Verfahren resultiert in robustere Ergebnisse und zu höheren *Accuracies*. (Hegelich, 2016)

2.9 K-*Nearest-Neigbor*

Die K-*Nearest-Neigbor*-Methode schließt von schon gelabelten Daten auf einen neuen Datenpunkt. (Bishop, 2006) Wenn wir einen Datensatz D in einem

Raum mit einem Abstandsmaß

$$D = \{(x_i, y_i)\}_{i=1}^{N} \text{ mit } x_i \in \mathbb{R}^D \text{ als Features und die Klassen } y_i \in \{1, ..., C\} \tag{2.21}$$

und einen neuen Datenpunkt x, der einer Klasse zuzuordnen ist, vorliegen haben, wird wie folgt vorgegangen:
Es wird eine Menge $N_k(x)$ gebildet, welche alle Datenpunkte enthält, die x am nächsten nach dem Abstandsmaß des Raums (beispielsweise einfach dem euklidischen Abstand) sind. Die Klassifizierung wird zu der Klasse, die die Mehrheit von $N_k(x)$ stellt, vorgenommen: (Bishop, 2006)

$$\begin{aligned} p(y = c|x, k) &= \frac{1}{k} \sum_{i \in N_k(x)} \mathbb{I}(y_i = c), \\ \hat{y} &= argmax_c p(y = c|x, k), \\ &\mathbb{I} \text{ ist die Indikator-Funktion.} \end{aligned} \tag{2.22}$$

Kapitel 3

Entwicklung der Prototypen

Aufbauend auf den im vorherigen Kapitel erläuterten Methoden werden nun die algorithmischen Prototypen, die im Rahmen dieser Arbeit erstellt wurden und die die Machbarkeit von NLP-Anwendungen in der öffentlichen Verwaltung demonstrieren, sowohl vorgestellt als auch diskutiert. Die Codes wurden auf der Programmier-Plattform KNIME (Berthold et al., 2008) und in den Programmiersprachen python und R implementiert.

3.1 Trendgebirge

Bei der Visualisierung von Datensätzen gibt es viele Ansätze, die manchmal aufgrund ihrer Struktur nur wenige Informationen beinhalten können. Visualisierungstypen wie Balkendiagramme oder *Wordclouds* sind aufgrund ihrer 2-Dimensionalität in ihrer Aussagekraft begrenzt. Der hier vorgestellte Ansatz betritt in diesem Bereich die 3. Dimension.

Ein Trendgebirge hat das Ziel, wichtige Schlagworte und deren semantische Nähe in Dokumenten oder Korpora an Dokumenten zu identifizieren und in Form eines simulierten, 3-dimensionalen Gebirges zu visualisieren. Damit ist es möglich, Themen in großen Text-Korpora auf einen Blick zu erkennen. Für die Auswertungen werden im Nachfolgenden typische Regierungstexte verwendet. In dem besprochenen Beispiel werden Texte anaylsiert, die im Rahmen der Datenstrategie der deutschen Bundesregierung öffentlich auf Wirtschaftsverbands-, Forschungs- und Zivilgesellschaftsseite entstanden sind.

3.1.1 *Preprocessing*

Um Ergebnisse der Analyse zu verbessern, sind zunächst ein paar Text- Reinigungsschritte notwendig. Da die Texte meist in PDF-Form vorliegen, muss

zuerst eine Konvertierung in das auf KNIME übliche *Document*-Format vorgenommen werden. Nach dieser Konvertierung werden alle *Markup-Tags* gefiltert. *POS-Tagging* wird mithilfe des *Stanford Taggers*[1] unternommen. Mithilfe des *STTS* (*StuTTgart Part of Speech*[2])- *Tagger* werden dann nur Fremdwörter und Nomen ausgewählt. *Lemmatization* wird ebenfalls mit dem *Stanford Lemmatizer* (*Stanford Core NLP library*[3]) durchgeführt. *Punctuation Erasure*, also das Herauslöschen von Punktuationssymbolen, wird trotz der bisherigen Filterungs-Schritte für den Fall durchgeführt, dass gewisse Satzzeichen noch in den Daten enthalten sind. Dasselbe gilt für Ziffern und Zahlen. Danach werden alle Wörter, die weniger als 4 Buchstaben beinhalten, ebenfalls entfernt, um beispielsweise Akronyme auszusortieren. Je nach Analysetext werden danach noch manuell mithilfe von *Regexp-Expressions*[4] verschiedene Wörter aussortiert. *Stopwords*[5] werden ebenfalls herausgefiltert (die Redundanz dieses Schrittes ist offensichtlich, eine gewissen Redundanz sorgt allerdings für eine größere Robustheit). Ein *Case Converter*, also eine Funktion, die alle Wörter in diesem Fall in ihre entsprechende Kleinschreibung konvertiert, stellt sicher, dass keine Wörter doppelt mit unterschiedlicher Groß- oder Kleinschreibung im Korpus vorkommen.

Mit den gereinigten Daten werden nun verschiedene Analysen vorgenommen. Ziel der Analysen ist es, eine 3D-Einbettung der Begriffe zu erstellen und damit ein Gebirge entstehen zu lassen. Dies wird nicht einfach durch ein 3D-*Embedding* erreicht.
Der Ort des Berges wird hauptsächlich durch ein 2D-*Embedding* mittels *Word2Vec* festgelegt. Die Höhe des Berges wird entweder durch die Häufigkeit des Wortes im Dokumenten-Korpus oder durch die Zentralität des Wortes innerhalb eines Themas der *Topic-Modeling*-Analyse (LDA) festgesetzt.

[1] Nähere Informationen: http://nlp.stanford.edu/software/tagger.shtml
[2] Für weitere Informationen: https://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/germantagsets/
[3] Nähere Informationen: https://stanfordnlp.github.io/CoreNLP/index.html
[4] Eine Einführung zu Regexp: https://regexone.com/
[5] Eine nähere Erklärung: https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

3.1.2 Analyse

Ort eines Berges auf der Ebene

Für diese Analyse wird *Word2Vec* eingesetzt. Es wird eine *Window-Size* von 5, eine minimale Wortfrequenz von 5, eine *Batch-Size* [6] von 1000 und für die Dimension des letzten *Layer* 2 angesetzt. Wie in 2.4 erläutert, wird der *Continuous-Bag-of-Words*-Ansatz verwendet. Insgesamt werden je nach Analyse und vorhandener Infrastruktur verschiedene Anzahl an Epochen ausgewählt (ca. 5-10). Die ermittelten x- und y-Werte werden als Platzierung der Berge auf der x-y-Ebene festgelegt.

Höhe eines Berges

Methode 1: *Topic-Modeling* Für die *Topic-Modeling*-Analyse wird eine *Latent-Dirichlet-Allocation*-Analyse durchgeführt. Die Anzahl der zu identifizierenden Themen wird mithilfe der *Elbow*-Methode gefunden. Alpha ist 0,5 und beta 0,001, da es sonst Schwierigkeiten bei der Abgrenzung der einzelnen Themen bei einem höheren Beta-Wert gibt. Der *Weight*-Parameter, der angibt, wie wichtig ein Begriff für ein Thema ist, definiert die Höhe des Berges für das Trendgebirge.

Methode 2: *Word-Counts* Mithilfe eines *Bag-of-Words*-Ansatzes[7] wird für jedes Wort die absolute *Term-Frequency* ermittelt. Mithilfe des *Snowball-Stemmers*[8] werden Wörter, die auf den gleichem Wortstamm zurückzuführen sind, aggregiert. Am Ende werden die Top-20 Wörter, die am häufigsten vorkommen, für das Trendgebirge ausgewählt. Die Zähl-Werte stellen also die Höhe des Berges dar.

Generierung des Trendgebirges

Die x-,y- und z-Werte werden zusammengeführt. Beim *Topic-Modeling* wird für eine bessere Übersichtlichkeit ein Thema und deren Wörter ausgewählt. Danach werden die Themen aufsteigend sortiert.

[6] Siehe 3.3.4

[7] Eine Erläuterung: https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[8] Für weitere Informationen, siehe https://snowballstem.org/

Nun folgt, wie eine Visualisierung generiert wird, die einem Gebirge nahe kommt.

Es wird ein 2D-*Array* erstellt, welches in jedem Punkt die Höhe des Gebirges an dieser Stelle beinhaltet. Die Indizes eines Eintrages geben den Ort auf der Ebene an, der Wert des Eintrags steht für die Höhe. Z.B. gibt es inital im *Array* nur Nullen als Einträge, weil noch keine Worte eingefügt sind. Der erste 3D-Punkt eines vorher gefundenen Top-Begriffes wird an der entsprechenden Stelle im *Array* abgespeichert. Das wird für alle ausgewählten Wörter durchgeführt. Danach sind einige Einträge in diesem *Array* nicht 0. Diese sind die Gipfel der Berge. Wenn man den *Array* nun als 3D-Modell plottet, sieht man nur strichförmige Ausreißer zu diesen zukünftigen Gipfeln hin. Anbei ein Bild aus dem Entstehungsprozess:

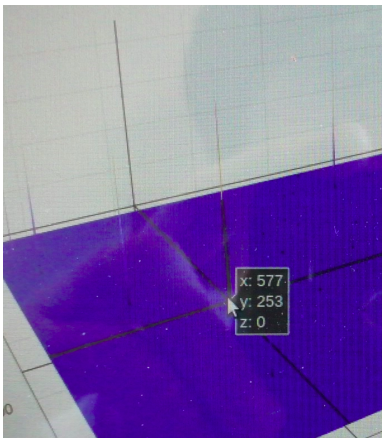

ABBILDUNG 3.1: Aus dem Entstehungsprozess des Trendgebirges: Hier ragen nur die einzelnen Punkte der Worte aus der Ebene hervor. Nun wird im Folgenden dargelegt, wie daraus ein Gebirge entsteht.

Nun folgt, wie eine bergähnliche Umgebung dieser Punkte graphisch erzeugt wird.

Erstellung von Sample-Bergen Das *R-Package plotly* beinhaltet einen Beispiel-Plot *Volcano*. Diese realistische Nachbildung eines Vulkans wurde zum Vorbild für alle zu generierenden Berge (siehe Abbildung 3.2)

Diese Datei liegt in der gleichen Form vor wie das schon erstellte *Array*: In einem 2D-*Array* sind als Werte jeweils die Höhe des Plots enthalten. Durch das Abschneiden und Spiegeln der Vulkan-Vorlage in Kombination mit der Bearbeitung mit einer in der Amplitude konstant abnehmenden

Sinus-Funktion können so unterschiedlich aussehende Berg-*Samples* erstellt werden. Insgesamt wurden 3 Sample-Berge erstellt (siehe Abbildungen 3.3, 3.4 und 3.5).

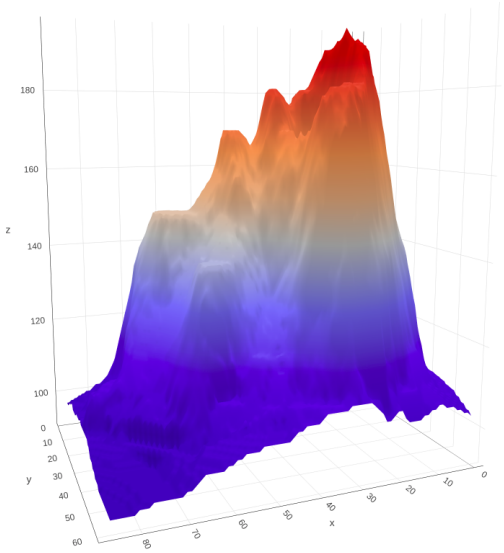

ABBILDUNG 3.2: Aus dieser Vorlage wurden verschiedene Bergsamples generiert.

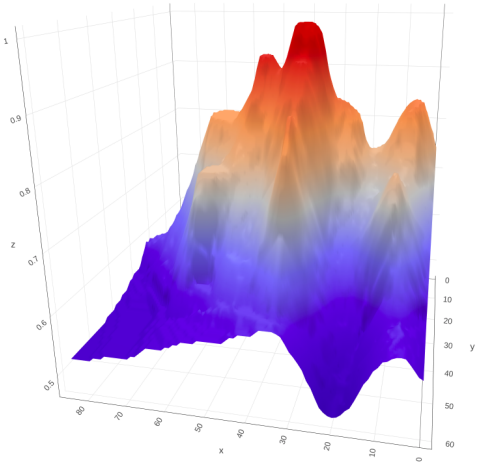

ABBILDUNG 3.3: Ein Bergsample. Es ist zu bemerken, wie sich die linke Seite dieses Berges mit der des Volcano-Berges ähneln.

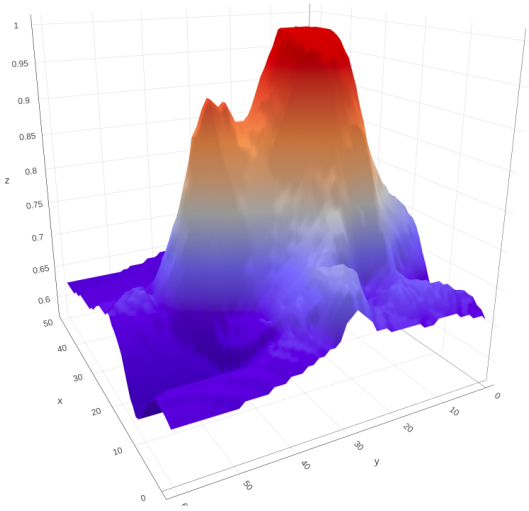

ABBILDUNG 3.4: Das zweite Bergsample

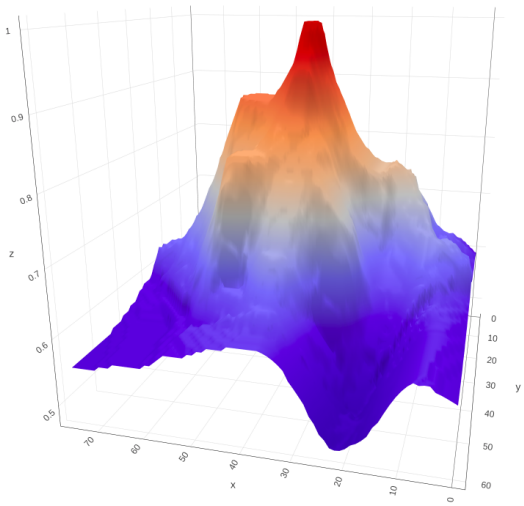

ABBILDUNG 3.5: Das dritte Bergsample

Einfügen der Sample-Berge Die rechte Seite des Sample-Berges in Abbildung 3.3 ist lediglich eine parallele Kopie der linken Seite von 3.2, welche auf der rechten Seite zusätzlich mit einer Sinus-Funktion bearbeitet wurde, um eine vollständig symmetrische Bergform zu vermeiden.

Aus der Menge an Berg-*Samples* wird zufällig eines ausgewählt und je nach *Row-Number* des zu verarbeitenden Wortes eine Drehung von 90 Grad unternommen, damit eine Häufung eines Berg-*Samples* zusätzlich vermieden werden kann. Der *Array* (das Gitter) des Berges wird auf 1 normiert, sodass der Gipfel den Wert 1 hat. Dieser *Array* wird nun im Gesamten mit der Höhe des ersten Wortes multipliziert; d.h., dass ein Berg mit der Höhe des darzustellenden Worts generiert wird.
Dieser *Array* wird nun so in den *Array* des gesamten Gebirges eingefügt, indem einfach die 0-Werte mit den neuen Werten des einzufügenden Berges überschrieben werden. Dies wird mit alle Wörter in der Liste nacheinander durchgeführt.

Dabei gibt es aber Konflikte: Was, wenn zwei Gipfel zu nahe aneinander liegen, dass sich die Berge also überschneiden?
Die Antwort auf diese Frage begründet gleichzeitig, warum die Gipfelbegriffe in aufsteigender Reihenfolge sortiert werden mussten: Berge, die größer sind, sollte natürlich erst nach den kleineren Bergen eingefügt werden, damit sie nicht von kleineren Berge überdeckt werden, was zu sehr unnatürlichen Brüchen im Plot führt.
Um zu vermeiden, dass gewisse Berge und damit auch Wörter von anderen Bergen derart überlagert werden, dass sie verschwinden, wird die Platzierung der Berge wie folgt nachjustiert:

Die Beispielberge haben eine rechteckförmige Grundfläche. Wenn ein neuer Berg an einen Ort gesetzt werden soll, an welchem die Überschneidung mit einem schon existierenden Berg größer als eine gewisse Pixel-Anzahl ist (z.B. 23 Pixel in eine Achsenrichtung), dann wird der Berg von dem schon existierenden Berg in die entgegengesetzte Richtung weggeschoben (z.B. wenn die linke obere Ecke eine zu große Überschneidung hat, wird der Berg nach rechts unten verschoben).

Ein weiteres Problem ist, dass die Berge sehr kantig werden können. Manchmal gibt es viel zu glatte, abgeschnittene Bergwände, die von einem großen

zu einem kleinen Berg oder zur Grundfläche überleiten und dadurch eine unnatürliche Optik verursachen. Deswegen wird in Überschneidungsbereichen ein gewichteter Mittelwert der Werte der beiden überschneidenden Berge gebildet und in den *Array* eingefügt (je nach Nähe zum einen oder zum anderen Berg). Diese Gewichtung kann linear vorgenommen werden, kann aber auch nach einer Sigmoid-Funktion für einen etwas abrupteren Wechsel der Gewichtungsfunktion erfolgen. In den Anschauungsbeispielen unten ist folgende Sigmoid-Funktion eingesetzt worden:

$$f(x) = \frac{1}{(1 + \exp^{6x-3})} \tag{3.1}$$

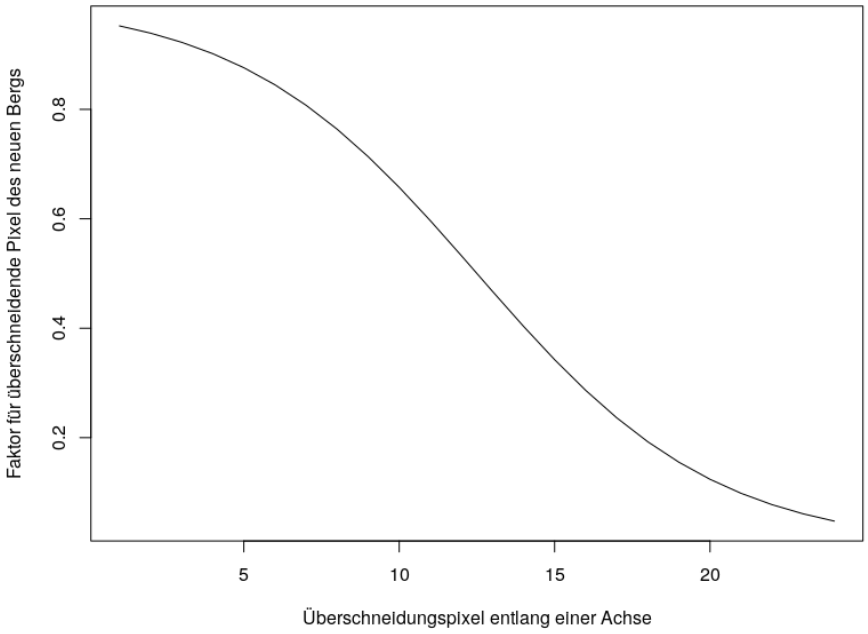

ABBILDUNG 3.6: Gewichtungsfaktor bei Überschneidungen

Ein neu hinzukommender Berg wird, je nachdem, wie weit er Überschneidungspixel mit einem schon existierenden Berg hat, bei der Bildung des gewichteten Mittelwerts mit den Höhewerten des schon existierenden Bergs machnmal mehr oder manchmal weniger gewichtet: Je näher man sich am Rand des Berggitters befindet, desto weniger Gewicht hat der Höhenwert bei der Mittelwertbildung. Das Gewicht fällt zum Rand hin ab; dieses abfallende Gewicht ist in der Abbildung dargestellt.

Gleiches gilt in umgekehrter Weise für den schon existierenden Berg, sodass die beiden Berge genau in der Mitte der Überschneidung (also bei dem gezeigten Beispiel bei Pixel 12 bzw. 13) ungefähr gleich gewichtet werden (jeweils mit ca. $0,5$). Damit wird ein natürlicher Übergang zwischen den Bergen sichergestellt.

Nun gibt es noch das weitere Problem, dass die Täler meist direkt auf die Grundfläche, also auf die Höhe 0, abfallen und dort konstant sind. Die Täler sollten sich eher möglichst den umgebenden Bergen anpassen. Hierzu werden *Convolutions* angewendet. Alle Bildpunkte werden mit einer Mischung aus einer *Box-Blur*-Matrix und einer Schärfungs-Matrix, welche über die normierte Summe beider hergestellt wird, geändert: Somit entspricht ein neuer Wert eines Punktes dem Durchschnittswert aller Nachbarn, die in diagonaler Richtung von ihm bis zu einer maximalen Weite (welche von der Größe der Matrix abhängig ist, z.B. 10x10) liegen. Die in den unten gezeigten Beispielen verwendete *Convolution*-Matrix ist meistens folgende:

$$\frac{1}{2N}\begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ 0 & \ddots & 0 & 1 & 0 \\ \vdots & 0 & 2N & 0 & \vdots \\ 0 & 1 & 0 & \ddots & 0 \\ 1 & 0 & \cdots & 0 & 1 \end{pmatrix} \tag{3.2}$$

3.1.3 Ergebnis

Der erstellte Algorithmus auf der Programmier-Plattform KNIME zeigt ein robustes Verhalten mit PDF-Eingaben. Es sind keine weiteren Eingaben nötig. Dies geht über bisher bekannte Arbeiten in diesem Bereich, wie z.B. die Arbeit von Dorian Grosch (Grosch, 2018), hinaus. Diese Arbeiten beruhen auf von Hand von Experten erstellten, qualitativen Analysen, welche mithilfe eines Trendgebirges visualisiert werden. Die vorliegende Arbeit stellt einen neuen Ansatz vor, welcher die quantitative Analyse von PDFs mittels *Word2Vec*, *Term-Frequency* und LDA mit einer damit abgestimmten Visualisierung in Form des Trendgebirges verbindet.
Ergebnis-Plots stehen nach Ausführung des Algorithmus als interaktives 3D-Modell zur Verfügung. Die beispielhaften Bilder der produzierten 3D-Modellen

wurden mit einem Textkorpus mit 5 Dokumenten erstellt:

Dem Bericht der Kommission für Wettbewerbsrecht (Kommission-Wettbewerbsrecht-4.0, 2019), einem Beitrag von FIfF, dem Forum für InformatikerInnen für Frieden und gesellschaftliche Verantwortung e.V. (Thiel, 2019), dem Gutachten der Datenethikkommission (Datenethikkommission, 2019), einem Paper des Rats für Sozial- und Wirtschaftsdaten 'Zur Entwicklung der Dateninfrastruktur in Deutschland' (Habich et al., 2010) und einem Bericht über eine Dateninfrastruktur für die medizinische Forschung (Krawczak & Weichert, 2017). Diese Auswahl wurde zufällig getroffen.

Die resultierenden Plots können wie folgt aussehen:

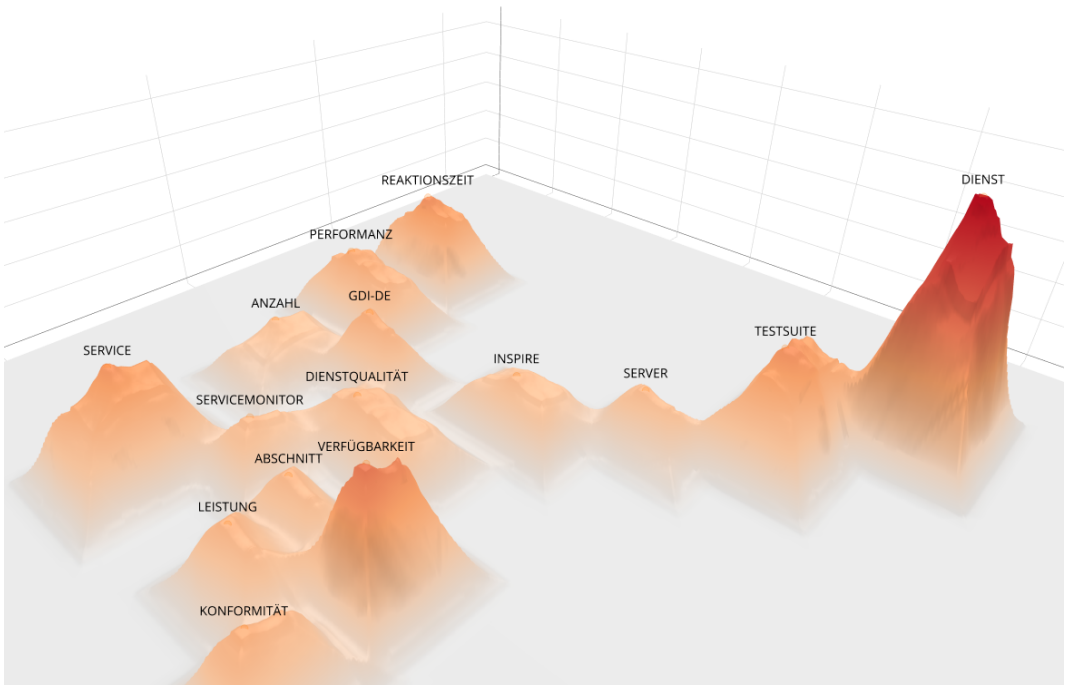

ABBILDUNG 3.7: Trendgebirge: Beispielplot 1

Die Wahl der *Convolution*-Matrix für möglichst natürliche Übergange wurde oben erwähnt. Hier sei gezeigt, wenn beispielsweise eine reine *Box-Blur*-Matrix für die Berge verwendet wird:

$$\frac{1}{N^2}\begin{pmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & \ddots & 1 & \vdots & 1 \\ \vdots & 1 & 1 & 1 & \vdots \\ 1 & \vdots & 1 & \ddots & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix} \tag{3.3}$$

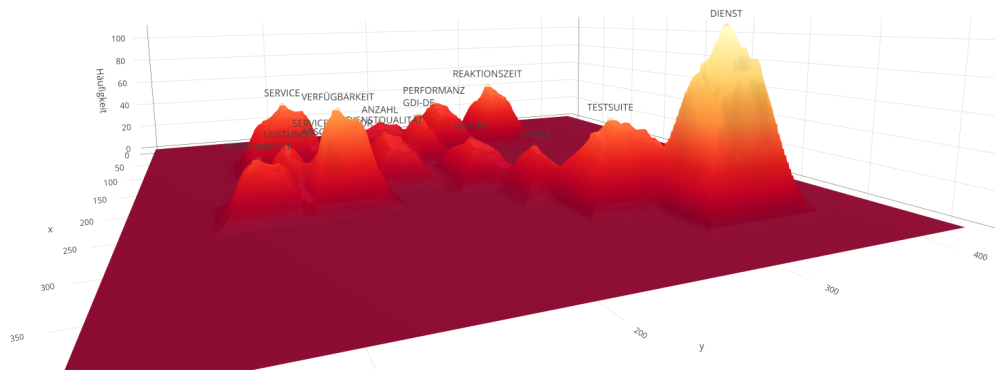

ABBILDUNG 3.8: Trendgebirge: Beispielplot 2

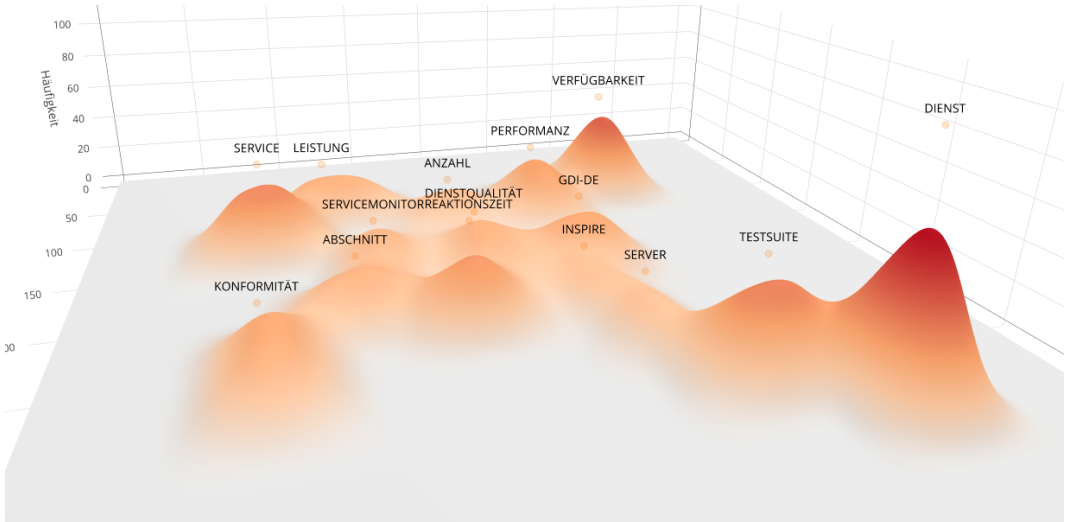

ABBILDUNG 3.9: Trendgebirge mit *Box-Blur*-Matrix

Es ist zu beobachten, dass eine wüstenartige Landschaft entsteht, die so nicht intendiert und damit den vorherigen Ergebnisplots hinten anzustellen ist.

3.1.4 Diskussion

Zugunsten einer vollständigen Anzeige aller Analyseergebnisse in der Visualisierung mussten die algorithmischen *Embedding*-Positionen des *Word2Vec*-Algorithmus verzerrt dargestellt werden, vergleichbar mit dem Anwenden einer *Jitter*-Funktion auf Datenpunkten.
Bei der Interpretation der Ergebnisse muss allerdings immer mit einer leichten Verschiebung der genauen Positionen der Berge gerechnet werden. Die

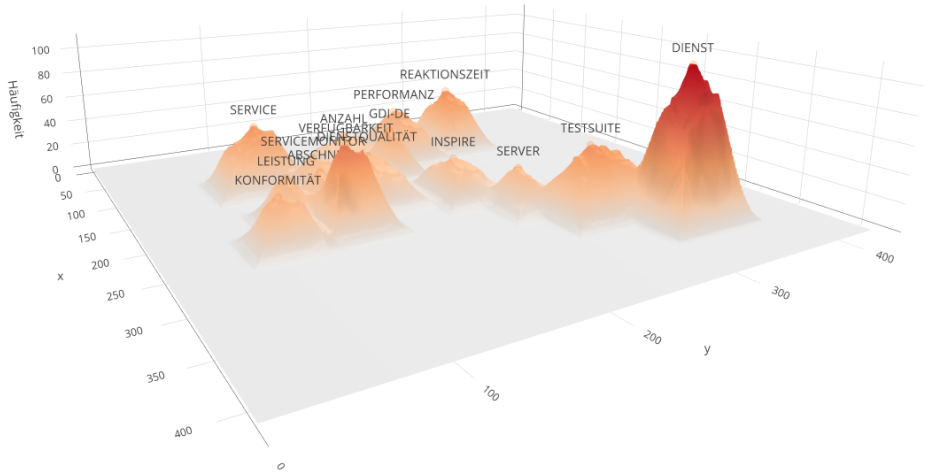

ABBILDUNG 3.10: Im Vergleich: Trendgebirge ohne *Box-Blur*-Matrix; mit im vorherigen Kapitel genannter *Convolution*-Matrix

Verzerrungen stellen die Sichtbarkeit aller Berge sicher und erhöhen die Übersichtlichkeit der gesamten Graphik.

Die Algorithmen, die für die Ermittlung der Höhe der Berge verwendet wurden, also die LDA-Analyse und die Worthäufigkeit, beschränken sich auf die Behandlung von Einzelwörtern. Bezüge zwischen Wörtern, z.B. in einem Satz, werden nicht hergestellt. Dieser Aspekt wird beim nachfolgenden Prototypen des *Co-Occurrence-Network* untersucht.

Die Bergform wird für jeden Berg zufällig ausgewählt und dient lediglich der Anschauung. Interpretationen von Formen von Bergen haben keine Grundlage in den analysierten Textdaten.

Die Funktionsweise des Algorithmus und entstandenen Visualisierungen, die Übersichtlichkeit der Anordnungen sowie die Möglichkeit, verschiedenen Unterdebatten und deren inhaltliche Nähe auf einen Blick sehen zu können, wurde bei internen Vorstellungen begrüßt. Der Algorithmus kann als Orientierungshilfe über Themen in einem Korpus dienen. Gerade für Menschen, die nicht über viel Zeit verfügen, sich in Fachdebatten einzulesen, kann diese Art der Visualisierung eine Hilfestellung sein.

Kritisiert wurde bei ersten Präsentationen, dass die Visualisierung für Personen, die sich gut in einem Themengebiet auskennen, eher weniger neue

Erkenntnisse bietet.

Grundsätzlich sind Anwendungen, um Politiker:innen einen intuitiven Überblick über Diskurse zu geben sowie im Bereich der Transparenz und der Information von Bürger:innen in Zukunft denkbar.

3.2 *Co-Occurrence-Network*

Ein *Co-Occurrence-Network* zeigt, welche Wörter mit welchen anderen Wörtern häufig in einem Satz oder Absatz oder in einem gewissen Abstand zwischen Wörtern miteinander vorkommen. Im Netzwerk symbolisieren die Knoten die Wörter und die Kanten zeigen an, dass zwei Wörter häufig in der angegebenen Texteinheit miteinander vorkommen.

Diese Visualisierung kann dem Betrachtenden, also Politiker:innen, Bürger:innen etc., einen Überblick über die wichtigsten Schlagworte geben, indem die zentralen Knoten, die mit den meisten anderen Knoten verbunden sind, analysiert werden. Sie kann zudem auch Themen-*Cluster* anzeigen, wenn man eng-vernetzte Subgraphen und somit eng verbundene Wörter im Text betrachtet.

Für diese Arbeit sei ein gemeinsames Auftreten von Wörtern auf Satzebene als Kriterium für eine Kante gewählt.

3.2.1 Algorithmus

Zunächst wird ein Text-Korpus ausgesucht; im unten erläuterten Beispiel werden PDFs analysiert. Das *Preprocessing* erfolgt analog wie oben beim Trendgebirge dargestellt (siehe 3.1.1). Daher wird auf eine Darstellung der einzelnen Schritte hier verzichtet.

Nachdem die Wörter der Dokumente mittels eines *Bag-of-Words*-Ansatzes extrahiert worden sind, kann das gemeinsame Vorkommen dieser Wörter in Sätzen aller Dokumente gezählt werden (*Co-Occurrence - Count*). Danach muss für die Erstellung des Netzwerkes ein *Key* für ein paar an häufig miteinander vorkommenden Wörtern erstellt werden. Dieser wird einfach durch das Aneinanderhängen der beiden Strings der Wörter erstellt. Es wird nun

nach diesem *Key* gruppiert und nach dem *Co-Occurrence - Count* aufsummiert, da gleiche Wort-Paare in verschiedenen Dokumenten vorkommen können. Letzlich werden noch alle *Co-Occurrence-Counts* unter 6 ebenfalls im Sinne der Übersichtlichkeit des Netzwerks entfernt.

Nun liegt eine Tabelle vor, in der jede Zeile eine Kante zwischen zwei Wörtern inklusive ihres Gewichts darstellt. Aus der Tabelle wird ein Netzwerk erstellt und dieses Netzwerk mit dem Kamada-Kawai-*Layout* (Kamada, Kawai et al., 1989) geplottet. Der erste Plot ist das gesamte, resultierende Netzwerk mit allen Knoten. Der zweite zeigt der Übersichtlichkeit wegen nur den größten zusammenhängenden Komponenten des Netzwerks.

3.2.2 Ergebnis

Als Beispielanwendung wurde derselbe Textxorpus wie der der Beispielanwendung des Trendgebirges analysiert: Der Bericht der Kommission für Wettbewerbsrecht (Kommission-Wettbewerbsrecht-4.0, 2019), einen Beitrag von FIfF, dem Forum für InformatikerInnen für Frieden und gesellschaftliche Verantwortung e.V. (Thiel, 2019), dem Gutachten der Datenethikkommission (Datenethikkommission, 2019), einem Paper des Rats für Sozial- und Wirtschaftsdaten 'Zur Entwicklung der Dateninfrastruktur in Deutschland' (Habich et al., 2010) und einem Bericht über eine Dateninfrastruktur für die medizinische Forschung (Krawczak & Weichert, 2017). Diese Auswahl wurde zufällig getroffen. Der resultierende Plot des Gesamtnetzwerks ist hier zu sehen:

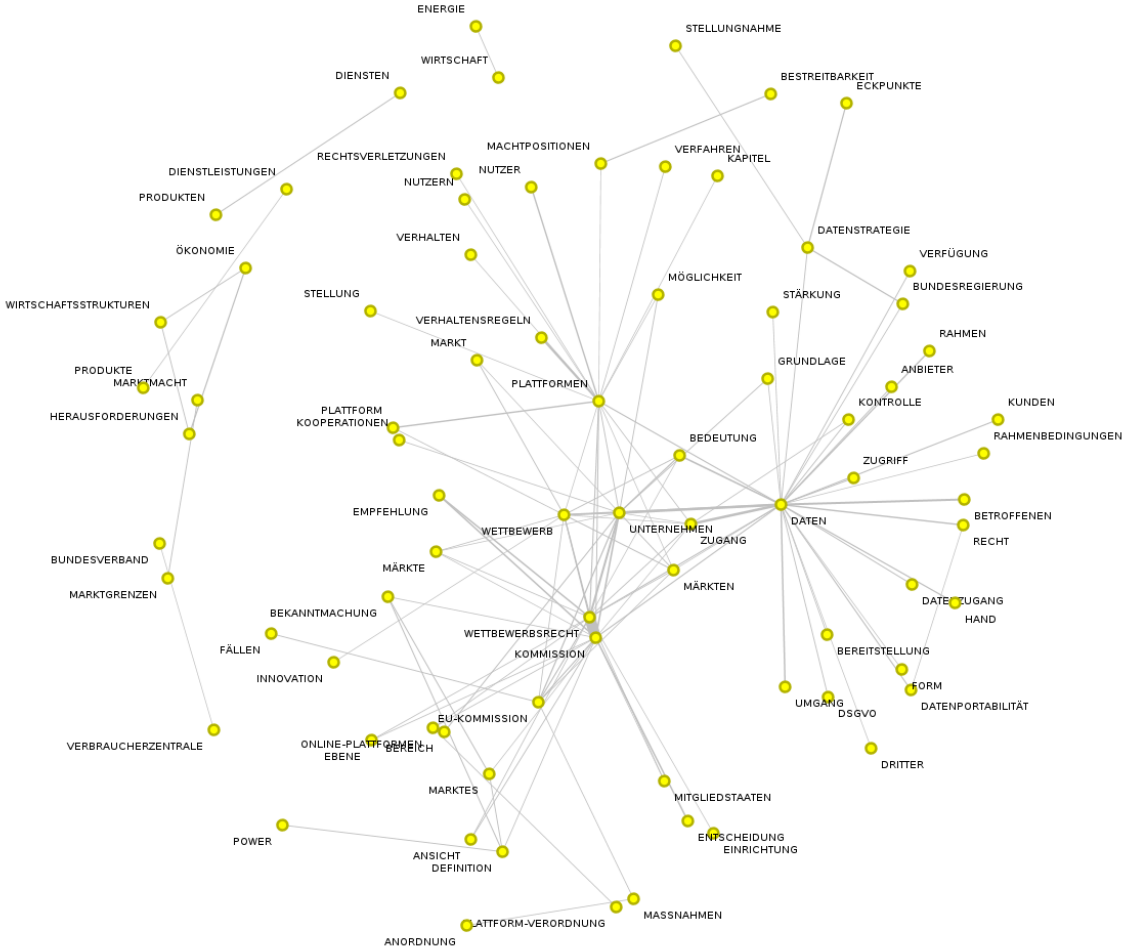

ABBILDUNG 3.11: Cooccurrence-Network

Hier sehen Sie den größten zusammenhängenden Komponenten vergrößert:

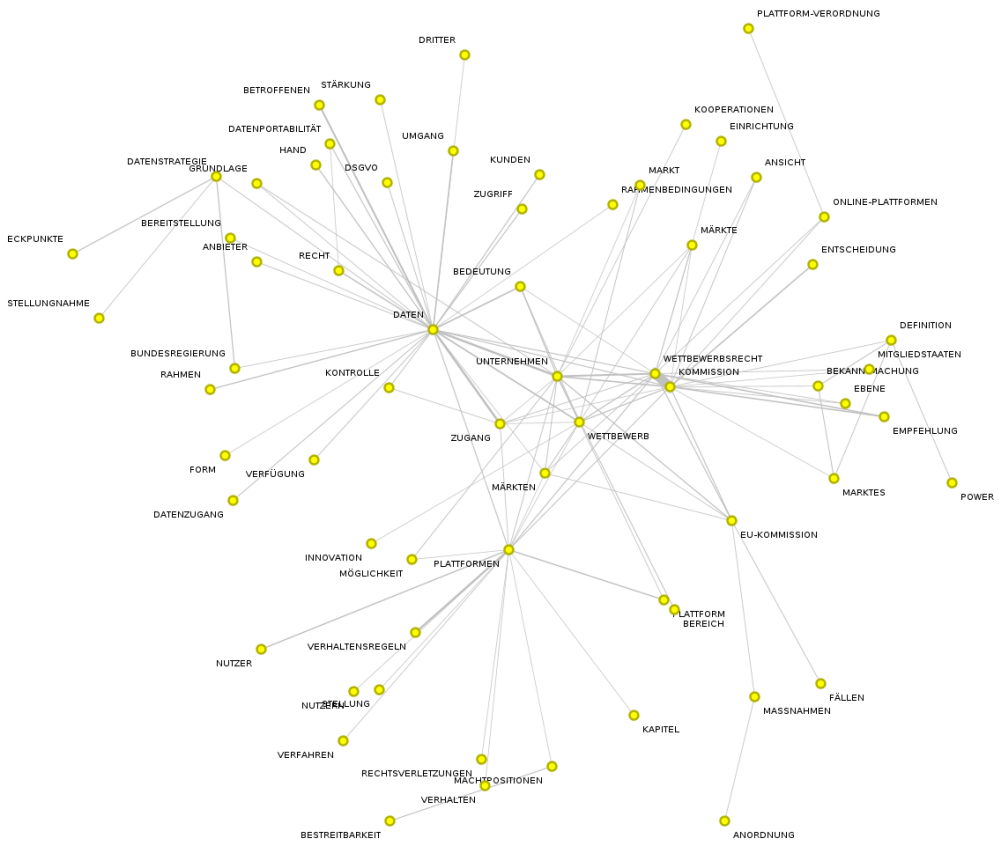

ABBILDUNG 3.12: Cooccurrence-Network: Größter Komponent

3.2.3 Diskussion

Die Platzierung der Knoten auf der Fläche sagt nichts über eine semantische Nähe o.Ä. aus. Die Anordnung des Netzwerks erfolgt allein aufgrund des Kamada-Kawai-*Layout* (Kamada, Kawai et al., 1989) und entbehrt daher jeglicher interpretierbaren Grundlage.

Zudem behandelt die Analyse alle Wörter gleich und nicht alle Dokumente gleich. Um politische Aussagen aus Visualisierungen zu ziehen, ist es wichtig zu beachten, dass längere Berichte oder Stellungnahmen die Visualisierungen mehr beeinflussen, als kurze Stellungnahmen. Dies zu berücksichtigen wäre Gegenstand weiterer Forschung.

Im Gegensatz zur Diskussion in 3.1.4 können *Co-Occurrence-Networks* eine Orientierungshilfe für einen Einstieg in Debatten sein. Aufgrund der mangelnden Übersichtlichkeit ist jedoch das Argument, dass alles auf einen Blick

ersichtlich ist, hier nicht gültig. Häufig beinhalten die Darstellungen zu viele Informationen gleichzeitig und so beschränkt sich die Erkenntnis des Betrachters auf die Schlussfolgerung, dass es sich um ein komplexes Thema handelt, in welchem alle Themen auf komplizierte Art und Weise zusammenhängen. Ein *Co-Occurrence-Network* liefert eher Fachexperten oder in diesem Thema Belesenen Informationen, indem es nach intensiver Betrachtung, z.B. durch ungeahnte Kanten zwischen zwei Wörtern, Hinweise auf engere Zusammenhänge als angenommen geben kann.

3.3 Klassifizierung von Dokumenten

Ein zentrales Element von Bürokratien ist die Einordnung. Die Welt wird in viele kleine Bereiche unterteilt, von der Aufteilung von Themenblöcken zwischen Ministerien über deren Aufteilungen in Abteilungen bis in die Fachthemenreferate und spezielle Bereiche von Fachthemen, welche von einzelnen Mitarbeiterinnen oder Mitarbeitern abgedeckt wird. Nur derartig, mit dem Prinzip der Arbeitsteilung, kann sich eine Regierung mit allen Fragen des öffentlichen und privaten Lebens auseinander setzen und dieses annhähernd allumfassende Themenspektrum abdecken. Andere Ansätze für die Organisation des öffentlichen Lebens gibt es, aber die Bürokratie ist mit dem Prinzip der Kategorisierung und Unterteilung so gewachsen, wie sie heute ist.

Das Prinzip der Zuordnung wird bei allen aufkommenden Themen von Staatsangestellten regelmäßig angewandt. Sei es eine Anfrage eines Bürgers zu einem speziellen Thema, welche an Fachexperten weitergeleitet werden muss, sei es ein neues zu regulierendes Themenfeld oder seien es Fragen des Parlaments oder der Presse an die Bundesregierung: Immer wieder müssen Themen zugeordnet werden.

Text-Klassifizierungsalgorithmen können die Anwendung dieses Prinzips beschleunigen und raffinieren. Es kommt häufiger vor, dass eine Email an falsche Fachexperten weitergeleitet wird, die sich gar nicht mit dem betreffenden Thema befasst. Das führt meist zu Verzögerungen, wenn nicht auch in Einzelfällen auch zu Kompetenz-Streitigkeiten. Im Nachfolgenden wird ein Beispiel-Anwendungstest verschiedener Klassifizierungsalgorithmen auf Texten der öffentlichen Verwaltung durchgeführt.

3.3.1 Einleitung

Ursprünglich sollte auf dem bisherigen Korpus der verschiedenen Stellungnahmen zur Daten-Strategie auch dieser Prototpy basieren. Allerdings hat sich dies aufgrund der Größe des für das Training nutzbaren Datensatzes, welche aus Datenschutz-technischen Gründen nur die öffentlich-zugänglichen Stellungnahmen umfasste, nicht angeboten. Weil sich der Datensatz der kleinen Anfragen mit einem realen *Use-Case* aus dem Regierungsalltag anbot, wurde der nachfolgende Prototyp auf diesem Korpus aufgebaut.

Kleine Anfragen gehören zum täglichen Geschäft aller Ministerien. Das Parlament hat das Recht, das Agieren der Bundesregierung zu hinterfragen. Für Abgeordnete in Oppositionsfraktionen ist die kleine Anfrage die Möglichkeit, Missstände zu hinterfragen und mangelnde Antworten öffentlichkeitswirksam zu verwerten. Über den Parlamentspräsidenten werden die kleinen Anfragen an die Regierung, also zunächst das Bundeskanzleramt weitergeleitet. Dort entscheidet eine Mitarbeiterin/ ein Mitarbeiter darüber, welches Ressort für die Beantwortung zuständig ist und leitet die Anfrage dementsprechend weiter.

Der nachfolgende Prototyp beschäftigt sich mit der Frage, ob diese Zuordnung mithilfe von Klassifizierungsalgorithmen algorithmisch unterstützt und teil-automatisiert werden kann.

3.3.2 Datensatz

Dank *kleineanfragen.de* (OpenKnowledgeFoundation, 2019) gibt es einen Datensatz aller kleinen Anfragen an die Bundesregierung und ihre Antworten maschinen-verarbeitbar online. Wie in der nachfolgenden Abbildung zu sehen ist, hat jede kleine Anfrage eine Drucksachennummer.

Deutscher Bundestag **Drucksache 19/19878**

19. Wahlperiode 11.06.2020

Antwort

der Bundesregierung

auf die Kleine Anfrage der Abgeordneten Matthias Seestern-Pauly, Britta Katharina Dassler, Katja Suding, weiterer Abgeordneter und der Fraktion der FDP – Drucksache 19/19618 –

Digitale Bildungsinhalte in der frühkindlichen Bildung

Vorbemerkung der Fragesteller

In der Phase der familienergänzenden Kindertagesbetreuung (bei den Drei- bis unter Sechsjährigen) ist die familienergänzende, institutionelle Kindertagesbetreuung in Deutschland längst selbstverständlich geworden; rund 95 Prozent der Kinder besuchen in dieser Phase eine Kindertageseinrichtung (vgl. Nationaler Bildungsbericht 2018, S. 64 ff.)

ABBILDUNG 3.13: PDF-Version einer kleinen Anfrage. Verändert nach (OpenKnowledgeFoundation, 2019)

Da *kleineanfragen.de* die Anfragen in .txt-Format[9] zur Verfügung stellt, wurden die Daten durch einfache Iteration (und Überprüfung des HTTP-Statuscode auf den 200-Status) über die Drucksachennummer in eine CSV-Datei eingelesen. So wurden Datensätze verschiedener Größen heruntergeladen. Die Daten liegen dann in dieser Form vor:

S Titel	S Datum	S Fragesteller	S Antwortgeber	S Anfrage
Förderprogramme mit dem Sc...	2018-03-07	Die Linke	Bundesministerium für Wirtschaft und En...	Die Antwort wurde namens der Bundesregierung mit Schreiben des Bur 2. März 2018 übermittelt. Die Drucksache enthält zusätzlich – in kleinerer Schrifttype – den Fraget Deutscher Bundestag Drucksache 19/1101 19. Wahlperiode 07.03.2018
Export von Kleinwaffen in 2017	2018-03-07	Die Linke	Bundesministerium für Wirtschaft und En...	Die Antwort wurde namens der Bundesregierung mit Schreiben des Bur 5. März 2018 übermittelt. Die Drucksache enthält zusätzlich – in kleinerer Schrifttype – den Fraget Deutscher Bundestag Drucksache 19/1102 19. Wahlperiode 07.03.2018
Umfang und Prüfung von ausl...	2018-03-07	Bündnis 90/Die Grünen	Bundesministerium für Wirtschaft und En...	Die Antwort wurde namens der Bundesregierung mit Schreiben des Bur 27. Februar 2018 übermittelt. Die Drucksache enthält zusätzlich – in kleinerer Schrifttype – den Fraget Deutscher Bundestag Drucksache 19/1103 19. Wahlperiode 07.03.2018
Versorgungssicherheit in Deut...	2018-03-07	AfD	Bundesministerium für Wirtschaft und En...	Die Antwort wurde namens der Bundesregierung mit Schreiben des Bur 26. Februar 2018 übermittelt. Die Drucksache enthält zusätzlich – in kleinerer Schrifttype – den Fraget Deutscher Bundestag Drucksache 19/1104 19. Wahlperiode 07.03.2018

ABBILDUNG 3.14: Auszug aus dem Datensatz der kleinen Anfragen

[9] erreichbar unter diesem Link mit einem Platzhalter für die jeweilige <Drucksachennummer>: https://kleineanfragen.de/bundestag/19/<Drucksachennummer>.txt

Das Feld Anfrage beinhaltet den gesamten Text der Anfrage, welche meist mehrere Seiten Text umfasst.
Es wurden Daten*Samples* verschiedener Größen heruntergeladen: Von einem kleinen *Snippet* von 1,9 MB mit 50 zufälligen Anfragen bis zum gesamten Datensatz aller Anfragen seit Dezember 2017 von 261 MB, was 6089 Anfragen entspricht.
Eine Erkundung des Datensatzes zeigt, welches Ministerium oder staatliche Stelle wie viele Anfragen erhalten haben. Eine Analyse des Datensatzes aller Anfragen des Jahres 2019 gibt einige Vorabanalysen preis. Die Top-10-Ministerien sind hier dargestellt:

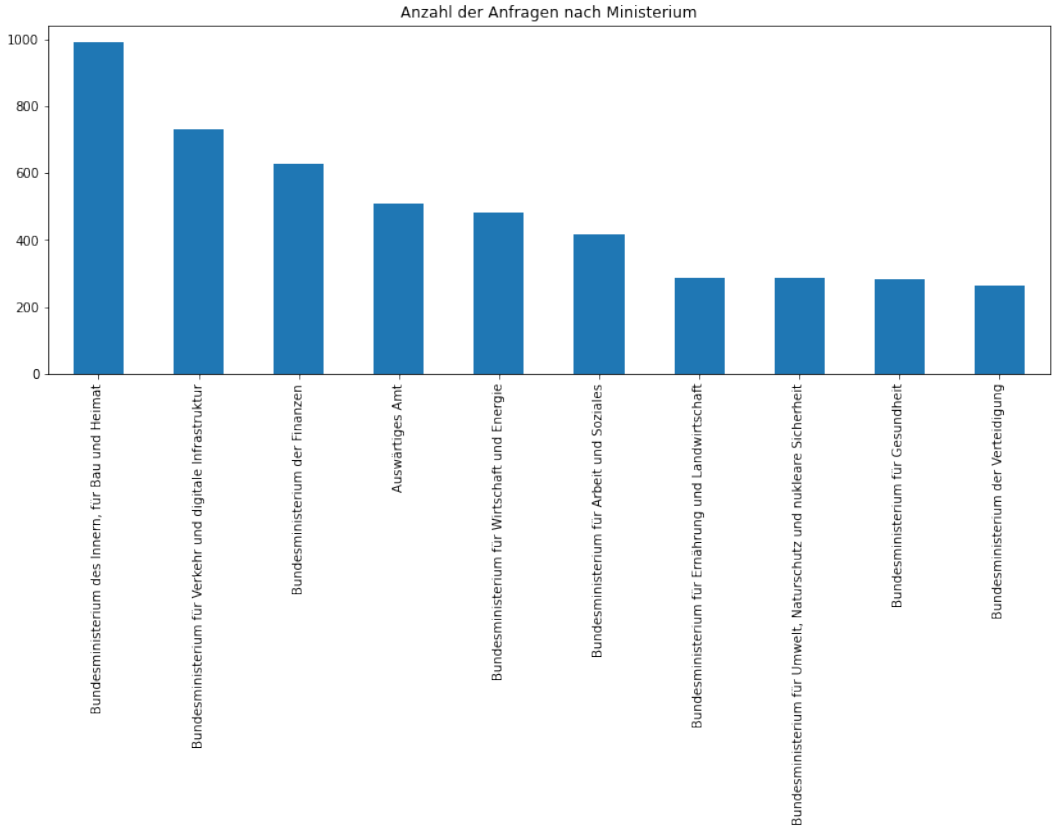

ABBILDUNG 3.15: Die Top-10-Ministerien nach erhaltenen kleinen Anfragen

Im Nachfolgenden wird für die Analse der *Support-Vector-Machines*, des *Decision-Tree* und des *k-Nearest-Neighbor-Classifiers* ein Datensatz mit 716 kleinen Anfragen und für das Training des neuronalen Netzes ein Datensatz mit 6090 kleinen Anfragen beispielhaft herangezogen. Diese Datensätze wurden aus maschinellen Leistungsbeschränkungen ausgewählt. Um dem Algorithmus die Erkennung zu erschweren, wurden die Namen der antwortenden Ministerien / staatlichen Stellen, also das Ziel der Klassifizierung, aus dem

Text entfernt. Weil sonst zwischen Frage und Antwort weder thematischen noch stilistischen Differenzen bei einer qualitativen Prüfung feststellbar waren, wurde darauf verzichtet, die thematisch ähnlichen Antworten von den Fragen zu trennen. Es ist das Thema anschließender Forschung, festzustellen, wie groß die Auswirkungen auf das trainierte Modell tatsächlich sind.

3.3.3 Klassifizierung mit herkömmlichen Methoden des maschinellen Lernens

Algorithmus

Zuerst wird der Datensatz eingelesen, in das von KNIME präferierte Dokument-Format konvertiert und die Spalten Titel, Antwortgeber und Text ausgewählt. Im *Preprocessing* werden Satzzeichen entfernt, der Stamm der Wörter gebildet (Snowball-Stemmer[10]), Zahlen herausgefiltert, alle Wörter unter 5 Buchstaben entfernt, alles zu Kleinschreibung konvertiert und Sonderzeichen und Stopwords entfernt. Anschließend wurden sowohl 1-*Grams*, also Einzelwörter, als auch 1- und 2-*Grams*, also Einzelwörter und Paare von Wörtern, die im Text nebeneinander vorkommen, separat untersucht. 1-*Gram* Mithilfe des *Keygraph-Keyword-Extractor* werden pro Dokument 25 zentrale Begriffe ausgewählt. In den Anfangsgraphen wurden die 30 häufigsten Wörter aufgenommen. Die 12 Wörter mit dem größten *Key-Score* wurden dem Graphen anschließend hinzugefügt, nähere Erläuterungen siehe 2.5. Jeder dieser Begriffe bekommt eine Dimension in einem Vektorraum zugewiesen, sodass, wenn ein Dokument über ein Wort verfügt, der Wert auf der entsprechenden Achse 1 und nicht 0 ist und so jedem Dokument ein Punkt in einem hochdimensionalen Vektorraum zugewiesen wird. Es folgt eine Partitionierung des Datensatzes in einen Test- und Trainingsdatensatz mit einem Verhältnis von 70 zu 30 und eine Angleichung der Kohorten mithilfe von stratified sampling (Sampling aus homogenen Untergruppen anstelle aus dem gesamten Datensatzes). Im Anschluss werden die Modelle trainiert, siehe unten.

1- und 2-*Grams* Zunächst werden, ähnlich wie oben, 1-*Grams* aus dem Text extrahiert. Nun werden zusätzlich alle 2-*Grams* erstellt und alle, die weniger als 15 Mal in einem Dokument vorkommen, entfernt. Nun bekommt jedes 1- als auch 2-*Gram* eine Dimension in einem Vektorraum zugewiesen. Ein Dokument bekommt dann analog zu oben einen Punkt in diesem Vektorraum

[10] Für weitere Informationen, siehe https://snowballstem.org/

aufgrund seiner Wortvorkommnisse zugewiesen. Anders als oben wird allerdings die Frequenz eines 1- oder 2-*Grams* mit berücksichtigt, sodass dann das Dokument in der entsprechenden Dimension nicht nur die Werte 0 und 1 haben kann, sondern einfach die Anzahl, wie häufig das *Gram* vorkommt, als Achsenwert abgebildet wird:

Document	D schwer	D abbiegeassistenzsystem	D bundesregier	D verpfli...	D antwort	D druck...	D deutsch
"abbiegeassistenzsystem schwer lastkraftwa...	48	34	32	19	15	9	9
"abfindungszahl millionenhoh ehemal vorsta...	0	0	37	0	3	4	139
"abfluss mittel bundesforderprogramm breit...	0	0	9	0	0	21	8
"abgas schiff insbesond seeschiff"	0	0	17	0	0	3	14
"abgeruf mittel breitbandausbau"	0	0	0	0	0	20	8
"abgeruf mittel breitbandausbau"	0	0	1	0	0	16	8
"abgeruf mittel breitbandausbau"	0	0	0	0	0	36	2
"abgeruf mittel breitbandausbau baden-wur...	0	0	4	0	0	26	8
"abgeruf mittel breitbandausbau brandenbu...	0	0	4	0	0	8	5
"abgeruf mittel breitbandausbau mecklenbu...	0	0	0	0	0	12	8
"abgeruf mittel breitbandausbau rheinland-...	0	0	4	0	0	7	5

ABBILDUNG 3.16: *Embedding* einzelner Anfragen

Dies ist bei den Prozessen unterschiedlich, weil mit der jeweilig angewandten Methode eine bessere Performance zu beobachten war.

Dann wird wieder der Datensatz in einen Trainings- und Testteil unterteilt.

Für beide Methoden gilt, dass im Anschluss die Klassifizierung mit einem *Decision-Tree*, *Support-Vector-Machines* und K-*Nearest-Neigbor* (K=3) durchgeführt wird.

Resultate

Der Datensatz mit 716 kleinen Anfragen wurde vor allem aus Kapazitäts-Gründen nun wie oben beschrieben ausgewertet. Kleinere Datensätze wurden ebenfalls ausgewertet, weil aber mit der Größe des Datensatzes auch eine Performanz-Steigerung zu beobachten war, wird im Nachfolgenden nur die Analyse des genannten Datensatzes vorgestellt.
Anzumerken ist, dass es ingesamt 21 Klassen gab (Ministerien bzw. staatliche Stellen). Das bedeutet, dass eine zufällige Zuteilung eines *Samples* mit 4,8% Wahrscheinlichkeit bei einer angenommenen Gleichverteilung korrekt ist. Alles, was eine bessere Accuracy als 4,8% hat, ist also besser als der Zufall.

Classification Algorithms: Performance in %	
Algorithm	**Accuracy**
Decision-Tree (1-*Gram*)	32,1
Decision-Tree (2-*Gram*)	54,4
Support Vector Machines (1-*Gram*)	42,8
Support Vector Machines (2-*Gram*)	54,9
K-Nearest Neighbor (1-*Gram*)	23,3
K-Nearest Neighbor (2-*Gram*)	34,9

Alle Methoden basierend auf 1- und/oder 2-*Grams* zeigten eine bessere Performance bei 1- und 2-*Grams* (in der Tabelle verkürzt nur mit 2-*Gram* bezeichnet) als bei den jeweilig entsprechenden 1-*Grams*-Trainingsprozessen. Da diese Ergebnisse zwar deutlich besser als eine zufällige Zuweisung sind, aber dennoch noch viel Potential bezüglich ihrer Performanz besitzen, wird nun ein Ansatz mittels *Deep-Learning* untersucht.

3.3.4 Klassifizierung mittels *Deep-Learning*

Parallel zu dem obigen Ansatz wird ebenfalls die Wirkungsweise von Künstlicher Intelligenz bei Anwendung auf dieses Problem untersucht.

Algorithmus

Nach Entfernen von Vorkommnissen des antwortenden Ministeriums oder der staatlichen Stelle in der Anfrage und dem Löschen von fehlerbehafteten Zeilen werden für jede Anfrage das antwortende Ministerium in eine eindeutige Zahl umgewandelt, um die Klassifikation im späteren Verlauf zu ermöglichen. Die *Buffer-Size* wird auf 50000 gesetzt. Diese gibt in dem randomisierten Zieh-Prozess bezüglich der neuen Daten-Reihenfolge an, aus wie vielen Elementen das nächste gezogen wird. Die *Batch-Size* wird auf 4 gesetzt. Sie unterteilt die Anfragen in 4-er Gruppen, welche gemeinsam durch den *Forward-Propagation*-Prozess geschickt werden. Nach der Bewältigung der 4 Dokumente wird der nächste Iterationsschritt der Optimierung durchgeführt, also die Gewichte im Netz durch den Gradienten weiter optimiert. Die *Validation-Size* wird auf rund ein Zehntel des gesamten Datensatzes gesetzt. Sie dient, genau wie die *Test-Size*, die auf ein Fünftel gesetzt wird, der Unterteilung in Test-, Trainings- und Validations-Datensatz. Die Größe des Vokabulars ist 10000 und die maximale Sequenz-Länge wird auf 25000 gesetzt. Letztere wird deswegen so hoch gesetzt, weil die Länge von kleinen Anfragen sehr variieren kann und ein Abschneiden der Anfrage durch eine

zu kleine Sequenz-Länge vermieden werden soll. Wenn eine Anfrage nicht über 25000 Wörter verfügt, werden *Pads*, also Platzhalter, eingefügt.

Im Anschluss wird der Datensatz nach der *Buffer-Size* durchmischt. Für die Standardisierung, Tokenisierung, die Erstellung eines Vokabulars und um *Tokens* auf Ganzzahlen abzubilden, wird die [tf.text]-API[11] verwendet. Die Abbildung auf die Ganzzahlen, auch *Vectorization* wird benötigt, weil jedes Wort in den Texten durch eine Ganzzahl ersetzt wird, damit diese Zahlenreihen dann als *Input* für das Modell dienen können.

Im Weiteren wird das Vokabular gebildet, indem die *Tokens* nach Häufigkeit sortiert und die häufigsten *Tokens* für die Analyse übernommen werden.

Erst dann wird der Datensatz in einen Trainigs- und in einen Validations-Anteil segmentiert. Das *Keras-Layer Text-Vectorization* bildet *Batches* und führt das *Padding* der vektorisierten Daten durch. Das *Padding* ist erforderlich, wenn Texte kürzer sind als die maximale Sequenz-Länge von 25000 Wörtern. Dann werden die restlichen Elemente mit einem Platzhalter aufgefüllt.

Die Tabelle wird nun in einen 80%-igen Trainings-, einen 10%-igen Validations- und in einen 20%-igen Test-Teil aufgetrennt.

Die eben gebildeten Datensätze sind keine Sammlungen an Beispiel-Label-Paaren, sondern an *Batches*. Jedes *Batch* ist ein Paar von vielen Texten - viele *Labels*, die als *Arrays* dargestellt werden.

Danach folgt die Modelldefinition. Das Modell umfasst

1. ein *Embedding-Layer* mit 64 Dimensionen und Maskierung, welche die große Anzahl an *Pads*, die für eine Angleichung der Längen der Anfragen entscheidend war, maskiert, sodass sie im weiteren Verlauf ignoriert werden,

2. ein *1-D-Convolution-Layer* mit *Kernel-Size* von 5, RuLU-Aktivierungsfunktion, *Strides* von 2 und einem Padding, welches keine zusätzlichen Rand-Werte hinzufügt, sondern eine Reduzierung der Dimensionen in Kauf nimmt. Zusätzlich umfasst das Model

3. ein *Global-MaxPooling* in einer Dimension sowie

4. ein volles *Layer*, welches für jede Klasse ein Neuron umfasst, sodass für alle Klassen eine Wahrscheinlichkeit errechnet wird.

Für die Kompilierung des Modells wird der Optimierer 'Adam' ausgewählt. Die *Loss*-Funktion ist die *sparseCategoricalCrossentropy*. Final wird das Modell 10 Epochen lang trainiert und mithilfe des *Test-Set* ausgewertet. Zu guter

[11] https://www.tensorflow.org/tutorials/tensorflow$_{t}ext/intro$

Letzt wird das Modell exportiert und steht so bereit, auf weitere Datensätze angewendet zu werden.

Resultate

Weil 26 Ministerien / staatliche Stellen im Datensatz vorkommen, ist alles, was also besser als 3,8% *Accuracy* ist, besser als der Zufall.

Die*Accuracy* beim Training mit dem Datensatz mit allen kleinen Anfragen zum Jahre 2019 erreicht 74,89%. Die Ergebnisse des neuronalen Netzes übertreffen also die der oben diskutierten herkömmlichen Ansätze.

3.3.5 Diskussion

Die Ergebnisse des *K-Nearest-Neigbor*-Algorithmus sind aufgrund des sehr generellen und wenig differenzierten Ansatzes bei dieser komplexen Fragestellung nicht unerwartet. Besser schneiden die *Decision Trees* ab. Bei diesen ist wahrscheinlich die große Anzahl an Variablen im Vergleich zu wenigen Datenpunkten der Grund für Leistungseinbußen. Es ist nachvollziehbar, warum *Support-Vector-Machines* mit ihrem Raumtrennungsansatz anhand von Hyperebenen bei dieser konkreten Aufgabenstellung am besten abschneiden.
Der verwendete *Deep-Learning*-Ansatz ist hingegen auf das konkrete Ziel hin ausgerichtet. Die Verwendung der im Vergleich zu den anderen Ansätzen größeren Menge an Parametern lässt zudem auch eine bessere Leistung dieses Ansatzes erwarten.
Aufgrund von technischen Beschränkungen konnten keine komplexeren Modelle mit mehr und weiteren andersartigen *Layers* trainiert und getestet werden. Dies hätte möglicherweise zu noch besseren Ergebnissen geführt.

Das oben angesprochene Prinzip der Unterteilung, welches ein grundlegendes Merkmal von Verwaltungen ist, bietet auf jeden Fall viele Anwendungsmöglichkeiten von Klassifizierungsalgorithmen in diesem Bereich. Die Chancen auf einheitlichere, einfachere und vor allem schnellere Bearbeitungen sind gerade für die öffentliche Verwaltung verlockend.

Gleichzeitig ist bei der Einführung derartiger Algorithmen auch immer mit Skepsis zu rechnen. Die Gefahr des Verlusts des eigenen Arbeitsplatzes ist im öffentlichen Sektor häufig eher weniger gegeben, was allerdings immer

noch Raum für Ängste und Vorbehalte bietet. Die Perspektive auf eine Verwaltung, in der viele kleine Entscheidungen von Maschinen getroffen werden, ist für viele im Allgemeinen dystopisch. Um diesem Rechnung zu tragen und auch um rechtlichen Problemen entgegenzutreten, ist eine algorithmische Unterstützung bis hin zu einer Teil-Automatisierung denkbar. Dies wirkt Ängsten entgegen und belässt die Letztentscheidung beim Menschen, was die Dystopie einer unmenschlichen und Maschinell-gelenkten Verwaltung ausschließt.

3.3.6 Fazit

In Kapitel 3.3 wurde in einem prototypischen Anwendungstest die Machbarkeit gezeigt, dass die Zuteilung von kleinen Anfragen zum jeweiligen antwortenden Ministerium teil-automatisiert bzw. algorithmisch unterstützt werden kann. Weitere Forschungsarbeit ist nötig: Um die *Accuracy* weiter zu steigern, können noch weitere Klassifikationsalgorithmen mit einem größeren Datensatz getestet werden. Im Nachfolgenden könnte eine Studie in der Praxis gemeinsam mit dem zuständigen Referat im Bundeskanzleramt als erster Testlauf gestartet werden, welcher den Fokus auf die Akzeptanz und die praktische Anwendbarkeit des Algorithmus legen kann.

Kapitel 4

Fazit und Ausblick

Ich möchte auf Carl Runges Zitat zurückkommen: Diese Arbeit zeigt anhand einiger Beispiele, welche Möglichkeiten es im öffentlichen Sektor gibt, um sich zu bücken, Probleme zu ergreifen und zu lösen:
Das Trendgebirge als Zusammenfassungs- und Darstellungsform von Stellungnahmen, Strategien, Auslandskorrespondenzen und Förderanträgen kann bei Zeitknappheit dem Leser helfen, die wichtigsten Schlagworte und deren semantische Nähe zu erfassen. Darüber hinaus ermöglicht es das *Co-Ocurrence-Network*, die Zusammenhänge zwischen Schlagwörtern und Themen-*Cluster* auf einen Blick sichtbar zu machen. Das Klassifizierungsbeispiel mit kleinen Anfragen macht deutlich, welche Möglichkeiten das Prinzip der Klassifizierung gerade in der hierarchischen Organisationsform von Verwaltungen hat.

Die Vorteile der Technologien sind offensichtlich:
Mit der Analyse und Visualisierung von Datenmengen ist es möglich, politische Entscheidungen mit Hilfe einer fundierten Datenbasis zu unterstützen. Eine Erhöhung der Geschwindigkeit der Arbeitsprozesse sowie mehr Freiraum für die Angestellten, Beamtinnen und Beamten ist zu erwarten. Der Freiheitsgewinn wird nicht von jedem als positiv gewertet. Der öffentliche Sektor bietet sichere Arbeitsverhältnisse und zieht nach persönlicher Erfahrung des Autors Menschen an, welche die Sicherheit hoch schätzen. Mit der Angst, eine automatisierbare und ersetzbare Arbeit zu verlieren, muss umsichtig umgegangen werden.
Der Vorteil für die Einführung von neuen Technologien im öffentlichen Sektor ist, dass das Argument, dass Arbeitsplätze 'wegrationalisiert' werden, aufgrund der Unkündbarkeit von großen Teilen der Belegschaft sehr eingeschränkt gilt. Das erleichtert das dortige Werben für eine erweiterte Anwendung. Selbstverständlich kann der Staat, indem er Stellen nicht erneut besetzen muss, auf lange Sicht zusätzlich finanzielle Ressourcen sparen und

die Mittel anderweitig einsetzen.
Zusätzlich ist eine Vereinheitlichung zu erwarten. Allgemeine Datenbanken, die politische Statements wie beispielsweise Sprechpunkte von Sprechzetteln aller Referate beinhalten, können inhaltliche Differenzen oder entgegengesetzte Formulierungen verschiedener Referate zu Tage treten lassen. Je früher Konflikte identifiziert werden, desto weniger zeitaufwändig ist tendenziell das Ausräumen der Meinungsdifferenz.
Die Vorteile sind für den Staat, aber auch für Unternehmen, die sich derartiger Probleme annehmen, auf lokaler, regionaler, Landes-, Bundes-, supra- und internationaler Ebene groß.

Ein starker Wille kann Berge versetzen. Gleichzeitig bleiben noch viele Herausforderungen bei der Einführung der Technologien in Verwaltungen zu bewältigen.
Die größte Herausforderung ist eine generelle Skepsis gegenüber den neuen Methoden. Teilweise muss noch mehr Verständnis bezüglich der Funktionsweise der jeweiligen Algorithmen geschaffen werden. Wissensvertiefungen und interne kritische Diskussionen in Workshops können dazu beitragen. Das Argument, dass die Analyse der Daten keine wirklich neuen Erkenntnisgewinne liefert, welches bei Fachexperten, die sich in speziellen Debatten sehr gut auskennen und welche mit übersichtsschaffenden Analysen konfrontiert werden, durchaus begründet sein kann, kann sogar generell der Einführung der Technologien entgegenwirken. Diesem Argument muss mit weiteren Erklärungen über die genaue Wirkungsweise und Ziele der verwendeten Algorithmen entgegen getreten werden.

Es gibt aber noch weitere Herausforderungen. Interne Vorschriften, dass gewisse Datensätze bei verschiedenen staatlichen Institutionen nicht zusammengeführt werden dürfen, müssen überprüft werden, sodass keine Prozesse unnötig verlangsamt werden.
Zusätzlich sind für viele Analysen die Anonymisierung und Synthetisierung von Datensätzen ein Grundbaustein. Dieser sorgt aber natürlich für einen Mehraufwand an Zeit und Ressourcen. Vor der Analyse von Datensätzen z.B. durch externe Partner könnten beim Staat angestellte Experten vor Herausgabe der Daten einen Standard an Anonymisierung und Synthetisierung sicherstellen. Diese könnten zusätzlich anderen Stellen innerhalb des öffentlichen Sektors helfen, die zu analysierenden Daten zu anonymisieren.

Grundsätzlich wiegen die Chancen der Anwendung von NLP im öffentlichen Sektor die Herausforderungen auf, zumal der öffentliche Druck der Gesellschaft auf die Verwaltung, sich zu modernisieren, mit hoher Wahrscheinlichkeit weiter zunehmen wird. Es wird sich in naher Zukunft zeigen, ob sich die staatlichen Stellen an das schnelle Tempo der Treiber der Digitalisierung, welche zunehmend Einfluss auf unseren Alltag ausüben, anpassen können. Diese Arbeit soll ein kleiner Schritt in Richtung einer modernen, effektiven und digitalisierten Verwaltung darstellen.

Literatur

Abu-Shanab, E. & Harb, Y. (2019). E-government research insights: Text mining analysis. *Electronic Commerce Research and Applications*, *38*.

Alto, V. (2019). *An intuitive introduction to Support Vector Machine*. Online. Abgerufen von https://medium.com/swlh/support-vector-machine-from-scratch-ce095a47dc5c am 24.12.2020.

Berthold, M., Cebron, N., Dill, F., Gabriel, T., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K. & Wiswedel, B. KNIME: The Konstanz Information Miner. In: *Studies in Classification, Data Analysis, and Knowledge Organization*. Heidelberg-Berlin: Springer, 2008, 319–326. ISBN: 978-3-540-78239-1.

Berwick, R. (2008). An Idiot's guide to Support Vector Machines (SVMs) [MIT Class] [pp.17–18]. http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Heidelberg: Springer.

Blei, D., Ng, A. & Jordan, M. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993

Bohman, S. Data Visualization: An Untapped Potential for Political Participation and Civic Engagement (A. Ko & E. Francesconi, Hrsg.). In: In *Electronic Government and the Information Systems Perspective* (A. Ko & E. Francesconi, Hrsg.). Hrsg. von Ko, A. & Francesconi, E. Cham: Springer International Publishing, 2015, 302–315. ISBN: 978-3-319-22389-6.

Datenethikkommission. (2019). *Gutachten der Datenethikkommission*. Bundesministerium des Innern, für Bau und Heimat. https://www.bmi.bund.de/SharedDocs/downloads/DE/publikationen/themen/it-digitalpolitik/gutachten-datenethikkommission.pdf?__blob=publicationFile&v=5

Dewi, A. & Thiel, K. (2017). *Topic Extraction: Optimizing the Number of Topics with the Elbow Method*. Online. Abgerufen von https://www.knime.com/blog/topic-extraction-optimizing-the-number-of-topics-with-the-elbow-method am 24.12.2020.

Grosch, D. (2018). *Das Gebirge der Metatrends*. Online. Abgerufen von https://www.oeffentlich it.de/trendgebirge am 24.12.2020.

Habich, R, Himmelreicher, R. K. & Huschka, D. (2010). *Zur Entwicklung der Dateninfrastruktur in Deutschland* (RatSWD Working Papers Nr. 157). German Data Forum (RatSWD). https://EconPapers.repec.org/RePEc:rsw:rswwps:rswwps157

Hagen, L., Uzuner, Kotfila, C., Harrison, T. M. & Lamanna, D. Understanding Citizens' Direct Policy Suggestions to the Federal Government: A Natural Language Processing and Topic Modeling Approach. In: *48th Hawaii International Conference on System Sciences*. 2015, 2134–2143. https://doi.org/10.1109/HICSS.2015.257.

Hegelich, S. (2016). Decision Trees and Random Forests: Machine Learning Techniques to Classify Rare Events. *European Policy Analysis*, *2*(1), 107–108. https://doi.org/https://doi.org/10.18278/epa.2.1.7

Hucker, M. (2020). *An intuitive introduction to Support Vector Machines*. Online. Abgerufen von https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02 am 24.12.2020.

Iriberri, A. & Leroy, G. Natural Language Processing and e-Government: Extracting Reusable Crime Report Information. In: *IEEE International Conference on Information Reuse and Integration*. 2007, 221–226. https://doi.org/10.1109/IRI.2007.4296624.

Kamada, T., Kawai, S. et al. (1989). An algorithm for drawing general undirected graphs. *Information processing letters*, *31*(1), 7–15.

Kiela, D. & Bottou, L. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In: *Proceedings of the 2014 Conference on empirical methods in natural language processing (EMNLP)*. 2014, 36–45.

Koehrsen, W. (2019). *Neural Network Embeddings Explained*. Online. Abgerufen von https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526 am 24.12.2020.

Kommission-Wettbewerbsrecht-4.0. (2019). *Ein neuer Wettbewerbsrahmen für die Digitalwirtschaft: Bericht der Kommission Wettbewerbsrecht 4.0*. Bundesministerium für Wirtschaft und Energie. https://www.bmwi.de/Redaktion/DE/Publikationen/Wirtschaft/be-richt-der-kommission-wettbewerbsrecht-4-0.pdf?__blob=publicationFi-le&v=10

Krawczak, M & Weichert, T. (2017). *Vorschlag einer modernen Dateninfrastruktur für die medizinische Forschung in Deutschland (Version 1.9)*. Christian-Albrechts-Universität zu Kiel und Netzwerk Datenschutzexpertise. https://www.uni-kiel.de/medinfo/documents/

Kwon, N., Shulman, S. & Hovy, E. Multidimensional text analysis for eRulemaking. In: *Proceedings of the 2006 international conference on Digital government research*. 2006, 157–166.

Li, K. R. X. D., F. (2020a). CS231n: Convolutional Neural Networks for Visual Recognition [Stanford Class]. https://cs231n.github.io/neural-networks-1/

Li, K. R. X. D., F. (2020b). CS231n: Convolutional Neural Networks for Visual Recognition [Stanford Class]. https://cs231n.github.io/optimization-2/

Li, K. R. X. D., F. (2020c). CS231n: Convolutional Neural Networks for Visual Recognition [Stanford Class]. https://cs231n.github.io/convolutional-networks/

Ludwig, J. (2020). *Image Convolution*. Online. Abgerufen von http://web.pdx.edu/jduh/courses/Archive/geog481w07/Students/Ludwig_ImageConvolution.pdf am 24.12.2020.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint*, 2–4.

Ohsawa, Y., Benson, N. E. & Yachida, M. KeyGraph: automatic indexing by co-occurrence graph based on building construction metaphor. In: *Proceedings IEEE International Forum on Research and Technology Advances in Digital Libraries -ADL'98-*. 1998, 12–18. https://doi.org/10.1109/ADL.1998.670375.

OpenKnowledgeFoundation. (2019). *KleineAnfragen.de*. Online. Abgerufen von https://kleineanfragen.de/ am 24.12.2020.

O'Neill, K., Weinthal, E. & Hunnicutt, P. (2017). Seeing complexity: visualization tools in global environmental politics and governance. *Journal of Environmental Studies and Sciences*, *7*(4), 490–506.

Papadopoulos, T. & Charalabidis, Y. What do governments plan in the field of artificial intelligence? Analysing national AI strategies using NLP. In: *Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance*. 2020, 100–111.

Quinn, K., Monroe, B., Colaresi, M., Crespin, M. H. & Radev, D. (2010). How to Analyze Political Attention with Minimal Assumptions and Costs.

American Journal of Political Science, *54*(1), 209–228. https://doi.org/https://doi.org/10.1111/j.1540-5907.2009.00427.x

Rao, G. & Dey, S. (2011). Decision Support for e-Governance: A Text Mining Approach. *CoRR*, *abs/1108.6198*. http://arxiv.org/abs/1108.6198

Stylios, G., Christodoulakis, D., Besharat, J., Vonitsanou, M., Kotrotsos, I., Koumpouri, A. & Stamou, S. (2010). Public opinion mining for governmental decisions. *Electronic Journal of e-Government*, *8*(2), 203–214.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Going deeper with convolutions. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, 1–9.

T, S. (2019). *Entropy: How Decision Trees Make Decisions*. Online. Abgerufen von https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8 am 24.12.2020.

Thiel, V. (2019). *Viel Lärm um Wenig - Ethische Richtlinien in der Algorithmenentwicklung*. FIfF, Forum für InformatikerInnen für Frieden und gesellschaftliche Verantwortung e.V. https://www.fiff.de/publikationen/fiff-kommunikation/fk-2020/fk-2020-1/fk-2020-1-content/fk-1-20-p21.pdf

Thorndike, R. (1953). Who belongs in the family. *Psychometrika*, *18*, 267–276.