h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

Hochschule Darmstadt

Fachbereiche Mathematik und Naturwissenschaften & Informatik

Multi-Label Branchenklassifikation von Web-Texten

Abschlussarbeit zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
im Studiengang Data Science

vorgelegt von
Dominik Mottl

Referent: Prof. Dr. Markus Döhring

Korreferentin: Prof. Dr. Uta Störl

Ausgabedatum: 03.09.2018

Abgabedatum: 18.02.2019

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 18.02.2019

Kurzfassung

Das *World Wide Web* beinhaltet eine nahezu unbegrenzte Menge an Informationen, jedoch liegt nur ein Bruchteil dieser Informationen in strukturierter Form vor. Der Großteil des Wissens ist in unstrukturierten Texten, wie bspw. Nachrichten- oder Wikipedia-Artikeln verborgen. Während ein Teil der strukturierten Informationen in Wissensdatenbanken, sogenannten *Knowledge Bases*, wie DBpedia oder in Open-Source-Projekten, wie Web Data Commons, aufbereitet und gesammelt werden, sind die meisten unstrukturierten Datenquellen noch weitgehend unerschlossen. Im Rahmen dieser Arbeit werden öffentlich zugängliche unstrukturierte Daten in Form von Nachrichtenartikeln aus dem Common-Crawl-Projekt unter Hinzunahme von Informationen aus DBpedia zum Generieren neuen Wissens genutzt. Dabei werden Methoden und Modelle aus dem *Information Retrieval* und *Natural Language Processing* dazu verwendet, die Möglichkeit und das Potential der Erweiterung von Knowlege Bases durch die enorme Menge frei zugänglicher Daten zu evaluieren. Zu diesen gehören das Extrahieren von Informationen per *Named Entity Recognition* und *Entity Linking*, die Reduzierung von Features mit dem Topic-Modell *Latent Dirichlet Allocation* und die Darstellung von Texten als numerische Vektoren, auf denen anschließend *Machine-Lerning*-Modelle zur Klassifikation trainiert werden. Der Kern der Arbeit ist die Klassifikation von Texten aus dem Common Crawl-Datensatz, die Firmen beinhalten, und daraus resultierend in einem Multi-Label Klassifikationsproblem eine oder mehrere Branchen zugeordnet bekommen. Dazu wird eine Datenaufbereitungs- und Modellbildungs-Pipeline aufgebaut, die zeigt, dass eine solche Klassifikation möglich ist. Als bestes der verwendeten Modelle schneidet ein lineares SVM-Modell ab, das auf Bag-of-Word-Features trainiert wird. Es erreicht eine *Exact Match Ratio* von 91,79%. Für die Multi-Label-, Micro- und Macro-Metriken erreicht es jeweils F1-Werte, die größer sind als 0,9. Die Pipeline ist so entworfen, dass sie per Cloud-Computing für große Datenmengen skaliert werden kann. Um das Potential der Pipeline für den Common-Crawl-Datensatz zu bewerten, wird dieser auf Texte mit enthaltenen Entitäten hin untersucht, indem ein Teil der Pipeline auf Apache Spark übertragen wird.

Abstract

The *World Wide Web* contains an almost unlimited amount of information, but only a fraction of this information is available in a structured form. Most of the knowledge is hidden in unstructured texts, such as news or Wikipedia articles. While a part of the structured information is prepared and collected in *knowledge bases*, like DBpedia or in open source projects, such as Web Data Commons, most unstructured data sources are still widely unexploited. In the context of this work, publicly accessible unstructured data in the form of news articles from the Common Crawl project are used with the addition of information from DBpedia to generate new knowledge. Methods and models from the *Information Retrieval* and *Natural Language Processing* are used to evaluate the possibility and potential of extending knowledge bases through the enormous amount of freely accessible data. These include extracting information via *Named Entity Recognition* and *Entity Linking*, the reduction of features with the topic model *Latent Dirichlet Allocation* and the representation of texts as numerical vectors on which *Machine-Lerning* models for classification are trained. The core of the work is the classification of texts from the Common Crawl dataset, which contain companies, and, as a result in a multi-label classification problem, one or more industries are assigned to. For this purpose, a data preparation and modelling pipeline is set up, which shows that such a classification is possible. The best of the models used is a linear SVM model that is trained on bag-of-word features. It achieves an *Exact Match Ratio* of 91,79%. For the multi-label, micro, and macro metrics, it reaches F1-Scores greater than 0,9. The pipeline is designed to be scaled for large amounts of data using cloud computing. To evaluate the potential of the pipeline for the Common Crawl data set, it is examined for texts with contained entities by transferring part of the pipeline to Apache Spark.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AWS	Amazon Web Services
BOW	Bag-of-Words
CC	Common Crawl
CNN	Convolutional Neural Network
CRF	Conditional Random Field
EL	Entity Linking
LDA	Latent Dirichlet Allocation
ML	Machine Learning
NER	Named Entity Recognition
RDF	Resource-Description-Framework
SVM	Support Vector Machine
URL	Uniform Resource Locator
WARC	Web ARChive

1 Einleitung

Das erste Kapitel beginnt mit der Motivation für das Thema der Arbeit. Im Anschluss wird in Abschnitt 1.2 die Zielsetzung erläutert und in Abschnitt 1.3 die Struktur der Arbeit aufgezeigt.

1.1 Motivation

In den Zeiten von „Big Data“ mit immer schneller wachsenden Datenmengen gewinnt die Extraktion von Informationen aus Daten immer mehr an Bedeutung. Die Herausforderung dabei ist, dass der Großteil der Daten unstrukturiert in Form von Texten, Audio- und Video-Formaten vorliegt. Der genaue Anteil der unstrukturierten Daten ist unbekannt, wurde jedoch lange Zeit auf 80% geschätzt [Gri08]. Mittlerweile schätzen einzelne Quellen, wie das Marktforschungsinstitut *International Data Corporation (IDC)* den Anteil jedoch auf 90% [GR11] oder sogar 95% [GH14].

Das *World Wide Web* stellt eine der wichtigsten öffentlich zugänglichen Informationsquellen der heutigen Zeit dar. Insbesondere in unstrukturierten Texten, wie in Nachrichtenartikeln oder Wikipedia-Artikeln, wird Wissen gesammelt und dargestellt. Den unstrukturierten Informationen stehen strukturierte Datenquellen wie *Knowledge Bases* gegenüber, in denen Wissen in strukturierter Form aufbereitet gesammelt wird.

Allein in den USA existieren im Jahr 2016 fast 6 Millionen Unternehmen [Uni18]. In Knowledge Bases wie DBpedia ist nur ein Bruchteil dieser Firmen bekannt. Selbst bei bekannten Firmen ist nicht immer die Branche hinterlegt. Könnten Texte einen Rückschluss auf die Branche liefern, so wäre die Hoffnung, dass entdeckte Firmen durch den Kontext der Texte, in denen sie auftreten, nach ihren Branchen klassifiziert, oder die Branchen zumindest eingegrenzt werden können. Aus den so erlangten Informationen könnte das vorhandene Wissen in Knowledge Bases erweitert werden. Dadurch wäre die Möglichkeit gegeben, einen besseren Überblick über die Wirtschaft zu erhalten und Veränderungen

bezüglich der Branchen zu erkennen. Firmen könnten dadurch potentielle Firmenkunden identifizieren, zu denen bisher keine Brancheninformationen vorliegen. Diese Firmen könnten mit einer Named Entity Recognition in Nachrichtentexten entdeckt werden und ihre Branche anschließend über die Branchenklassifikation von zugehörigen Texten, wie den Nachrichtentexten oder Texten auf der Firmenhomepage, automatisiert ermittelt werden.

Information Extraction bietet Möglichkeiten, Informationen aus unstrukturierten Texten zu extrahieren und so neues Wissen zu generieren. Zusammen mit bereits vorhandenen Daten und Modellen und Methoden aus dem *Machine Learning* ergeben sich vielseitige und neue Möglichkeiten, neues Wissen zu generieren. Mit heutigen Technologien wie Apache Hadoop und Apache Spark ist es zudem möglich, eine große Datenmenge in einer annehmbaren Zeit parallel zu verarbeiten.

1.2 Ziel der Arbeit

Ziel der Arbeit ist es, aufzuzeigen, wie Informationen aus unstrukturierten und frei zugänglichen Web-Texten mit Hilfe von Methoden aus der *Information Extraction* gewonnen und mit Daten aus Knowledge Bases kombiniert werden können, um neues Wissen zu Firmen zu generieren. Dazu werden Nachrichtenartikel von Webseiten extrahiert und mit Wissen aus einer Knowledge Base angereichert. Die so erzeugten Daten beschreiben ein Klassifikationsproblem, in dem Texte, in denen Firmen auftreten, nach Branchen klassifiziert werden.

Es wird die Annahme getroffen, dass die Anwesenheit von Firmen in einem Text Einfluss auf das Auftreten bestimmter Wörter hat und umgekehrt, dass sich Branchen aus Texten ableiten lassen. Außerdem werden die Informationen aus der Knowledge Base als Wahrheit, d.h. ihre Inhalte als richtig angesehen.

Da Firmen in mehreren Branchen gleichzeitig tätig sein und gleichzeitig mehrere Firmen in einem Text auftreten können, handelt es sich um ein Multi-Label Klassifikationsproblem. Dieses soll mit verschiedenen Modellen und Methoden aus dem *Machine Learning*, bzw. dem *Natural Language Processing* gelöst werden. Für die Modelle und Methoden wird Skalierbarkeit gefordert, sodass sie auf eine verteilte Architektur übertragen und so auf einer großen Datenmenge angewendet werden können. Bei den Modellen und Methoden werden keine Optimierungen der Parameter vorgenommen, da zunächst ein Überblick über das

Potential der vorgestellten Vorgehensweise gewonnen werden soll. Folgende Fragen sollen beantwortet werden:

- Wie lassen sich Nachrichtentexte aus dem Common-Crawl-Datensatz extrahieren?
- Wie viele potentielle Texte beinhaltet der Common-Crawl-Datensatz für das Branchenklassifikationsproblem?
- Können Firmen in den Texten identifiziert werden?
- Lassen sich Texte mit Informationen aus einer Knowledge Base anreichern?
- Welche Einflussfaktoren wirken sich auf die Ergebnisse der Klassifikation aus?
- Welche Methoden eignen sich, um ein Multi-Label Klassifikationsproblem auf den gegebenen Daten zu lösen?

Die Beantwortung der Fragen bietet einen Überblick und einen Einstieg in das Thema *Information Retrieval* (zu Deutsch Informationsgewinnung) aus unstrukturierten Texten unter Hinzunahme bekannter Informationen.

1.3 Struktur der Arbeit

In Kapitel 2 werden die Grundlagen thematisiert, die für das Verständnis der weiteren Arbeit relevant sind. Dabei werden die Themen *Machine Learning* und *Information Extraction* erläutert und die Methoden und Modelle vorgestellt, die in dieser Arbeit für den Aufbau und das Lösen des Multi-Label Klassifikationsproblems eingesetzt werden.

Das Kapitel 3 stellt die verwendeten Datenquellen und die Daten vor. Neben der Beschreibung der Daten findet die Exploration dieser statt. Zur Exploration zählt ebenfalls die Untersuchung des Common-Crawl-Datensatz auf relevante Daten für das Klassifikationsproblem.

Kapitel 4 beschreibt das Klassifikationsszenario. Dabei wird der Aufbau einer Pipeline vorgestellt, die sich von der Aufbereitung der Daten, über die Modellbildung bis hin zur Auswertung der Modelle erstreckt.

In Kapitel 5 erfolgt die Implementierung der beschriebenen Pipeline für das Klassifikations-Szenario. Zunächst wird ein Überblick über die verwendeten Technologien gegeben und erläutert, wie diese die Methoden und Modelle aus dem Machinen Learning und der Information Extraction umsetzen. Anschließend wird die Implementierung der Pipeline aufgezeigt.

Die Auswertung der Pipeline findet in Kapitel 6 statt. Dabei werden sowohl die aufbereiteten Daten, als auch die Ergebnisse der Modelle untersucht und bewertet.

Den Schluss bildet Kapitel 7, in dem eine Zusammenfassung über die gewonnen Ergebnisse und Erkenntnisse gegeben wird. In einem Ausblick werden abschließend weitere Anknüpfungspunkte an diese Arbeit aufgezeigt.

2 Grundlagen

In diesem Kapitel werden die Grundlagen geschaffen, die für das Verständnis der weiteren Kapitel benötigt werden. Dazu wird zunächst in Abschnitt 2.1 auf das Thema *Webcrawling* eingegangen, um die Herkunft eines Teils der Daten transparenter zu machen. Anschließend wird in Abschnitt 2.2 das *Machine Learning* anhand einzelner Verfahren dargestellt, die im weiteren Verlauf zum Einsatz kommen. Zuletzt wird in Abschnitt 2.3 das Thema *Infomation Extraction* und dessen Aspekte thematisiert, das bei der Verarbeitung von Texten in dieser Arbeit relevant ist.

2.1 Webcrawling

Unter *Webcrawling* [MRS09, Kapitel 20] versteht man das Sammeln von Seiten aus dem Web, um diese zu indexieren und später darin gezielt Informationen suchen zu können. Dabei sollen so schnell und effizient wie möglich eine große Anzahl von Webseiten zusammen mit der Link-Struktur, die sie untereinander verbindet, gesammelt werden. Ein Webcrawler startet mit einem oder mehreren *Uniform Resource Locators (URLs)* als Ausgangspunkt. Von den initialen URLs wählt der Crawler einen aus und ruft die zugehörige Webseite ab. Diese Webseite wird daraufhin geparst, um den Seiteninhalt und die Links auf der Seite zu extrahieren. Die extrahierten Links werden gespeichert und nacheinander abgearbeitet. Ein Webcrawler arbeitet sich also durch den Webgraphen, einen gerichteten Graphen, der durch die Verlinkung von Webseiten untereinander aufgebaut wird

2.2 Machine Learning

Im *Machine Learning (ML)* wird zum Lösen eines Klassifikationsproblems ein sogenannter Klassifikator erstellt, der später für ein gegebenes Dokument d eine Klasse c_j aus einer festen Auswahl von $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ Klassen vorhersagen soll. Klassen werden bei der Informationsgewinnung aus Texten und der Analyse von Texten (dies wird auch

als *Text Mining*) auch *Kategorien* oder *Labels* genannt [MRS09, Kapitel 13]. Ein Klassifikator wird durch automatisiertes Lernen der Eigenschaften der Kategorien anhand von Trainingsdaten erstellt bzw. trainiert [FS07, Kapitel 4]. Dabei wird zwischen zwei wesentlichen Vorgehensweisen unterschieden: dem überwachten Lernen (*supervised learning*) und dem nicht-überwachten Lernen (*unsupervised learning*). Beim überwachten Lernen findet das Trainieren der Modelle anhand von Dokumenten und deren Features statt, deren wahre Klasse bekannt ist. Im Gegensatz dazu beinhalten die Trainingsdaten im nicht-überwachten Lernen nur die Dokumente und deren Features selbst. Die Klasse der Dokumente ist während des Trainings unbekannt. Dokumente sollen hier anhand ihrer gegebenen Informationen in sinnvolle Gruppen, sogenannte *Cluster*, eingeteilt werden [FS07, Kapitel 5]. Eine Kombination der beiden Ansätze stellt das halb-überwachte Lernen (*semi-supervised learning*) dar. Bei den Ansätzen aus dieser Kategorie stehen eine kleine Menge von gelabelten und eine große Menge von nicht-gelabelten Trainingsdaten zur Verfügung, die gemeinsam für das Trainieren der Modelle verwendet werden.

In den folgenden Abschnitten werden die Modelle aus dem ML erklärt, die in dieser Arbeit verwendet werden. Dabei handelt es sich um vier Modelle, die dem überwachten Lernen zuzuordnen sind und um ein Modell aus dem nicht-überwachten Lernen. Die Modelle aus dem überwachten Lernen werden in dieser Arbeit für die Klassifikation der Texte eingesetzt (genauere Gründe zur Wahl der einzelnen Modelle siehe Kapitel 4.4.5), während das Modell aus dem nicht-überwachten Lernen beim Generieren der Features zur Reduktion der Dimensionalität eingesetzt wird (siehe Kapitel 4.4.4 S. 61).

In den folgenden Abschnitten werden die *supervised* ML-Modelle *Naive Bayes*, *logistische Regression*, *Entscheidungsbäume* und *Convolutional Neural Networks* vorgestellt. Aus dem Bereich des *unsupervised learnings* wird das Modell *Latent Dirichlet Allocation* erläutert.

2.2.1 Naive Bayes

Ein einfaches und daher sehr beliebtes Modell für die Klassifikation von Texten ist der *multinomiale Naive-Bayes-Klassifikator*, wie er von Manning et al. beschrieben wird [MRS09, Kapitel 13]. Dabei handelt es sich um ein probabilistisches Modell aus dem *supervised learning*. Die Wahrscheinlichkeit, dass ein Dokument d mit n_d Termen zur Klasse c gehört,

lässt sich folgendermaßen berechnen:

$$P(c|d) \propto P(c) \prod_{i \leq k \leq n_d} P(t_k|c), \tag{2.2.1}$$

wobei $P(t_k|c)$ die bedingte Wahrscheinlichkeit ist, dass der Term t_k in einem Dokument der Klasse c auftritt und $P(c)$ die A-priori-Wahrscheinlichkeit für die Klasse c darstellt.

Der Naive-Bayes-Klassifikator basiert auf dem Satz von Bayes und ordnet ein Dokument der wahrscheinlichsten Klasse c_{map} (*maximum a posteriori*) zu, indem er folgendes Optimierungsproblem löst:

$$c_{map} = \arg\max_{c \in \mathbb{C}} P(c|d) \tag{2.2.2}$$

$$= \arg\max_{c \in \mathbb{C}} \frac{P(d|c)P(c)}{P(d)} \tag{2.2.3}$$

$$= \arg\max_{c \in \mathbb{C}} P(d|c)P(c) \tag{2.2.4}$$

Dabei werden zwei Annahmen getroffen:
1. Bedingte Unabhängigkeit: Die Terme in einer Klasse sind unabhängig voneinander

$$P(c|d) = P(\langle t_1, \ldots, t_{n_d} \rangle | c) = \prod_{1 \leq k \leq n_d} P(X_k = t_k|c) \tag{2.2.5}$$

2. Positionelle Unabhängigkeit: Die bedingte Wahrscheinlichkeit für einen Term ist unabhängig von seiner Position im Dokument

$$P(X_{k_1} = t|c) = P(X_{k_2} = t|c), \tag{2.2.6}$$

für alle Positionen k_1, k_2, Terme t und Klassen c.

Aufgrund dieser beiden „naiven“ Annahmen für natürliche Sprache bezüglich der Unabhängigkeiten wird das Modell als *Naive Bayes* bezeichnet.

Die Schätzer $\hat{P}(c)$ für die A-priori-Wahrscheinlichkeit und $\hat{P}(t_k|c)$ für die bedingte Wahrscheinlichkeit werden per Maximum-Likelihood-Schätzung unter Zuhilfenahme von rela-

tiven Häufigkeiten bestimmt. Für die A-priori-Wahrscheinlichkeit gilt für den Schätzer:

$$\hat{P}(c) = \frac{N_c}{N}, \tag{2.2.7}$$

wobei N_c die Anzahl der Dokumente in Klasse c ist und N die gesamte Anzahl an Dokumenten. Der Schätzer für die bedingte Wahrscheinlichkeit wird folgendermaßen berechnet:

$$\hat{P}(t_k|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}, \tag{2.2.8}$$

wobei T_{ct} angibt, wie häufig t unter c auftritt.

2.2.2 Logistische Regression

Die *logistische Regression* [HTF08] [JWHT13] ist die Anwendung der Regressionsanalyse auf ein Klassifikationsproblem. Im binären Fall wird zwischen zwei Klassen entschieden: der positiven Klasse 1 und der negativen Klasse 0. Dabei soll für eine Beobachtung $\mathbf{x}_i$ die Wahrscheinlichkeit $p(X) = P(Y_i = 1|X = \mathbf{x}_i)$ bestimmt werden, so dass die abhängige Variable $Y \in \{0, 1\}$ den Wert 1 annimmt, also der positiven Klasse angehört. Das logistische Regressionsmodell folgt der Form:

$$p(X) = P(Y_i = 1|X = \mathbf{x}_i) = \frac{e^{\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}}}{1 + e^{\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}}} \ , \tag{2.2.9}$$

wobei $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})$ eine Beobachtung mit p unabhängigen Variablen sei. Über die Chancen-Verhältnisse

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}} \ , \tag{2.2.10}$$

auch *Odds* genannt, erhält man nach deren Transformation durch Logarithmieren die sogenannten Logits, als lineare Regressionsgleichung:

$$\log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta} \tag{2.2.11}$$

Bei der logistischen Regression müssen die Koeffizienten β_0 und $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)$ geschätzt werden. Die Schätzung findet über die *Maximum Likelihood Methode* statt, die für die

logistische Regression folgendermaßen aussieht:

$$l(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^{N} \left\{ y_i(\beta 0 + \boldsymbol{\beta}^T \mathbf{x}_i) - \log(1 + e^{\beta_0 + \mathbf{x}_i^T}) \right\} \tag{2.2.12}$$

Die Schätzer $\hat{\beta}_0$ und $\hat{\boldsymbol{\beta}}$ werden so gewählt, dass sie die Likelihood-Funktion maximieren.

2.2.3 Support Vector Machines

Eine *Support Vector Machine (SVM)* ist ein Vektorraum-basiertes Klassifikationsverfahren aus dem Bereich *supervised learning*, das wie von Manning et al. [MRS09, Kapitel 15] beschrieben, insbesondere für die Klassifikation von Texten eingesetzt wird. Ziel ist es, eine Entscheidungsgrenze zwischen zwei Klassen zu finden, die einen maximalen Abstand zu den Punkten in den Trainingsdaten aufweist. Im Folgenden werden binäre Klassifikationsprobleme betrachtet, die durch einen linearen Klassifikator getrennt werden sollen (siehe Abbildung 2.1). Der Abstand von der Entscheidungsgrenze zu den nächsten Datenpunkten wird *Margin* genannt. Bei der Entscheidungsgrenze handelt es sich um eine Hyperebene, die durch ein Intercept b und einen Normalenvektor $\boldsymbol{\omega}$ definiert ist. In der Literatur wird der Normalenvektor auch oft *weight vector* genannt. Da der Normalenvektor senkrecht auf der Hyperebene steht, erfüllen alle Punkte $\mathbf{x}$ auf der Hyperebene $\boldsymbol{\omega}^T \mathbf{x} = -b$. Die Wahl der Entscheidungsgrenze, die durch die Maximierung des Margins bestimmt wird, ist abhängig von einer kleinen Teilmenge der Daten, den sogenannten *Support Vectors*. Alle anderen Datenpunkte spielen bei der Bestimmung der Hyperebene keine Rolle.

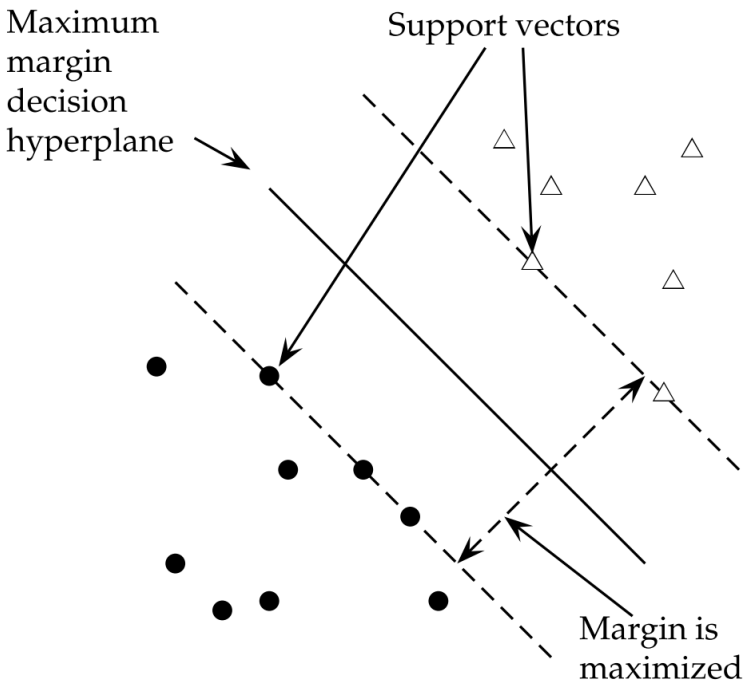


Abbildung 2.1: Linearer Ansatz mit SVMs bei einem binären Klassifikationsproblem [MRS09, Kapitel 15].

Für eine Trainingsmenge $\mathbb{D} = \{(\mathbf{x}_i, y_i)\}$, bei der jeder Eintrag ein Tupel von Datenpunkt $\mathbf{x}_i$ und Klasse $\mathbf{y}_i$ darstellt, werden den Klassen bei den SVMs die Werte $+1$ und -1 zugewiesen. Der lineare Klassifikator hat folgende Form:

$$f(\mathbf{x}) = \text{sign}(\boldsymbol{\omega}^T\mathbf{x} + b) \tag{2.2.13}$$

Der Margin $\rho = \frac{2}{|\boldsymbol{\omega}|}$ soll maximiert werden, was das Gleiche ist, wie $\frac{|\boldsymbol{\omega}|}{2}$ zu minimieren. Das Optimierungsproblem, das sich daraus ergibt, besteht darin, $\boldsymbol{\omega}$ und b so zu wählen, dass:

- $\frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega}$ minimiert wird und
- für alle $\{(\mathbf{x}_i, y_i)\}$ $y_i(\boldsymbol{\omega}^T\mathbf{x} + b) \geq 1$ gilt.

Für hochdimensionale Klassifikationsprobleme, wie es im Textmining der Fall ist, sind die Daten in der Regel nicht linear trennbar. In diesem Fall soll der Großteil der Daten linear getrennt werden, indem das Rauschen durch einzelne falsch klassifizierte Dokumente ignoriert wird. Dieses Vorgehen ist bei den SVMs durch die Einführung einer sogenannten *Schlupfvariable* ξ_i umgesetzt (siehe Abbildung 2.2).

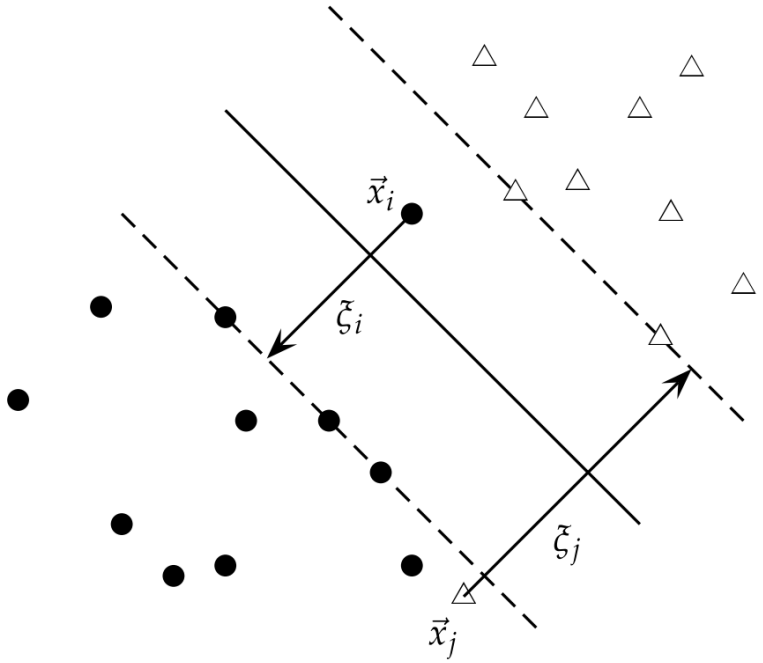


Abbildung 2.2: Lineares Klassifikationsproblem durch SVM mit Schlupfvariable (Quelle: [MRS09, Kapitel 15]).

Mit $\xi_i \geq 0$ werden Fehlklassifikationen von Datenpunkten $\mathbf{x}_i$ erlaubt, wobei ξ_i den Abstand zur korrekten Klasse darstellt. Die Summe der Fehlklassifikationen soll so klein wie möglich gehalten werden. Daher werden diese mit einer Konstante C stärker bestraft. Das Optimierungsproblem mit Schlupfvariable besteht darin, $\boldsymbol{\omega}$, b und $\xi_i \geq 0$ so zu wählen, dass:

- $\frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + C\sum_i \xi_i$ minimiert wird und
- für alle $\{(\boldsymbol{x}_i, y_i)\}$, $y_i(\boldsymbol{\omega}^T\mathbf{x} + b) \geq 1 - \xi_i$ gilt.

2.2.4 Entscheidungsbäume

Ein weiteres *supervised learning*-Verfahren zum Lösen von Klassifikationsproblemen stellen *Entscheidungsbäume* [HTF08], auch *Decision Trees* genannt, dar, die unter Verwendung der CART-Methode (*Classification and Regression Trees*) als sogenannte *Classification Trees* eingesetzt werden können. Entscheidungsbäume partitionieren den Feature-Raum anhand der Features rekursiv in binäre Regionen. In einem zweidimensionalen Feature-Raum werden diese Regionen durch Rechtecke dargestellt, wie sie in Abbildung 2.3 zu sehen sind.

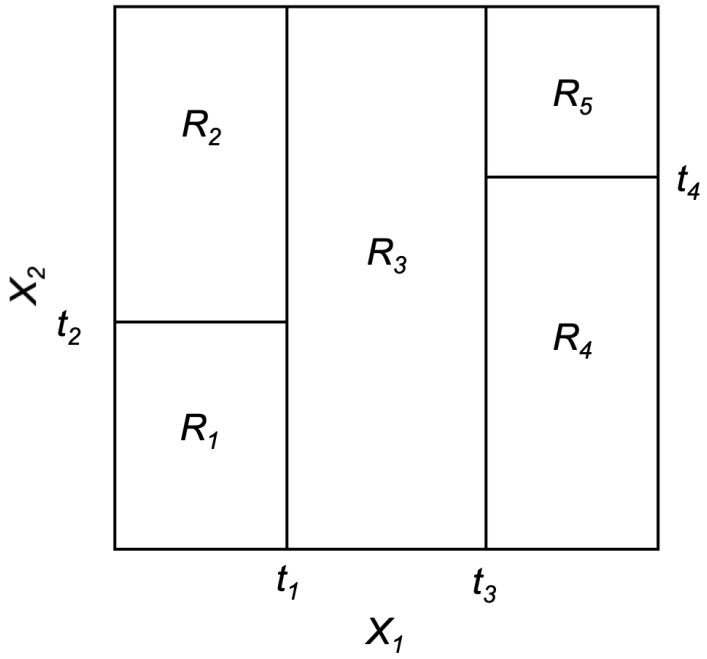


Abbildung 2.3: Aufteilung in Regionen im Entscheidungsbaum-Modell; in Anlehnung an [HTF08].

Die Aufteilung des Raums erfolgt anhand der Features und kann als Binär-Baum, wie in Abbildung 2.4 zu sehen ist, dargestellt werden. Dies hat den Vorteil, dass die Modelle dadurch leicht zu interpretieren sind. Der Wurzel-Knoten im Binär-Baum repräsentiert den gesamten Raum. Hier befinden sich alle Beobachtungen.

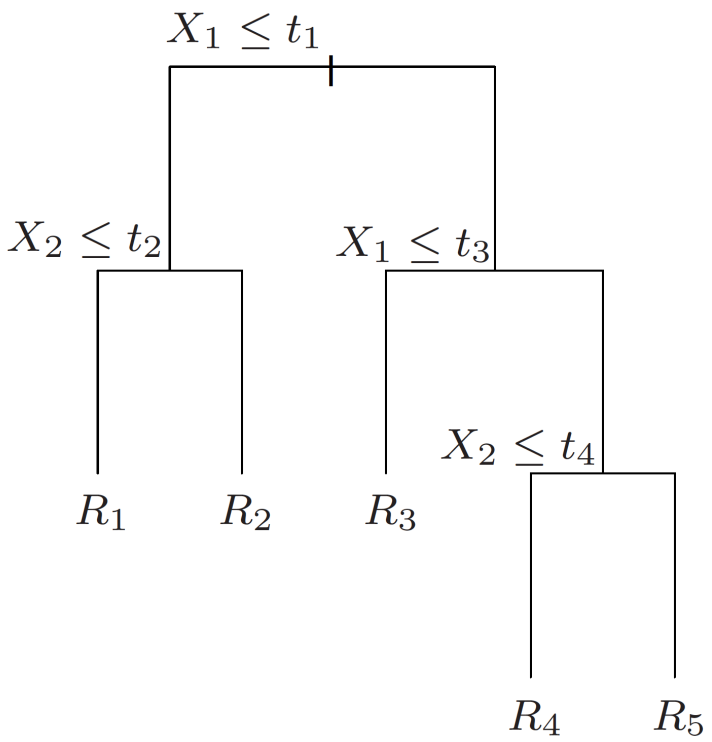


Abbildung 2.4: Aufbau Entscheidungsbaum als Binär-Baum [HTF08].

Im ersten Schritt wird der Raum anhand einer Split-Variable j und einem Split-Point s in die zwei Regionen R_1 und R_2 aufgesplittet. Für den Binär-Baum bedeutet dies, dass zwei Blatt-Knoten entstehen, die den beiden Regionen entsprechen. Die Beobachtungen verteilen sich auf die Blatt-Knoten bzw. Regionen der Form:

$$R_1(j,s) = \{X|X_j \leq s\} \text{ und } R_2(j,s) = \{X|X_j > s\} \tag{2.2.14}$$

Die beiden Regionen können jeweils erneut aufgeteilt und die Beobachtungen dadurch weiter auf die neu entstandenen Blatt-Knoten verteilt werden. Ziel der Entscheidungsbäume beim Splitten ist j und s so zu wählen, dass möglichst „reine“ Knoten entstehen. Dies bedeutet, dass die Beobachtungen, die sich in den Knoten bzw. in den zugehörigen Regionen befinden, möglichst der selben Klasse k angehören. Der Anteil einer Klasse k im Knoten m, der N_m Beobachtungen enthält und Region R_m repräsentiert, ist folgendermaßen definiert:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{2.2.15}$$

Anhand der Verteilungen der Klassen innerhalb der Knoten und Regionen, die durch die Splits entstehen, werden für jeden Split die beste Split-Variable j und deren bester Split-Point s ermittelt. Die Reinheit spielt in den Knoten eine entscheidende Rolle, denn bei der Klassifikation durch Entscheidungsbäume werden einer Beobachtung im Knoten m die Klasse k zugeordnet, die die Mehrheit im Blatt-Knoten besitzt: $k(m) = \text{argmax}_k \hat{p}_{mk}$. Dies bedeutet, je reiner ein Knoten ist, desto genauer bzw. eindeutiger fällt die Klassifikation aus. Aus diesem Grund werden zur Bestimmung der Splits sogenannte *Unreinheitsmaße* (*impurity measures*) verwendet. Zu diesen gehören:

- Fehlklassifikationsrate: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}$
- Gini-Index: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- Kreuzentropie: $-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

Die Split-Variable und der Split-Point, der unter Betrachtung des gewählten Unreinheitsmaßes zur geringsten Unreinheit führt, werden für den Split verwendet. Entscheidungsbäume können so weit aufgebaut werden, dass sich in den Blatt-Knoten nur noch eine Beobachtung befindet. Dies würde allerdings zu einem *Overfitting* führen, also einer Überanpassung des Modells an die Trainingsdaten und einer schlechten Performance des Klassifikators auf beliebigen Daten. Um diesem Problem entgegenzuwirken, kann eine minimale Knotengröße angegeben werden, bei deren Erreichen das weitere Splitten gestoppt wird. Eine weitere Maßnahme ist das *Zurückschneiden* (*Pruning*) der Entscheidungsbäume auf höhere Knoten.

2.2.5 Topic Modelle

Bei der *LDA* handelt es sich um ein sogenanntes Topic Modell, in dessen Zusammenhang zum Zeitpunkt dieser Arbeit viele Forschungsaktivitäten vorhanden sind. Die LDA ist ein generatives Wahrscheinlichkeitsmodell, das dem *unsupervised learning* zuzuordnen ist und von Blei et al. [BNJ03] vorgestellt wurde. Geeignet ist das Modell für diskrete Daten, insbesondere für Textkorpora, um *Inhalte* (hier: *Themen*) in Dokumenten und Beziehungen zwischen Dokumenten effizient zu ermitteln. Sie sind für die Untersuchung großer Datenmengen geeignet. LDAs lassen sich nicht nur auf Texte anwenden, sondern auch auf Anwendungsgebiete wie der Bioinformatik oder der inhaltsbasierten Bildverarbeitung. Im Kontext dieser Arbeit werden LDAs zur Untersuchung von Textkorpora mit M Dokumenten betrachtet.

Die Grundidee des LDA-Modells ist es, dass Dokumente als zufällige Mischung von *verborgenen Themen* (engl. *latent topics*) dargestellt werden, wobei jedes Thema durch eine Verteilung von Termen charakterisiert wird. LDA geht von folgendem dreistufigen, hierarchischen, generativen Prozess (siehe Abbildung 2.5) für jedes Dokument **d** in einem Korpus aus:

1. Wähle die Anzahl der Wörter N für das Dokument **d** nach der Poisson-Verteilung: $N \sim \textbf{Poisson}(\xi)$
2. Wähle die Variable θ für die Themenverteilung nach der Dirichlet-Verteilung: $\theta \sim \textbf{Dir}(\alpha)$
3. Für jeden Term t_n der N Worte:
   (a) Wähle ein Thema z_n nach der Multinomial-Verteilung: $\mathrm{z}_n \sim \textbf{Multinomial}(\theta)$
   (b) Wähle ein Wort t_n abhängig vom Thema z_n nach der Multinomial-Verteilung: $t_n \sim \textbf{Multinomial}(\beta_{\mathrm{z}_n})$

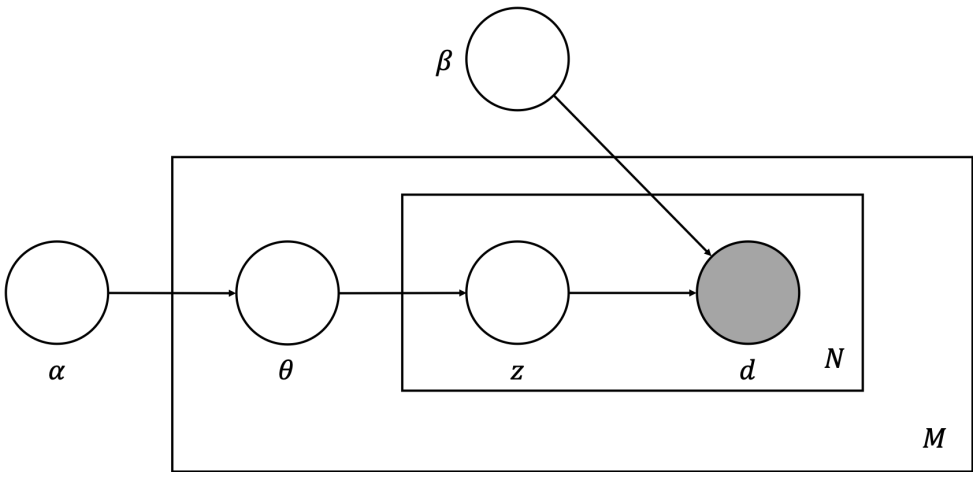


Abbildung 2.5: Grafische Darstellung LDA Modell (angelehnt an [BNJ03]).

Im Modell werden mehrere Annahmen getroffen:

- Die Dimensionalität k der Dirichlet-Verteilung wird als bekannt angenommen und wird festgesetzt.

- Die Wahrscheinlichkeiten der Terme werden parametrisiert per $k \times V$ Matrix β, wobei V das Vokabular sei und $\beta_{ij} = p(w^j = 1 \mid z^i = 1)$ als feste Menge behandelt wird, die geschätzt werden muss.
- Die Annahme der Poisson-Verteilung wird als nicht kritisch für das Modell angesehen und kann bei Bedarf durch eine andere Verteilung für die Dokumentenlänge ersetzt werden.

Im Modell sind nur die Terme t_n des Dokumentes $\mathbf{d}$ als Variablen bekannt. Alle anderen Variablen, wie die Themenverteilung θ, die Themen $\mathbf{z}$ selbst, die Parametrisierung α für die Dirichlet-Verteilung der Themen und die Parametrisierung β für die Multinomial-Verteilung der Terme sind unbekannt und müssen geschätzt werden. Die A-posteriori-Wahrscheinlichkeit der latenten Variablen bei einem gegebenen Dokument ist folgendermaßen definiert:

$$p(\theta, \mathbf{z} \mid \mathbf{d}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{d} \mid \alpha, \beta)}{p(\mathbf{d} \mid \alpha, \beta)}, \text{ wobei} \tag{2.2.16}$$

$$p(\mathbf{d} \mid \alpha, \beta) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int (\prod_{i=1}^{k} \theta_i^{\alpha_i - 1})(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j}) d\theta \text{ sei.} \tag{2.2.17}$$

Sie ist durch die Kopplung von θ und β nicht exakt unlösbar und muss daher numerisch approximiert werden. Diese Unlösbarkeit wird als Inferenz-Problem bezeichnet. Zum Approximieren der A-posteriori-Wahrscheinlichkeit wir die *variationale Verteilung* (*variational inference*) verwendet:

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = q(\theta \mid \gamma \prod_{n=1}^{N} q(z_n \mid \phi_n)) \ , \tag{2.2.18}$$

wobei der Dirichlet Parameter γ und die Multinomial Parameter $(\phi_1, ..., \phi_N)$ die freien variationalen Parameter darstellen. Außerdem wird das Modell, wie in Abbildung 2.6 zu sehen ist, erweitert, indem die β_i als Zufallsvariablen angesehen werden. Diese werden unabhängig von einander aus einer austauschbaren Dirichlet-Verteilung mit dem skalaren Parameter η gezogen. Dadurch wird eine Glättung erreicht, die Wörtern, die nicht in der Trainingsmenge vorkommen und somit auch den neuen Dokumenten bei der Parameterschätzung keine Null-Wahrscheinlichkeiten zuweist.

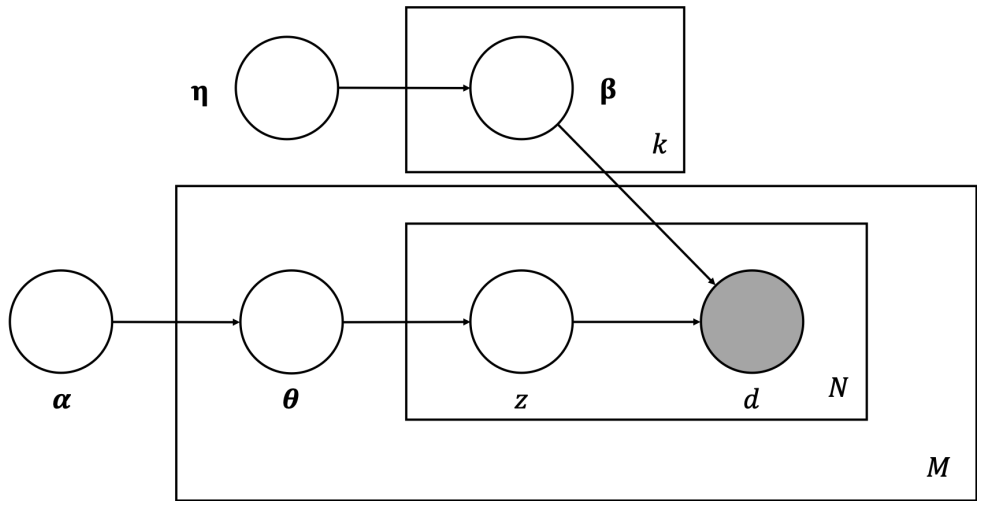


Abbildung 2.6: Grafische Darstellung des geglätteten LDA-Modells (angelehnt an [BNJ03]).

Zum Approximieren der Parameter im LDA-Modell stellen Blei et al. [BNJ03] eine variationale Bayes'sche Inferenz Methode vor, die per EM-Algorithmus (Estimation-Maximization-Algorithmus) die Parameter des geglätteten Modells schätzt. Dazu wird die eingeführte variationale Verteilung verwendet.

2.2.6 Convolutional Neural Networks

Bei einem *Convolutional Neural Network (CNN)* handelt es sich um ein Modell, das den Künstlichen Neuronalen Netzen (KNN)[Zur92] zuzuordnen ist. Ursprünglich kommen die CNNs aus der Bildverarbeitung, wie von Matsugu et al. [MMMK03] beschrieben, um einen effizienten Ansatz für die Klassifikation von Bildern zu bieten. CNNs lassen sich jedoch auch auf die Klassifikation von Texten anwenden, wie Kim [Kim14] gezeigt hat. In dieser Arbeit wird Kims Ansatz verwendet, der in die Kategorie *supervised learning* fällt.

Das CNN besteht im einfachsten Fall aus vier Schichten, sogenannten *Layern* (siehe Abbildung 2.7). Die erste Schicht ist der *Input-Layer*. Hier sind die Terme des zu klassifizieirenden Textes zur weiteren Verarbeitung als k-dimensionale Vektoren dargestellt. Kim verwendet zur Repräsentation der Terme vortrainierte Word2Vec-Vektoren (siehe Mikolov et al. [MSC+13]), die wiederum selbst von Neuronalen Netzen trainiert wurden. Als zweite Schicht wird ein *Convolutional-Layer* eingesetzt, der per Faltungen (engl.: *convolutions*) aus den Input-Vektoren neue Features generiert. Eine Faltung wird von einem

Filter $\mathbf{w} \in \mathbb{R}^{hk}$ durchgeführt, wobei h die Fenstergröße des Filters angibt. Ein neues Feature c_i wird von einem Fenster der Worte $\mathbf{x}_{i:i+h-1}$ folgendermaßen generiert:

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b) \tag{2.2.19}$$

Dabei ist $b \in \mathbb{R}$ ein Bias und f eine nicht-lineare Aktivierungsfunktion. Der Filter wird als *Sliding Window* auf den kompletten Text angewendet, um eine *Feature Map*

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{2.2.20}$$

mit $b \in \mathbb{R}^{n-h+1}$ zu generieren. Im *Max-Pooling-Layer* wird anschließend der maximale Wert $\hat{c} = \max\{\mathbf{c}\}$ als Feature selektiert, das dem zugehörigen Filter entspricht. Da ein CNN mehrere Filter verwendet, werden mehrere Features generiert (1-zu-1-Beziehung zwischen Filter und Feature). Die Features werden der letzten Schicht, dem *Fully-Connected-Layer*, übergeben. Diese Schicht ist *vollständig verbunden* (engl.: *fully connected*), da jedes Neuron eine Verbindung zu jedem einzelnen der Features aus der vorherigen Schicht aufweist. Der Output dieser Schicht ist die Wahrscheinlichkeitsverteilung über den Labeln bzw. Klassen des Klassifikationsproblems, die per *Softmax-Funktion* abgebildet wird.

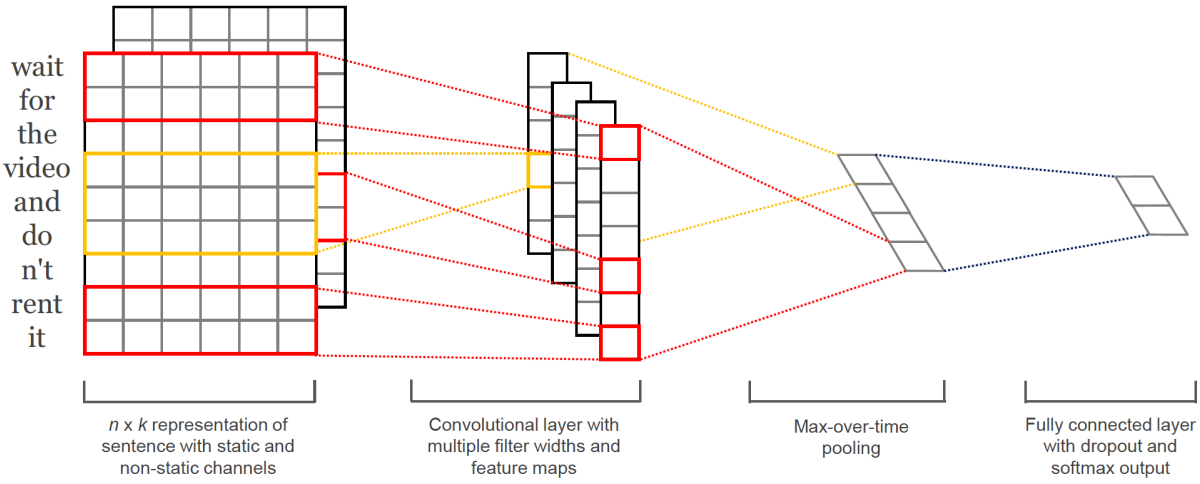


Abbildung 2.7: Architektur CNN zur Klassifizierung von Texten unter Verwendung von zwei verschiedenen Kanälen [Kim14].

Zur Veranschaulichung der Funktionsweise ist in Abbildung 2.8 ein CNN dargestellt, das Sätze nach positiven oder negativen Stimmungen klassifizieren soll. Als Input wird der Satz „i like star wars“ gewählt, dessen einzelne Worte in Vektoren der Dimensionalität drei dargestellt werden.

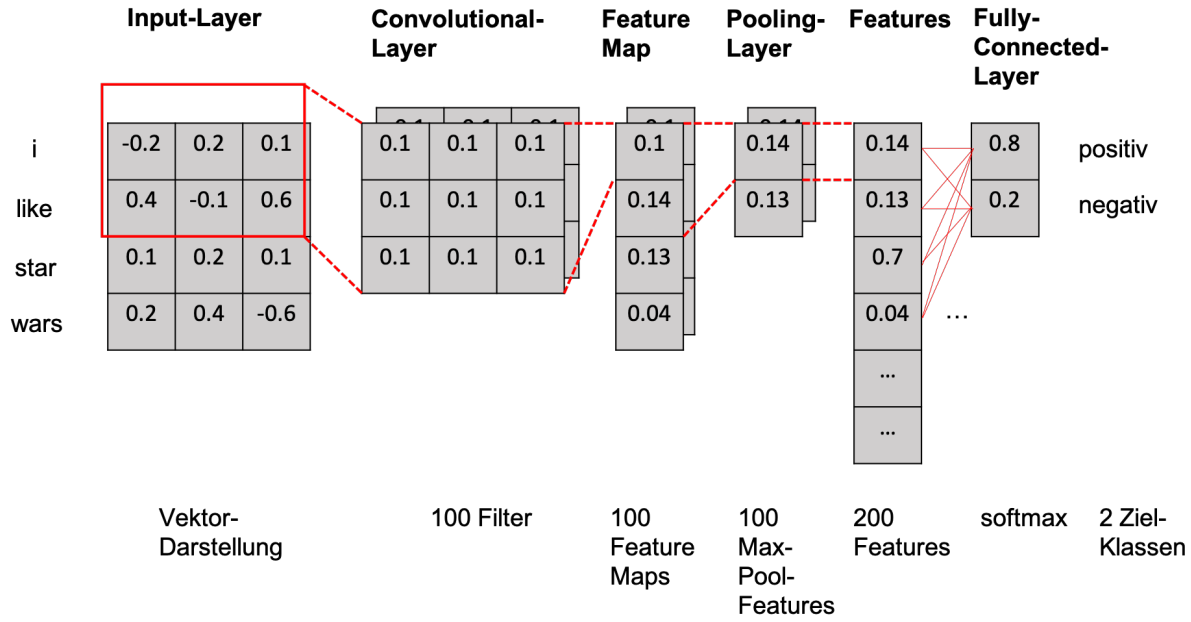


Abbildung 2.8: Beispiel für die Klassifikation eines Satzes auf positive oder negative Stimmung mit einem CNN mit 100 Filtern der Fenstergröße 3.

Die Vektordarstellung im Input-Layer würde in einem realen Anwendungsfall aus trainierten Word-Embeddings stammen, die per Lookup den Worten zugewiesen werden. Das Netz besitzt im Convolutional-Layer 100 Filter mit der Fensterbreite drei. Jeder Filter wird schrittweise über die Input-Matrix geschoben, um eine Feature Map zu erzeugen. Dabei wird bei jedem Schritt die Summe der elementenweisen Produkte gebildet. Damit die Feature Map der einzelnen Filter die gleiche Länge wie der Input-Satz aufweist, werden die Filter um eine Zeile über beide Seiten der Input-Matrix hinaus geschoben. Es entstehen dadurch 100 Feature Maps der Länge vier. Im Pooling-Layer wird die Länge der Feature-Vektoren auf zwei reduziert, indem ein Fenster der Breite zwei und der Schrittweite zwei über die einzelnen Feature Maps geschoben wird. In jedem Schritt wird aus den zwei betrachteten Einträgen der größte Wert ausgewählt. Im nächsten Schritt werden alle so entstandenen 100 Max-Pool-Feature-Vektoren der Länge zwei zu einem einzelnen Feature-Vektor der Länge 200 zusammengefügt. Jeder Eintrag des Feature-Vektors wird

mit jedem Neuron des Fully-Connected-Layer verbunden. Die Anzahl der Neuronen in diesem Layer entspricht der Anzahl der betrachteten Klassen. Im Beispiel sind es also zwei Neuronen. Unter Anwendung der Aktivierungsfunktion *Softmax* entsteht der Output, die Wahrscheinlichkeitsverteilung der Klassen. In diesem Fall gehört der Beispielsatz mit einer 80%igen Wahrscheinlichkeit der positiven Klasse an.

2.3 Information Extraction

Bei der *Information Extraction* [AZ12, Kapitel 2] geht es darum, strukturierte Informationen aus un- oder semi-strukturierten Texten zu extrahieren. Dabei spielt Information Extraction insbesondere im Text Mining eine wichtige Rolle, da hier in vielen Fällen Entitäten, wie beispielsweise Personen, Orte oder Organisationen, in Texten erkannt (siehe Abschnitt 2.3.1), zugeordnet (siehe Abschnitt 2.3.2) und in Verbindung zueinander gesetzt werden müssen. Information Extraction wird häufig bei der Vorverarbeitung der Daten eingesetzt, damit die daraus erhaltenen Informationen als Input für ML-Verfahren genutzt werden können.

2.3.1 Named Entity Recognition

Unter dem Begriff *Named Entity Recognition (NER)* [AZ12, Kapitel 2.2] versteht man das Identifizieren von Wörtern oder Wort-Sequenzen, die Entitäten wie Personen, Organisationen oder Orte darstellen. Ein Beispielsatz für die NER könnte so aussehen:

Apple Inc. Mitgründer Steve Jobs ist in Palo Alto gestorben.

Per NER könnten in dem Satz folgende Entitäten erkannt werden:

<ORGANISATION> Apple Inc. <|ORGANISATION>
Mitgründer <PERSON> Steve Jobs <|PERSON> ist in
<LOCATION> Palo Alto <|LOCATION> gestorben.

Da NER für viele weitere Aufgaben im Bereich Information Extraction vorausgesetzt wird und die Qualität der weiteren Verarbeitungsschritte von der Qualität der NER abhängt, gilt sie als der fundamentalste Task der Information Extraction. Die Lösungsansätze für das NER-Problem lassen sich in folgende beide Vorgehensweisen unterteilen: *regelbasierte Ansätze* und *statistische Modelle*.

Bei den regelbasierten Ansätzen werden manuell Regeln definiert, üblicherweise durch *reguläre Ausdrücke*, gegen die der gewünschte Text abgeglichen wird [AZ12, Kapitel 2.2.1]. Wird eine solche Regel erfüllt, indem beispielsweise ein bestimmter Term entdeckt wird, so folgt die Ausführung einer definierten Aktion. In der Praxis könnte eine Regel so aussehen, dass nach dem Term *Mr.* gesucht wird. Wird der Term gefunden, wird die Aktion ausgeführt, die den Term zusammen mit dem darauf folgenden Term als Entität *PERSON* markiert. Die manuelle Erstellung der Regeln per menschlicher Expertise ist sehr aufwändig. Automatisierte Ansätze, die auf gelabelten Trainingsdaten arbeiten, können diesen Aufwand reduzieren.

Aktuelle NER-Ansätze basieren auf statistischen Modellen, wie Hidden Markov Models, Maximum Entropy Models, Maximum Entropy Markov Models, SVMs oder Conditional Random Fields [AZ12, Kapitel 2.2]. Eine weitere Vorgehensweise ist die Verwendung von Neuronalen Netzwerken (siehe [LBS$^+$16]). Eine Gemeinsamkeit der genannten Methoden ist, dass der NER-Task als Sequenz-Labeling-Problem behandelt wird [AZ12, Kapitel 2.2.2]. Gegeben sei eine Sequenz von Beobachtungen $\mathbf{x} = (x_1, x_2, ..., x_n)$, wobei jede Beobachtung typischerweise als Feature-Vektor dargestellt wird. Jeder Beobachtung x_i soll ein Label y_i zugewiesen werden. Dabei hängt im Gegensatz zu Standard-Klassifikatoren y_i nicht nur von der zugehörigen Beobachtung x_i ab, sondern auch von anderen Beobachtungen und Labeln in der Sequenz, die sich typischerweise in der näheren Nachbarschaft der Position i befinden. Im Falle der NER bedeutet das, dass ein Satz einer Sequenz und jedes Wort im Satz einer Beobachtung entspricht.

Ein aktuelles und gleichzeitig sehr effektives Modell für das Sequenzen-Labeling bei der NER stellen die *Conditional Random Fields (CRFs)* dar, die von Lafferty et al. [LMP01] eingeführt wurden. Dabei handelt es sich im Allgemeinen um ein ungerichtetes Graphen-Modell, wobei das Label der aktuellen Beobachtung nicht nur vom Label der vorherigen Beobachtung abhängig ist, sondern ebenfalls vom Label der nächsten Beobachtung und von der Beobachtung selbst (siehe Abbildung 2.9 a). Eine spezielle Art der Umsetzung für die NER stellen die *linear-chain* CRFs dar. Dabei bilden die Sequenz der Beobachtungen $\mathbf{x} = (x_1, x_2, ..., x_n))$ und die zugehörige Sequenz verdeckter Zufallsvariablen $\mathbf{y} = (y_1, y_2, ..., y_n))$ eine lineare Kette, indem die y_i's über eine Kante miteinander verbunden sind (siehe Abbildung 2.9 b). Die Verteilung der linearen Kette sieht folgendermaßen

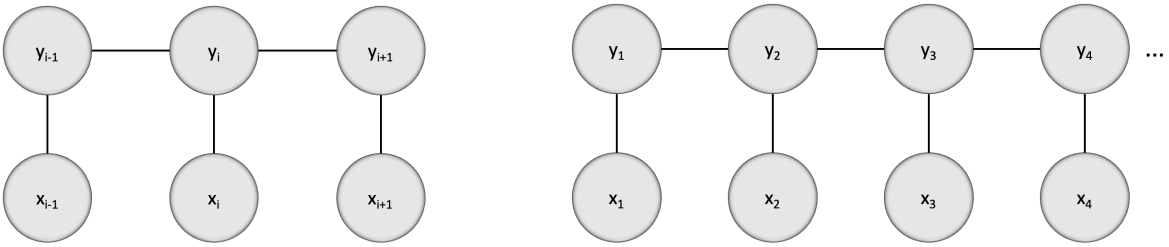
aus:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\exp(\sum_i \sum_j \lambda_j f_j(y_i, y_{i-1}, \mathbf{x}, i)), \tag{2.3.1}$$

wobei $Z(\mathbf{x})$ ein normalisierter Faktor über alle möglichen Label-Sequenzen darstellt:

$$Z(X) = \sum_{\mathbf{y}'} \exp(\sum_i \sum_j \lambda_j f_j(y'_i, y'_{i-1}, \mathbf{x}, i). \tag{2.3.2}$$

Die Funktion $f_j()$ ist dabei die Feature-Funktion und λ_j die zugehörige Gewichtung. Zum Trainieren der CRFs wird eine Maximum-Likelihood-Schätzung eingesetzt, die $p(\mathbf{y}|\mathbf{x})$ maximiert [AZ12, Kapitel 2.2.2] [AZ12, Kapitel 4.4.4].



(a) (b)

Abbildung 2.9: (a) Ungerichtetes Graphen-Modell der CRFs und (b) Lineare Verkettung CRFs.

2.3.2 Entity Linking

Unter *Entity Linking (EL)* [HYB+11] [SWH15] [MJGsB11] versteht man das Verknüpfen von in Texten erwähnten Entitäten mit den zugehörigen Entitäten in einer Knowledge Base, wie in dieser Arbeit mit der Knowledge Base *DBpedia* (siehe Kapitel 3.2). Diese Aufgabe wird dadurch erschwert, dass die Namen der erwähnten Entitäten oftmals mehrdeutig sind (engl.: ambiguous), der selbe Name also mehrere Bedeutungen haben kann, bzw. zu mehreren Entitäten in einer Knowledge Base gehören kann. So kann der Name „Washington“ für einen Bundesstaat, eine Stadt oder den ersten Präsidenten der Vereinigten Staaten stehen.

Der Prozess des EL lässt sich in drei wesentliche Teilaufgaben unterteilen, die jeweils aufeinander aufbauen [MJGsB11]:

1. **Spotting:** Im ersten Schritt wird ein Text auf die Erwähnung von Entitäten untersucht. Dazu werden Methoden aus der NER (siehe Abschnitt 2.3.1) eingesetzt.

2. **Candidate Selection:** Den entdeckten Entitäten aus dem Text werden mögliche Kandidaten aus der Knowledge Base zugeordnet.

3. **Disambiguation:** Im letzten Schritt wird der Kontext um die erwähnten Entitäten untersucht, um den wahrscheinlichsten Kandidaten für das EL zu finden. Die Aufgabe kann als Ranking-Problem angesehen werden, dessen Ziel es ist, die korrekte Entität aus der Knowledge Base an Position 1 zu setzen, abhängig von der Ähnlichkeit des Kontextes der erwähnten Entität mit den Kontexten der Kandidaten.

Das genaue Vorgehen und die verwendeten Methoden sind vom eingesetzten Tool abhängig. In Kapitel 5.1.2 wird beschrieben, wie das EL in dieser Arbeit eingesetzt und umgesetzt wird.

3 Daten

In diesem Kapitel werden die Daten beschrieben und explorativ untersucht, auf deren Basis das Klassifikationsproblem (siehe Kapitel 1.2) beschrieben und gelöst (siehe folgende Kapitel) werden soll. Dabei handelt es sich um den Common-Crawl-Datensatz (siehe Abschnitt 3.1) aus dem die Nachrichtentexte extrahiert werden, und die Knowledge Base DBpedia (siehe Abschnitt 3.2), die Brancheninformationen zu Firmen enthält.

3.1 CommonCrawl

Die Web-Texte in Form von Nachrichtenartikel, die als Input für das Klassifikationsproblem dienen, stammen aus dem populären Crawling-Projekt *Common Crawl (CC)* [Com19]. Grund für die Wahl von CC ist, dass es sich dabei um die größte frei zugängliche Quelle von extrahierten Webseiten und deren Metadaten handelt. Damit müssen diese im Rahmen dieser Arbeit nicht eigenständig gecrawlt werden. Das Projekt wird von der Non-Profit-Organisation CC geleitet, die eine frei zugängliche Teil-Kopie des Internets für Forschungszwecke und Analysen erstellt. Das CC-Projekt ist 2007 gestartet und betreibt unter dem Namen CCBot einen eigenen Webcrawler, der auf *Apache Nutch* basiert und *Apache Hadoop* nutzt. Zum Zeitpunkt November 2018 beinhaltet das aktuellste CC-Archiv 2,6 Milliarden Webseiten und 220 Terabytes an unkomprimierten Daten, die zwischen dem 12. und dem 22. November gecrawlt wurden. Der November-2018-Crawl beinhaltet 640 Millionen neue URLs, die in keinem vorherigen Crawl vorkommen. Diese neuen URLs stammen aus Sitemaps, RSS- und Atom Feeds der 60 Millionen höchst gerankten Domains des Webgraphen der vorherigen drei Crawls[1] und von Links, die maximal 10 *Sprünge* von den Top 40 Millionen Domains des Webraphen-Datensatzes entfernt sind. Weitere URLs stammen von Zufallsstichproben aus dem Oktober-2018-Crawl und von 50 Millionen externen Links aus Zufallsstichproben von Wikipedia-Daten[2].

[1] http://commoncrawl.org/2018/11/web-graphs-aug-sep-oct-2018/
[2] https://dumps.wikimedia.org/

3.1.1 Format

Die Ergebnisse des Crawls liegen jeweils in drei verschiedenen Formaten vor [Com19]:

- **Web ARChive (WARC):** Das WARC-Format ist das Rohformat der gecrawlten Daten. Es speichert die HTTP-Response der kontaktierten Webseite, Informationen zum Request der Webseite und Metadaten zum Crawl-Prozess selbst. Ein Ausschnitt aus einer WARC-Datei ist im Anhang unter Abbildung A.1 zu finden.
- **WAT:** WAT-Dateien enthalten alle wichtigen Metadaten aus den WARC-Dateien. Sollte die gecrawlte Webseite HTML beinhalten, so befinden sich in den Metadaten unter anderem der HTTP-Header und die auf der Webseite enthaltenen Links. Die Informationen werden als JSON abgespeichert, um diese einfacher zugreifbar zu machen, indem man sie beispielsweise in die Python-Datenstruktur *Dictionary* lädt. Das Grundgerüst einer WAT-Datei ist im Anhang unter Abbildung A.2 zu finden.
- **WET:** Die WET-Dateien enthalten ausschließlich den auf den Webseiten enthaltenen Klartext mit zugehörigen Metadaten, wie die URL der Webseite und die Länge des Klartextes. Ein Ausschnitt aus einer WARC-Datei ist im Anhang unter Abbildung A.3 zu finden.

3.1.2 Zugriff

Der CC-Datensatz wird auf *Amazon Web Services (AWS)* gehostet. Hier werden die einzelnen monatlichen Crawls jeweils in einem separaten Pfad innerhalb des CC *Buckets* verwaltet. Auf die Daten des Crawls kann per HTTP per

https://commoncrawl.s3.amazonaws.com/crawl-data/CC-MAIN-<JAHR>-<CRAWL-NUMMER>/

oder direkt über AWS per

s3://commoncrawl/crawl-data/CC-MAIN-<JAHR>-<CRAWL-NUMMER>/

zugegriffen werden.

Die Daten des Crawls sind dann wiederum in Segmente und innerhalb der Segmente nach ihrem Dateityp (WARC, WAT und WET) eingeteilt. Abgelegt werden die Daten als 1 GB große komprimierte Dateien im *gzip*-Format. Ein kompletter Pfad zu einer WARC-Datei in AWS sieht folgendermaßen aus:

s3://commoncrawl/crawl-data/CC-MAIN-2018-43/segments/1539583508988.18/warc/CC-MAIN-20181015080248-20181015101748-00000.warc.gz

Die Zahlenfolgen in den Segment- und Dateinamen setzen sich aus Zeitstempeln zusammen, aus denen der Zeitpunkt der Erstellung abgeleitet werden kann. Mit jedem Crawl werden Datei-Listen für die Segmente, die einzelnen Datentypen und die Index-Dateien bereitgestellt.

Damit die riesige Datenmenge des CC gezielt durchsucht werden kann, wird für jeden einzelnen Crawl ein Index bereitgestellt. Der Index ermöglicht das Auffinden bestimmter Seiten im Crawl, wobei nach einem URL, einem URL-Präfix, einer Subdomain oder einer Top-Level-Domain gesucht werden kann. Eine genaue Beschreibung des Indexes mit Verlinkungen auf die Dokumentation ist unter [Kre15a] zu finden. Die Index-Dateien stehen zum Download bereit und können selbst verwaltet werden, CC stellt jedoch auch einen eigenen Index-Server [Pyw19] bereit, der per Web-Oberfläche oder *Application Programming Interface (API)* [Kre15b] zugreifbar ist. Per API ist es möglich, neben den Such-URLs noch weitere Parameter, wie beispielsweise Filter, Bereichsabfragen oder Sortierungen, anzugeben. Als Ergebnis einer Anfrage erhält der User eine Liste mit allen passenden Seiten innerhalb des Crawls mit Angaben wie der genauen URL, in welchem Segment und welcher Datei sich die Seite befindet, an welcher Stelle die Seite innerhalb der Datei anfängt und wie lang die Seite ist. Mit einem HTTP-Request können die einzelnen Seiten dann gezielt angefragt und geladen werden.

3.1.3 Exploration

Die Quellen des CC-Datensatzes sind heterogen, was zu einem ebenfalls heterogenen Inhalt im CC führt. Enthalten sind unter anderem Nachrichten-Seiten, Soziale Netzwerke, Online Handelsplattformen und Firmen-Homepages. Im Kontext dieser Arbeit sind die CC-Inhalte relevant, die englischsprachige Texte enthalten, in denen Firmen referenziert werden. Um den Anteil der relevanten Daten im CC-Datensatz zu ermitteln, wird ein

Teil des CC-Datensatzes auf diese Inhalte untersucht. Dazu werden *WET*-Dateien (siehe Abschnitt 3.1.1) des verwendeten November-2018-Crawls iterativ durchlaufen. Für jedes Record wird zunächst mit der Python-Bibliothek *langdetect* bestimmt, ob dieses englische Texte enthält. Wenn das der Fall ist, wird eine NER und ein EL durchgeführt, um zu ermitteln, ob und gegebenenfalls welche Firmen im Text enthalten sind. Dabei werden die gleichen Tools eingesetzt, wie im Klassifikationsszenario (siehe Kapitel 5.2). Das Durchlaufen einer einzelnen WET-Datei dauert zwischen 9 und 11 Stunden. Das Verarbeiten einer einzelnen Datei kann nicht parallelisiert werden, da die Datei aufgrund ihres Formates sequentiell durchiteriert werden muss. Das Identifizieren der relevanten Inhalte auf allen 56.000 Dateien des Crawls ist kosten- und rechenintensiv. Daher wird ein Teil des Datensatzes untersucht und die Ergebnisse zum Abschätzen auf den gesamten Crawl hochgerechnet. Um möglichst viele Dateien in kurzer Zeit auswerten zu können, wird die Suche nach relevanten Daten auf einem *Apache Spark 2.1.0*-Cluster auf *Amazon AWS* durchgeführt. Der Grund dafür ist, dass so mehrere Dateien gleichzeitig auf mehreren Servern verarbeitet werden können und die Ergebnisse anschließend zusammengetragen werden. Der Aufbau der Datenverarbeitung auf Spark ist in Abbildung 3.1 vereinfacht dargestellt. Ein Server im Spark-Cluster arbeitet als Master-Knoten. Auf diesem Knoten werden keine Daten verarbeitet, sondern hier findet die Koordination des Spark-Clusters statt. Auf den Servern, die als Worker-Knoten ausgewiesen sind, können bei der Ausführung eines Spark-Jobs parallel Spark-Worker-Prozesse ausgeführt werden. Innerhalb der Spark-Worker-Prozesse können wiederum mehrere Aufgaben (Tasks) parallel ausgeführt werden. Genauere Informationen zu Amazon AWS und Apache Spark sind in deren Dokumentationen [Ama19] und [Apa19] zu finden. Auf jedem Worker-Knoten im verwendeten Cluster laufen Stanford CoreNLP und DBpedia Spotlight, damit die Daten lokal auf dem Server ausgewertet werden können, ohne Daten über das Netz austauschen zu müssen. Der Spark-Job zur Auswertung der CC-Daten ist in *Python 2.7* geschrieben, da in dieser Version alle benötigten Python-Bibliotheken unterstützt werden. Er basiert auf der Klasse *CCSparkJob* der Implementierung *cc-pyspark*[3]. Der Job erhält als Input eine Textdatei mit allen auf Amazon AWS gehosteten CC-WET-Dateien, die verarbeitet werden sollen. Das Durchsuchen einer Datei stellt einen Task dar. Somit können mehrere WET-Dateien parallel innerhalb und zwischen verschiedenen Spark-Workern, die sich gleichzeitig auf mehreren Servern befinden können, verarbeitet werden. Die Ergebnisse der einzelnen Tasks werden am Ende des Spark-Jobs zusammengetragen und aggregiert.

[3]https://github.com/commoncrawl/cc-pyspark

Das aggregierte Ergebnis bildet den Output des Jobs.

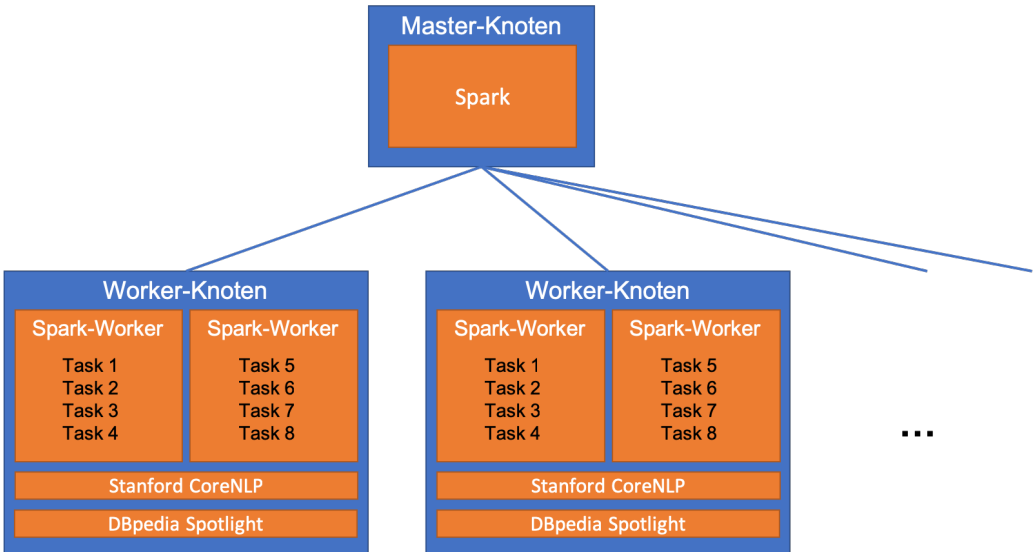


Abbildung 3.1: Architektur der Identifikation der relevanten Inhalte in CC per Spark auf Amazon AWS.

Die Ergebnisse für 712 untersuchte Dateien sind in Tabelle 3.1 zu sehen. Eine WET-Datei enthält durchschnittlich 44.511 Records. In 45% der Records werden englische Texte entdeckt. In 88% der englischen Records werden mit einer NER Firmen identifiziert und in 70% der englischen Records Firmen mit DBpedia gelinkt. Auf den kompletten Datensatz des November-2018-Crawls hochgerechnet bedeutet dies, dass schätzungsweise in 986.720.000 Records Firmen enthalten sind und 784.616.000 Records mit Informationen aus DBpedia angereichert werden können. Die Struktur und Qualität der Daten, sowie mögliche Fehlklassifikationen in der NER und dem EL werden dabei nicht berücksichtigt.

Kennzahl	Gemittelte Ergebnisse	Hochrechnung gesamter Datensatz
# Records:	44.511	2.492.616.000
# Englische Records:	20.040	1.122.240.000
# Records mit gespotteten Firmen (NER):	17.620	986.720.000
# Records mit gelinkten Firmen (EL):	14.011	784.616.000

Tabelle 3.1: Ergebnisse der Untersuchung des CC-Datensatzes auf relevante Inhalte.

3.2 DBpedia

Bei *DBpedia* [DBp19] handelt es sich um ein Gemeinschaftsprojekt der 2014 gegründeten *DBpedia Association*, die gegenwärtig in Leipzig ansässig ist und die an die Non-Profit-Organisation *Institute for Applied Informatics* angegliedert wurde. Im DBpedia-Projekt wird multilinguales Wissen in Form von strukturierten Informationen aus Wikipedia extrahiert und in einer *Knowledge Base* (im deutschen: *Wissensdatenbank* oder *Wissensbasis*) gespeichert. In der DBpedia Knowledge Base werden die Informationen öffentlich für Forschungszwecke als Open Knowledge Graph zugreifbar gemacht. Aufgrund der weiten Verbreitung in der Forschung und des vorhandenen EL-Tools *DBpedia Spotlight* (siehe Kapitel 5.1.2), das passend für DBpedia entwickelt wurde, wird die Knowledge Base in dieser Arbeit verwendet.

Für die Extraktion der Informationen [LIJ+15] werden Wikipedia-Editionen in 111 verschiedenen Sprachen verwendet. Informationen, die sich in Wikipedia befinden, sind teilweise strukturiert, wie beispielsweise durch Templates für Infoboxen. Infoboxen sind die Hauptquelle für die DBpedia-Extraktion, da die Inhalte in die DBpedia-Ontologie gemappt werden können. Extraktoren kommen aus dem Bereich Feature Extraction, die einzelne Features, wie Label oder Geokoordinaten extrahieren sollen oder Extraktoren aus dem Bereich Statistical Extraction, die Methoden aus dem Natural Language Processing verwenden.

3.2.1 Format

Die Daten liegen als Resource-Description-Framework (RDF)-Tripel in der Form

<SUBJECT> <PREDICATE> <OBJECT>, z.B.
<dbr:Apple_ Inc.> <dbo:industry> <dbr:Consumer_ electronics>

vor, wobei *dbr* und *dbo* für die *Uniform Resource Identifiers (URIs)* *http://dbpedia.org/resource/* (Präfix für eine Resource, die einen Wikipedia-Artikel repräsentiert) und *http://dbpedia.org/ontology/* (Präfix für die DBpedia Ontologie) stehen. Sie werden im Kontext dieser Arbeit als Bezeichner für zwei verschiedene Namensräume gesehen, die beide unabhängig voneinander dazu eingesetzt werden können, um Eigenschaften einer Entität darzustellen. Die Spezifikation von RDF ist auf der Webseite des *World Wide Web Consortium* (W3C) dokumentiert [W3C14]. Ein Ausschnitt aus der Darstellung einer Firma in DBpedia ist im Anhang in Abbildung A.8 zu sehen.

3.2.2 Zugriff

Die RDF-Tupel können heruntergeladen und selbst in einer Graph-Datenbank verwaltet werden. Ansonsten sind sie zugreifbar per HTML-Browser (siehe Anhang Abbildung A.8), RDF-Browser oder per SPARQL-Client. Bei *SPARQL* handelt es sich um eine *SQL*-ähnliche Abfragesprache für RDF-Graphen, deren Spezifikation und Dokumentation ebenfalls bei W3C liegt [W3C13]. DBpedia's *SPARQL* Web Service[4] kann per HTML-Browser oder per GET- und POST-Requests (REST-basiert) genutzt werden.

3.2.3 Exploration

In DBpedia werden Unternehmen primär über das Property *rdf:type* mit dem Value *dbo:Company* identifiziert. Die Anzahl der Firmen, die zum Zeitpunkt dieser Arbeit im per *dbo:Company* englischen DBpedia referenziert werden, beläuft sich auf 109.625 unterschiedliche Unternehmen weltweit. Etwas weniger als die Hälfte der Unternehmen, genau 53.079 Firmen, haben über die Eigenschaften *dbo:industry* und *dbp:industry* (steht für den Namensraum DBpedia Property) eine oder mehrere Branchen zugewiesen [5]. Mit *dbo:industry* wird bei 41.954 Unternehmen als Branche eine Klasse verwendet, wie beispielsweise *dbr:Internet*, während bei 11.139 Unternehmen mit *dbp:Industry* als Value ein

[4] `https://dbpedia.org/sparql`

[5] Die Präfixe „dbo“ und „dbp“ dienen der Einteilung innerhalb von DBpedia in zwei verschiedenen Namensräumen und haben keine weitere Relevanz für diese Thesis

String verwendet wird. Dabei entsteht eine Schnittmenge von 14 Entität bei denen beide Eigenschaften verwendet werden. Insgesamt werden über *dbo:industry* 6.757 Branchen referenziert, während es bei *dbp:industry* 5.829 Branchen sind. Eine Übersicht der genannten Aufteilung der Branchen ist in Abbildung 3.2 zu sehen.

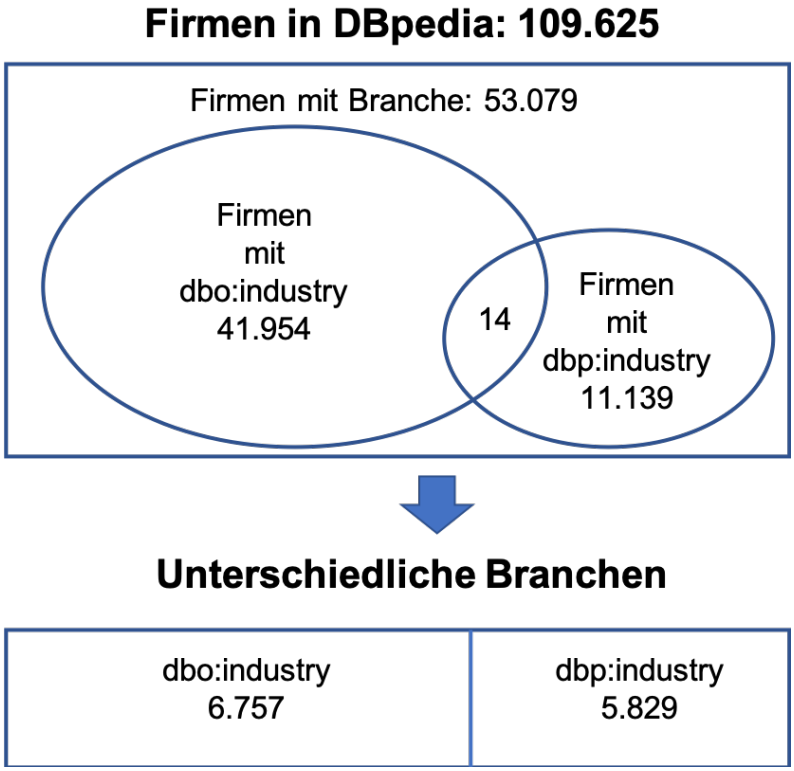


Abbildung 3.2: Aufteilung der Branchen in DBpedia.

Die große Anzahl an Branchen kommt dadurch zustande, dass viele Redundanzen auftreten, die bereits in den 30 häufigsten Branchen der jeweiligen Properties zu erkennen sind (siehe Abbildung 3.3), oder noch deutlicher in einer Übersicht aller verwendeten Branchen dieser Arbeit (siehe Anhang Abbildung A.6 und Abbildung A.7). So treten manche Branchen mehrfach unter verschiedener Schreibweise auf, wie bei *Financial_services* und *Financial_Services* oder unter unterschiedlicher Bezeichnung wie *Video_game_industry* und *Computer_and_video_game_industry*. Des Weiteren ist die Anzahl der Branchen auf die verschiedenen Granularitätsstufen der Bezeichner zurückzuführen. Während Branchen wie *Transport* und *Finance_and_Insurance* sehr allgemein gefasst sind, weisen Branchen wie *Rail_transport* oder *Real_Estate* eine feinere Granularität auf. Eine Hierarchie zwischen den Branchen wird bei DBpedia nicht berücksichtigt.

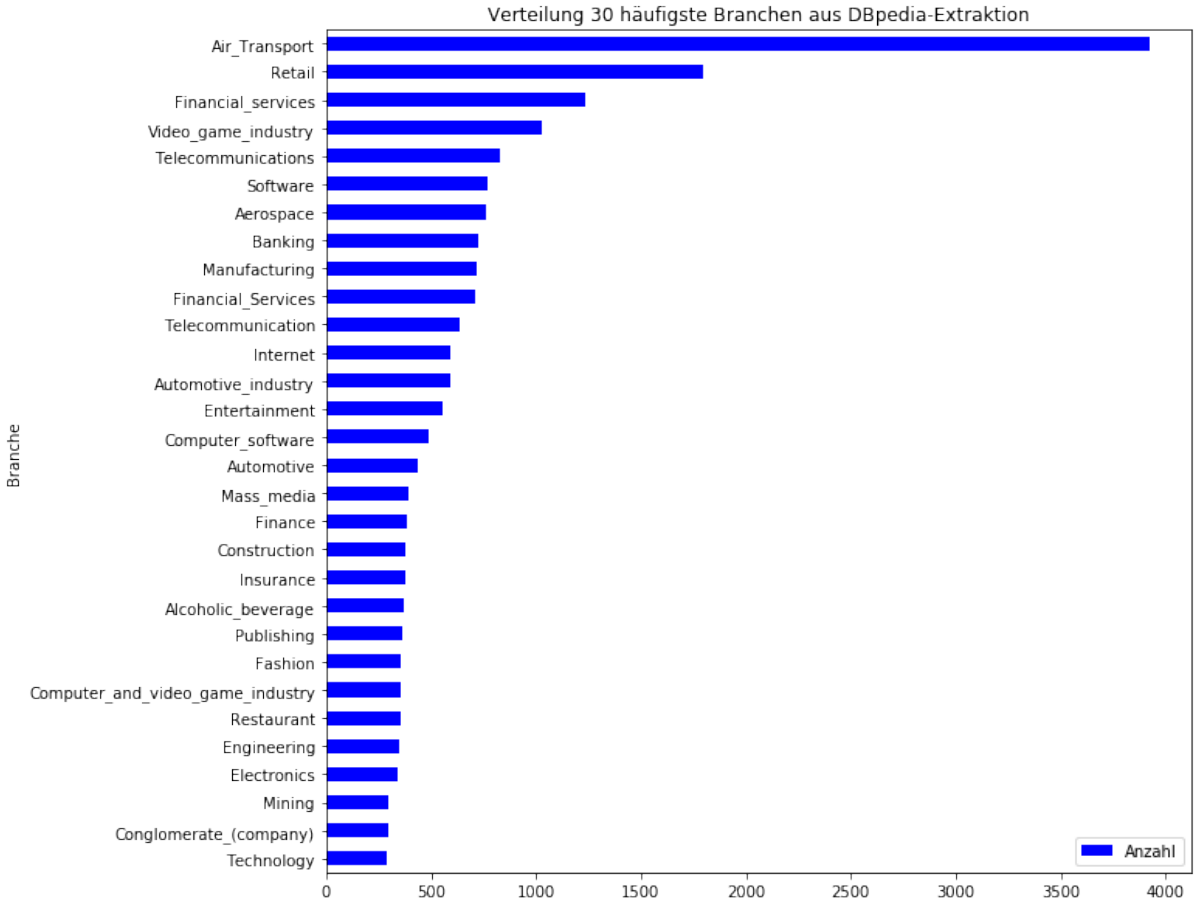


Abbildung 3.3: Häufigkeitsverteilung der 30 häufigsten DBpedia-Branchen.

Zwischen den beiden Eigenschaften kommt es zu einer Übereinstimmung, sodass zwischen den ausgewählten 85 *dbo:Industry*-Branchen (Grund für die Auswahl siehe Kapitel 4.4.1) und den 57 *dbp:Industry*-Branchen eine Schnittmenge von 40 identischen Branchen-Bezeichnungen und eine Vereinigungsmenge von 102 Branchen (siehe Anhang Abbildung A.6 und Abbildung A.7) entsteht.

Bei der Verteilung der Branchen pro Unternehmen (siehe Abbildung 3.4) fällt auf, dass der Großteil der Firmen in DBpedia einer einzelnen Branche zugeordnet wird, wobei für einzelne Firmen bis zu 20 Branchen angegeben werden. Anteilmäßig werden 86,24% der Firmen eine einzige Branche, 9,56% zwei Branchen und 4,2% der Firmen mehr als zwei Branchen zugeordnet. Die genauen Kennzahlen der Branchen pro Unternehmen befindet sich im Anhang in Tabelle A.1.

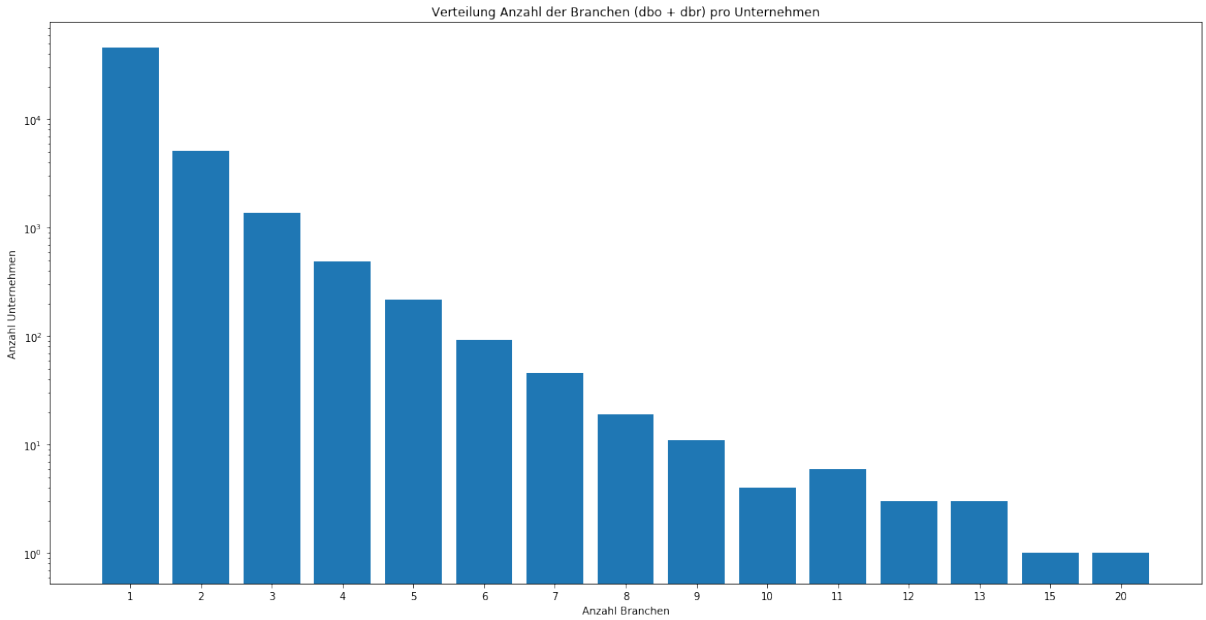


Abbildung 3.4: Häufigkeitsverteilung der Anzahl der Branchen pro Unternehmen (*dbo* + *dbp*) in DBpedia. Die y-Achse ist logarithmiert skaliert.

4 Aufbau des Klassifikations-Szenarios

Dieses Kapitel beinhaltet den Aufbau des Klassifikations-Szenarios. Dieses beginnt in Abschnitt 4.1 mit der Problembeschreibung. In Abschnitt 4.2 wird das Vorgehen aufgezeigt, mit welchen Schritten das Klassifikationsproblem in dieser Arbeit umgesetzt wird. Die einzelnen Schritte werden im Anschluss detailliert dargestellt. Diese lassen sich in drei wesentliche Schichten zusammenfassen. Die erste Schicht bildet die Datenaufbereitung, die in Abschnitt 4.3 thematisiert wird. Es folgt in Abschnitt 4.4 die Schicht der Modellbildung mit all ihren Facetten aus dem Modellbildungsprozess. Den Abschluss des Kapitel stellt Abschnitt 4.5 mit den Metriken zur Auswertung der Modell dar.

4.1 Problembeschreibung

In dieser Arbeit sollen Web-Texte, in denen Firmen auftauchen, nach Branchen klassifiziert werden. Dabei wird angenommen, dass der Text die Branchen der enthaltenen Firmen repräsentiert. Eine Firma kann in einer oder mehreren Branchen tätig sein. In einem Text können mehrere Firmen auftauchen. Daher ergibt sich, dass in einem Text mehrere Branchen repräsentiert sein können. Das Klassifikationsproblem, das sich daraus ergibt, ist ein *Multi-Label Klassifikationsproblem*, in dem einem Text $\mathbf{x}_i$ eine Teilmenge von Labeln Y_i aus einer endlichen Menge von Labeln $L = \{\lambda_j : j = 1...q\}$ zugeordnet wird [TKV09].

Für den Aufbau des Klassifikationsproblems werden Trainings- und Testdaten benötigt. Dies bedeutet, dass Web-Texte benötigt werden, für die die enthaltenen Branchen bekannt sind. Die Web-Texte, die für das Klassifikationsproblem verwendet werden, stammen aus dem CC-Datensatz (siehe Kapitel 3.1). Um die repräsentierten Branchen in den Texten zu ermitteln, muss identifiziert werden, in welchen Texten sich Entitäten befinden und im nächsten Schritt, zu welchen Firmen Informationen in der Knowledge Base bekannt sind.

Dazu werden mit einer NER Firmen in den Texten identifiziert und per EL diejenigen mit der Knowlege Base gelinkt, zu denen Informationen enthalten sind. Die Branchen der gelinkten Firmen, werden den Texten zugeordnet. Auf den so entstandenen Daten werden Modelle trainiert, die das Multi-Label Klassifikationsproblem lösen sollen.

4.2 Vorgehen

Das Klassifikationsproblem lässt sich auf mehrere Schritte bzw. Schichten herunterbrechen, die in den folgenden Unterkapiteln beschrieben werden. Eine grafische Darstellung des gesamten Klassifikationsprozesses dieser Arbeit ist in Abbildung 4.1 zu sehen. Die erste Schicht ist die Daten-Schicht, in der sich die Rohdaten, eine Knowledge Base und die CC-Daten befinden. Eine ausführliche Beschreibung der Daten und der Daten-Quellen ist in Kapitel 3 zu finden. Die zweite Schicht ist die Datenaufbereitung, die in Kapitel 4.3 beschrieben wird. Hier werden zur Vorbereitung der Daten etablierte Tools aus dem *Natural Language Processing* eingesetzt, um Entitäten in Nachrichtenartikeln zu erkennen und mit der Knowledge Base abzugleichen. Der Kern der Arbeit liegt in der Modellbildung- und der Auswertungsschicht. Hier wird mit den Daten aus den darüber liegenden Schichten ein Data Mining Prozess durchgeführt, angelehnt an den *Knowledge Discovery in Databases (KDD)*-Prozess, wie er von Fayyad et al. [Fay96] beschrieben wird und dem *CRoss-Industry Standard Process for Data Mining (CRISP-DM)* Referenzmodell von Shearer [She00]. Die einzelnen Schritte des angewendeten Prozesses werden in den Unterabschnitten von Abschnitt 4.4 und Abschnitt 4.5 beschrieben.

Abbildung 4.1: Grafische Darstellung Aufbau Klassifikationsproblem.

4.3 Datenaufbereitung

Die Datenaufbereitung setzt sich, wie in Abbildung 4.1 zu sehen ist, aus zwei Teilschritten zusammen:

News Extraktion

Im ersten Schritt der Datenaufbereitung werden Nachrichtenartikel aus dem CC-Datensatz extrahiert. Um die Datenmenge einzugrenzen und nicht die kompletten Crawl-Daten durchsuchen zu müssen, werden nicht alle Nachrichtenartikel aus dem gesamten CC-Datensatz extrahiert, sondern über den CC-Index gezielt Artikel von vorher festgelegten Nachrichtenseiten, die sich in einem einzelnen Crawl befinden. Insgesamt werden 21

verschiedene URLs zu 17 verschiedenen Nachrichtenseiten, die Nachrichten aus der Wirtschaft enthalten, verwendet. Nach diesen wird im November-2018-Crawl gesucht. Da die Anzahl der für eine URL im Crawl enthaltenen Nachrichtenartikel zwischen den verschiedenen Monatscrawls stark schwanken kann, werden auch URLs mit wenigen Treffern in dieser Arbeit beibehalten. Eine Übersicht mit allen verwendeten URLs ist im Anhang in Tabelle B.1 zu finden. Die URLs beinhalten Wildcards und sehen wie folgende URL aus:

*https://www.washingtonpost.com/world/**

Als Ergebnis für jeden Treffer im Crawl liefert die Index-Suche detaillierte Informationen zu der gefunden Seite, wie im Folgenden im JSON-Format zu sehen ist:


```
{
        "urlkey": "com,washingtonpost)/world/
                12-turkish-troops-killed-in-chopper-
                crash-in-kabul/2012/03/16/
                giqa3zvzfs_story.html",
        "timestamp": "20180719200800",
        "offset": "1023830717",
        "mime": "text/html",
        "digest": "ZQJ64CSJTNX3TLY4TU7XHPHL5APGBWJI",
        "mime-detected": "text/html",
        "filename": "crawl-data/CC-MAIN-2018-30/segments/
                1531676591216.51/warc/
                CC-MAIN-20180719183926-20180719203926-
                00498.warc.gz",
        "length": "32389",
        "url": "https://www.washingtonpost.com/world/
                12-turkish-troops-killed-in-chopper-
                crash-in-kabul/2012/03/16/
                gIQA3ZvzFS_story.html",
        "status": "200"
}
```


Über das Feld *urlkey* bzw. *url* kann zusammen mit dem Feld *timestamp*, das den Crawling-Zeitpunkt der Seite festhält, eine Seite eindeutig zugeordnet werden. Das Format der gecrawlten Seite wird im Feld *mime* dokumentiert. Eine Prüfsumme des Datensatzes ist im

Feld *digest* hinterlegt. In welchem Segment und welcher Datei die gecrawlte Seite abgelegt ist, ist im Feld *filename* hinterlegt. Mit den Feldern *offset* und *length* kann die Seite innerhalb der Datei identifiziert und extrahiert werden. Der HTTP-Statuscode ist im Feld *status* hinterlegt.

Anhand der Ergebnisse der Indexsuche werden die gefundenen Seiten in ihrem Rohformat (WARC-Format) aus dem Crawl geladen. Außerdem wird für die folgenden Schritte aus jeder Seite der Nachrichtentext extrahiert, sodass dieser als Plaintext vorliegt.

Named Entity Recognition und Entity Linking
Die zuvor extrahierten Nachrichtenartikel werden mit dem Einsatz von etablierten Tools auf enthaltene Entitäten untersucht. Als Entitäten werden in dieser Arbeit nur Firmen betrachtet. Mit einem NER-Tool wird anhand des gegebenen Textes festgestellt, welche Terme eine Firma darstellen. Beim Einsatz eines EL-Tools wird ebenfalls eine NER durchgeführt, jedoch dient hier als Grundlage nicht nur der gegebene Text, sondern auch die Inhalte der verwendeten Knowledge Base. Es werden dabei nur Terme identifiziert, die eine Firma darstellen, die in der Knowledge Base vorhanden ist. Für die gefundenen Terme wird eine Referenz auf das passende Objekt der Knowledge Base angegeben. In dieser Arbeit wird sowohl ein EL, als auch eine NER durchgeführt, um eine Aussage über die Performance des ELs und über den Umfang der enthaltenen Firmen in der Knowledge Base treffen zu können.

4.4 Modellbildung

Die Daten und Informationen, die in der Datenaufbereitung extrahiert werden, dienen als Input für die Modellbildung. Hierzu werden sie, wie in Abschnitt 4.4.1 beschrieben, entsprechend aufbereitet.

Mit den aufbereiteten Daten wird in Anlehnung an den *Knowledge Discovery in Databases*-Prozess, wie er von Fayyad et al. [Fay96] beschrieben wird, und dem *CRoss-Industry Standard Process for Data Mining (CRISP-DM)* Referenzmodell von Shearer [She00] ein selbst definierter DM-Prozess durchgeführt, wie er in Abbildung 4.2 zu sehen ist. Die Abbildung konkretisiert den Teil des Klassifikationsproblems, der in Abbildung 4.1 mit einem roten Kasten umrandet ist. Im ersten Schritt des Prozesses werden die aufbereiteten Daten genauer untersucht (siehe Abschnitt 4.4.2). Anschließend werden die Daten für die

Modellbildung vorverarbeitet. Die genauen Verarbeitungsschritte sind in Abschnitt 4.4.3 dargestellt. Aus den vorverarbeiteten Daten werden wie in Abschnitt 4.4.4 beschrieben, verschiedene Features extrahiert. Die Features und Kombinationen aus diesen dienen als Input für das Modell-Training. Die verschiedenen Modelle und deren Parameter werden in Abschnitt 4.4.5 aufgezeigt. Anhand der Performance der Modelle auf Validierungsdaten, unter Verwendung verschiedener Metriken, wie sie in Abschnitt 4.5 beschrieben sind, werden die Parameter im DM-Prozess angepasst. Diese Anpassungen können in allen Prozessschritten vorgenommen werden. Insgesamt handelt es sich bei dem durchgeführten DM-Prozess, wie bei den Referenzmodellen [Fay96][She00], um einen iterativen Prozess. Dieser kann, abhängig von den gewonnenen Erkenntnissen, mehrfach komplett oder teilweise durchlaufen werden. Dabei kann von jedem Prozessschritt zu einem beliebigen vorherigen Schritt zurückgesprungen werden, und es können Anpassungen vorgenommen werden.

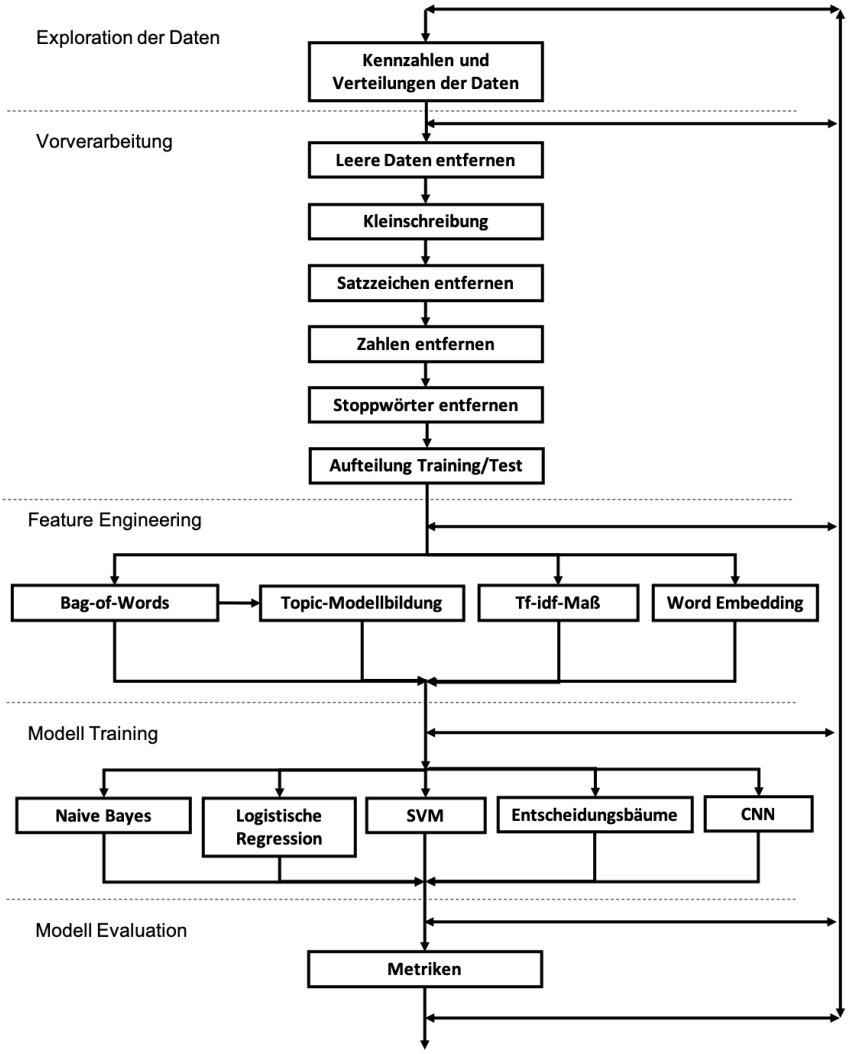


Abbildung 4.2: Grafische Darstellung Aufbau Modellbildung.

4.4.1 Datensatz

Bei dem zu lösenden Problem handelt es sich, wie zu Beginn des Kapitels beschrieben (siehe Abschnitt 4.1), um ein Multi-Label Klassifikationsproblem. Da die Anzahl der verschiedenen Branchen in DBpedia sehr groß ist und viele Redundanzen enthält (siehe Kapitel 3.2.3), werden die Klassen für das Klassifikationsproblem reduziert. Dazu wird im ersten Schritt eine Vorauswahl der DBpedia-Branchen getroffen, die in Abbildung 4.3 dargestellt ist. Für *dbo:industry* werden die 85 häufigsten Branchen verwendet, die zusammen über 50% der Referenzen ausmachen und alle mindestens bei 100 Firmen referenziert werden. Das Property *dbp:industry* wird spärlicher verwendet. Daher ist hier das Kriterium

bei der Auswahl, dass mindestens 20 Referenzen vorhanden sein müssen. Dieses Kriterium führt zu einer Auswahl der 57 häufigsten *dbp:industry*-Branchen. Mit dieser Auswahl wurde die Anzahl der verschiedenen Branchen auf 102 (Vereinigungsmenge) reduziert. Eine Übersicht der verwendeten Branchen ist in Anhang in Abbildung A.6 zu finden. Da zum einen der Rechenaufwand für 102 Klassen bei dem Training der Modelle hoch ist und zum anderen weiterhin Redundanzen in den Klassen auftreten, wird die Anzahl der verschiedenen Klassen, per Mapping auf eine kleinere Klassenstruktur, weiter reduziert. Als neue Klassenstruktur wird für diese Arbeit die Branchen-Einteilung der Nachrichtenseite *Marketwired*[1] gewählt. Diese besteht aus 24 abstrakten Branchen. Das Mapping der DBpedia-Branchen auf die Marketwire-Branchen findet manuell statt. Als Entscheidungshilfe für die Einteilung dienen die Sub-Branchen, die ebenfalls bei Marketwired angegeben sind. Die erstellten Mappings sind im Anhang B.1 zu finden. Für die DBpedia-Branche *Conglomerat* wird im Mapping eine eigene Klasse angelegt, da sie sich nicht in die Marketwire-Branchen einordnen lässt. Es wird außerdem eine Default-Klasse *MISC* (Abkürzung für *Miscellaneous* zu Deutsch *Verschiedenes*) eingeführt, deren Nutzung im nächsten Absatz beschrieben wird. Durch das Mapping wird die Anzahl der Klassen auf 26 reduziert und gleichzeitig die Redundanz entfernt, da verschiedene Schreibweisen oder Bezeichnungen für eine bestimmte DBpedia-Branche nun auf die gleiche Branche gemappt werden, und alle gemappten Branchen auf der gleichen Abstraktionsebene sind. Ein weiteres Ziel, das mit dem Mapping auf die Marketwired-Branchen verfolgt werden soll, ist das Vermeiden von Korrelationen zwischen den Klassen. Ohne Korrelationen sind die Branchen klarer voneinander getrennt, und Klassifikatoren können eindeutige Vorhersagen treffen.

[1] `http://www.marketwired.com/news_room/`

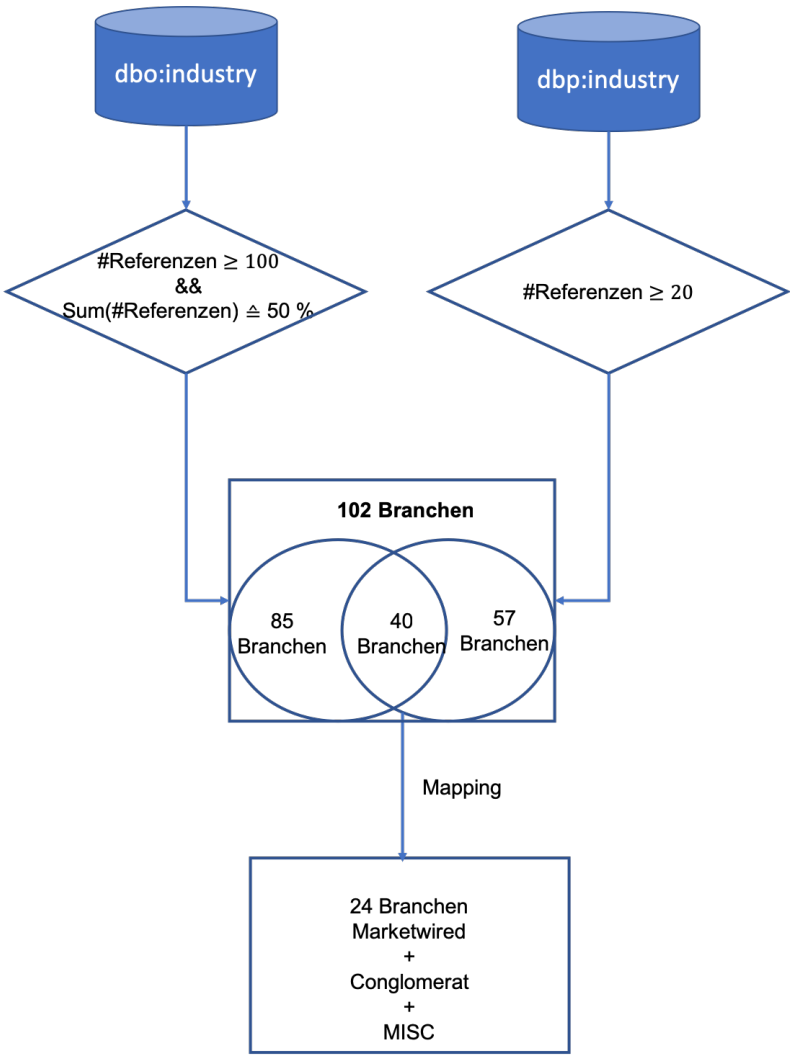


Abbildung 4.3: Grafische Darstellung der Kriterien der Klassenauswahl.

Anhand der Ergebnisse aus dem EL und dem erstellten Klassen-Mapping werden die Trainingsdaten erstellt, wie in Abbildung 4.4 dargestellt. Dazu werden die in den Nachrichtenartikeln erkannten und mit DBpedia gelinkte Firmen nach ihrer hinterlegten Branche abgefragt. Sollte über *dbo:industry* und *dbp:industry* keine Branche hinterlegt sein, so wird der Firma die Default-Klasse *MISC* zugewiesen. Im nächsten Schritt werden den DBpedia-Branchen über die beschriebene Mapping-Tabelle eine Klasse zugewiesen. Wird in der Mapping-Tabelle keine passende Klasse für eine DBpedia-Branche gefunden, so wird ebenfalls die Default-Klasse *MISC* zugeordnet. Im Anschluss an das Mapping werden die Trainingsdaten im JSON-Format generiert, in denen jeder Nachrichtenartikel ein JSON-Objekt darstellt. Die Klassen werden für die Modellbildung binär kodiert. Dies

bedeutet, dass jede der 26 Klassen (Klassenübersicht siehe Anhang Abbildung B.1) als Key-Feld verwendet wird und als Value eine 1 zugewiesen bekommt, falls mindestens eine der im Nachrichtenartikel gelinkten Firmen die jeweilige Klasse aufweist. Andernfalls wird der Value für die Klasse auf 0 gesetzt. Als weitere Felder im JSON-Objekt werden die URL und der Text des Nachrichtenartikels angegeben.

Für die Modellbildung werden zwei verschiedene Arten von Trainingsdaten generiert. Ein Datensatz auf der Basis der gesamten Dokumente, wie oben beschrieben, und ein Datensatz, bei dem nicht die gesamten Dokumente, sondern nur einzelne Paragraphen verwendetet werden.

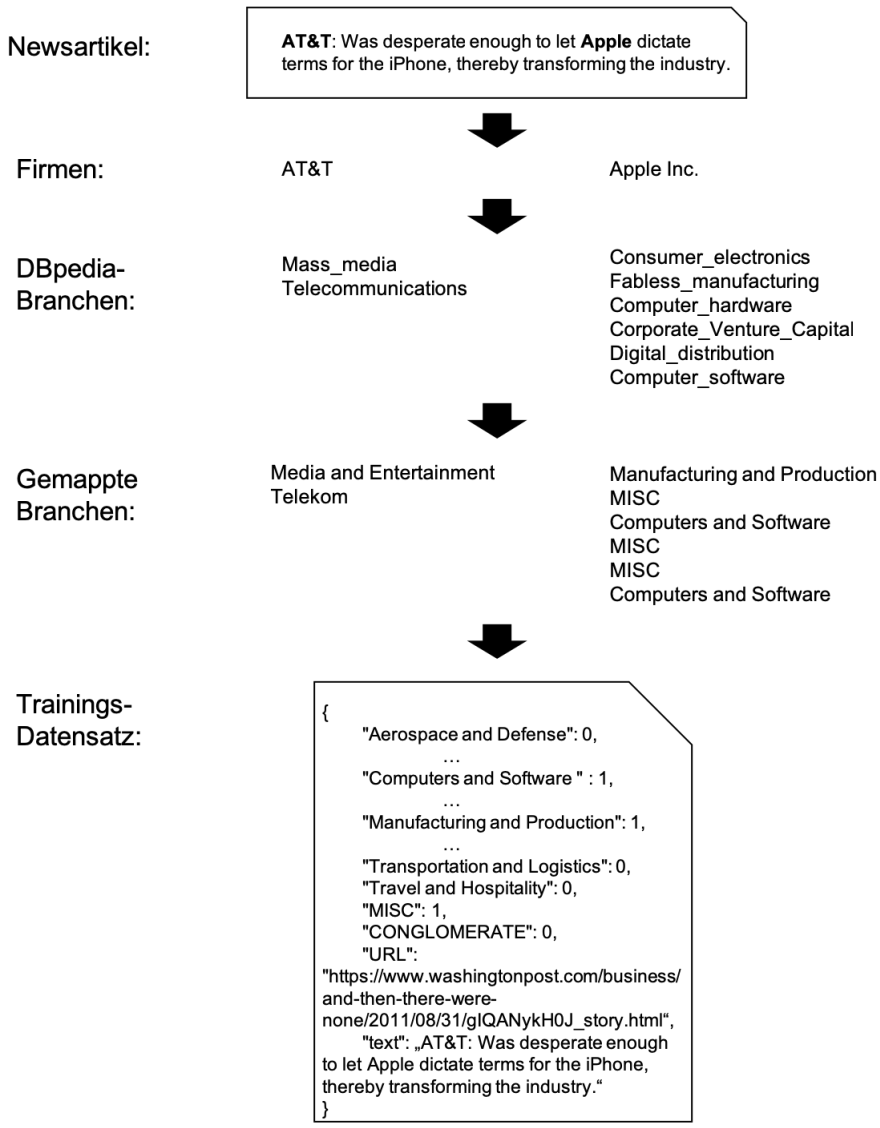


Abbildung 4.4: Grafische Darstellung Generierung Trainingsdaten.

4.4.2 Datenexploration

Der erste Schritt der Modellbildung nach der Erstellung der Trainingsdaten ist die explorative Untersuchung der vorhandenen Daten. Dabei sollen unter anderem folgende Punkte untersucht werden, um ein besseres Verständnis von den Daten zu bekommen:

- Die Verteilung der Entitäten in den Trainings- und Testdaten.
- Die Dimensionalität der Daten.
- Die Verteilung der Branchen.
- Die Verteilung der Anzahl der Branchen pro Dokument (Nachrichtenartikel oder Paragraph).
- Die Verteilung der Anzahl der Firmen pro Dokument (Nachrichtenartikel oder Paragraph).

Außerdem soll die paarweise Korrelation zwischen den verschiedenen Labeln untersucht werden. Ein einfaches Maß zum Ermitteln der Korrelation zwischen zwei binären (dichotomen) Variablen stellt der *Phi-Koeffizient* dar, der von Karl Pearson eingeführt wurde [Cra46, S. 282]. Das Auftreten zweier binärer Variablen lässt sich in einer *Vierfeldertafel* (*2x2 Kontingenztafel*) darstellen, wie sie in Tabelle 4.1 zu sehen ist.

	$\boldsymbol{y_2 = 1}$	$\boldsymbol{y_2 = 0}$	**Summe**
$\boldsymbol{y_1 = 1}$	n_{11}	n_{10}	$n_{1\cdot}$
$\boldsymbol{y_1 = 0}$	n_{01}	n_{00}	$n_{0\cdot}$
Summe	$n_{\cdot 1}$	$n_{\cdot 0}$	n

Tabelle 4.1: Kontingenztafel für Phi-Koeffizient zwischen zwei binären Variablen.

Der Phi-Koeffizient kann Werte zwischen -1 und $+1$ annehmen und lässt sich wie der Pearsonsche Korrelationskoeffizient interpretieren. Je näher der Wert für den Koeffizient an -1 oder $+1$ ist, desto größer die Korrelation zwischen den beiden Variablen. Definiert ist der Phi-Koeffizient folgendermaßen:

$$\phi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 0}n_{\cdot 1}}} \tag{4.4.1}$$

4.4.3 Vorverarbeitung

Unter dem Prozessschritt *Vorverarbeitung* werden in dieser Arbeit Arbeitsschritte und Methoden angewendet, um die Datenmenge und das „Rauschen“ durch nicht-relevante Informationen zu reduzieren. Im Folgenden werden die einzelnen Schritte und Methoden aufgelistet und erläutert:

Leere Daten entfernen:
In diesem Schritt sollen alle Dokumente entfernt werden, die keine Klasse zugeordnet bekommen haben. Dies können Dokumente sein, in denen keine Entität vorhanden ist oder solche, die bei der Extraktion der Daten als leere Dokumente geladen wurden. Außerdem werden Klassen entfernt, die in der Trainingsmenge nicht vertreten sind.

Kleinschreibung:
Kleinschreibung als Vorverarbeitungsschritt bedeutet, dass alle im Text verwendeten Terme in Kleinbuchstaben dargestellt werden. Es handelt sich dabei um eine Normalisierung, um die Variabilität der Terme zu verringern und somit auch die Anzahl der Features bei der Feature-Extraktion. Außerdem soll damit bezweckt werden, dass gleiche Wörter auch gleich behandelt werden, unabhängig von der Groß- und Kleinschreibung.

Satzzeichen entfernen:
Das Entfernen der Satzzeichen führt zur Reduzierung der Variabilität der Terme und somit auch zur Reduzierung der Anzahl der Features bei der Feature-Extraktion. Satzzeichen sind für das Klassifikationsproblem dieser Arbeit nicht relevant, da hier ein Zusammenhang von Texten und Branchen gefunden werden soll.

Zahlen entfernen:
Das Entfernen von Zahlen führt zur Reduzierung der Anzahl an Termen und somit zur Reduzierung der Features bei der Feature-Extraktion. Zahlen sind für das Klassifikationsproblem dieser Arbeit nicht relevant, da hier ein Zusammenhang von Texten und Branchen gefunden werden soll.

Stoppwörter entfernen:
Stoppwörter sind Terme, die sehr häufig in einer Sprache und somit in sehr vielen Dokumenten vorkommen. In der englischen Sprache gehören zu den Stoppwörtern Wörter

wie „a“, „to“ oder „be“. Solche Terme liefern nur sehr wenige bis keine Informationen und können daher entfernt werden, um die Datenmenge zu reduzieren [MRS09, Kapitel 2].

Aufteilung Test/Training:
Die Modelle, die im späteren Verlauf trainiert werden, sollen anschließend darauf getestet werden, wie gut sie neue Daten klassifizieren. Damit dies möglich ist, müssen in den Trainingsdaten, wie auch bei den Testdaten, die Branchen der Texte bekannt sein. Aus diesem Grund wird der aufbereitete Datensatz in Test- und Trainingsdaten eingeteilt. Diese Einteilung wird durch zufälliges Ziehen ohne Zurücklegen durchgeführt. Als Verhältnis wird hierbei 80% Trainingsdaten und 20% Testdaten gewählt.

4.4.4 Feature Engineering

Aus den vorverarbeiteten Texten werden die Features generiert, die den Modellen sowohl beim Training, als auch bei der Anwendung, als Input dienen sollen. Im Rahmen dieser Arbeit werden vier Methoden zur Feature-Generierung angewendet, die im Folgenden beschrieben werden:

Bag-of-Words-Modell:
Im *Bag-of-Words (BOW)* Modell [ZC18, Kapitel 3 + 4] wird ein Text durch einen Vektor von Zahlen dargestellt. Für jedes Wort im Vokabular enthält der Vektor einen Eintrag. Der Eintrag gibt an, wie oft das jeweilige Wort im gegebenen Text auftritt. Bei einem Vokabular von n Worten, hat der BOW-Vektor n Dimensionen. Ein einfaches Beispiel mit einem sehr kleinen Vokabular ist in Abbildung 4.5 zu sehen. Das Wort „is“ tritt im Beispiel 2 mal auf, daher enthält der BOW-Vektor an der Stelle für das entsprechende Wort den Eintrag 2. Wörter, die im Vokabular, aber nicht im gegebenen Text vorhanden sind, erhalten den Eintrag 0. Das BOW-Modell ist also folgendermaßen definiert:

$$\text{bow}(t, d) = \#\text{Auftauchen von Term } t \text{ in Dokument } d \tag{4.4.2}$$

Ein BOW-Vektor enthält keine Strukturen des repräsentierten Textes. Die Reihenfolge der Wörter spielt im BOW-Vektor keine Rolle und ist daher nicht enthalten. Die beiden Sätze „*It is cold outside, but it is warm inside.*“ und „*It is warm outside, but it is cold inside.*“ werden also trotz gegensätzlicher Aussage im BOW-Modell identisch dargestellt.

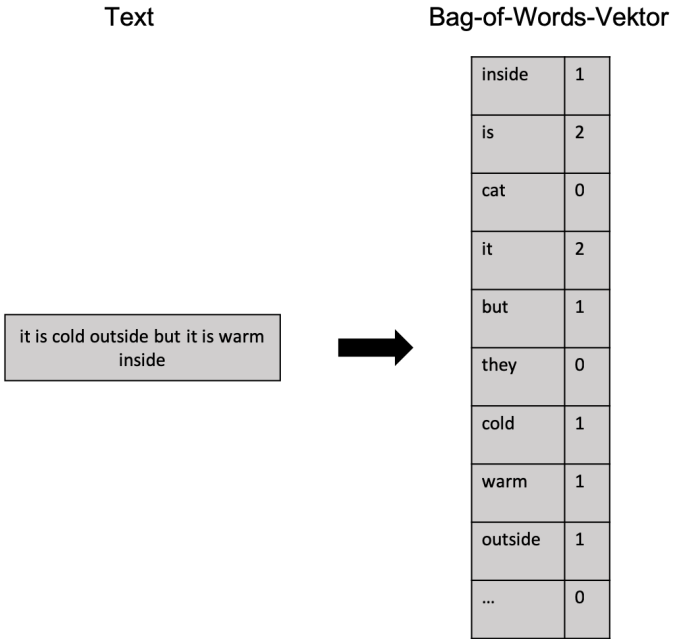


Abbildung 4.5: Grafische Darstellung des BOW-Modells.

In dieser Arbeit wird das vorgestellte BOW-Modell zu einem Bag-of-Words-and-Bigrams-Modell erweitert. Wie der Name bereits impliziert, werden nicht nur Vektoren aus einzelnen Wörtern (Unigrammen) verwendet, sondern zusätzlich Bigramme. Ein Bigramm setzt sich aus einer Sequenz aus zwei Worten zusammen, was bedeutet, dass die Reihenfolge der Wörter im Text eine Rolle spielt. Der Satz „*Ben played a computer game*“ erzeugt die folgenden Bigramme „Ben played“, „played a“, „a computer“ und „computer game“. Aus k unterschiedlichen Wörtern lassen sich k^2 verschiedene Bigramme bilden. Die zusätzliche Verwendung der Bigramme führt zu einem größeren Kostenaufwand bei der Verarbeitung, allerdings sind hier mehr Informationen als in den Unigrammen enthalten.

Tf-idf-Maß:
Tf-idf steht für „*term frequency-inverse document frequency*“ und ist eine Abwandlung des BOW-Modells mit dem Ziel, die charakteristischsten Terme eines Dokumentes zu identifizieren [ZC18, Kapitel 3 + 4][MRS09, Kapitel 6]. Im BOW-Modell werden alle Terme gleich behandelt und häufig auftretenden Termen wird dementsprechend ein hoher Wert zugewiesen. Das Problem an diesem Modell ist, dass die charakteristischsten Terme eines Dokumentes nicht die sind, die insgesamt (in allen Dokumenten) häufig auftreten, sondern die, die in einzelnen Dokumenten mehrfach auftreten und ansonsten nur selten vertreten

sind. In einem Star-Wars-Roman tritt beispielsweise mehrfach der Term „Jedi“ auf, der in anderen Romanen nicht vorkommt. Der Term ist also charakteristisch für Star-Wars-Romane. Zum Erreichen des Ziels, die Identifikation von charakteristischen Termen für bestimmte Dokumente, wird das BOW-Modell erweitert, bzw. normalisiert. Diese Normalisierung findet über die invertierte *Dokumenten-Frequenz* (*inverse document frequency*) statt, die folgendermaßen definiert ist:

$$\text{idf}(t) = log(\frac{N}{\#\text{Dokumente, in denen der Term } t \text{ auftaucht}}) \tag{4.4.3}$$

Wobei N die Anzahl aller Dokumente darstellt und der Logarithmus dafür verwendet wird, um einen Wertebereich zwischen 0 und 1 zu erhalten. Das Tf-idf-Maß wird dementsprechend folgendermaßen definiert:

$$\text{tf-idf}(t, d) = \text{bow}(t, d) * \text{idf}(t) \tag{4.4.4}$$

Wie zuvor im BOW-Modell wird in dieser Arbeit die Tf-idf-Gewichtung nicht nur auf Termen in Form von Unigrammen, sondern auch zusätzlich auf Bigrammen durchgeführt.

Word-Embeddings:
Bei *Word-Embeddings* handelt es sich im Allgemeinen um die Darstellung von Termen in Form von Wörtern und Phrasen als n-dimensionale numerische Vektoren. Ziel dieser Darstellung ist es, Ähnlichkeiten von Termen anhand der Ähnlichkeiten der Vektoren feststellen zu können. In dieser Arbeit werden vortrainierte Word-Embeddings verwendet, die auf dem Google-News-Dataset mit dem *word2vec*-Verfahren trainiert wurden [Goo13]. Diese haben sich bereits im Referenz-CNN-Modell von Kim [Kim14] bei der Klassifikation von Texten bewährt. Word2vec wurde von Mikolov et al. [MSC+13] eingeführt. Das Verfahren basiert auf dem *Skip-Gram-Modell* von Mikolov et al. [MCCD13], einem neuronalen Netzwerk, das aus einem *Input-Layer*, einem *Projection-Layer* und einem *Output-Layer* besteht. Das Modell soll für ein gegebenes Wort im Input-Layer, über die *Gewichtungsmatrix* im Projection-Layer, benachbarte Worte im Output-Layer vorhersagen. Dabei sollen die Word-Embeddings, die sich aus den Spalten der Gewichtungsmatrix ergeben, so trainiert werden, dass die durchschnittliche log-Wahrscheinlichkeit für benachbarte Worte in einem Korpus maximiert wird. Als Lernfunktion dient dabei eine hierarchische Softmax-Funktion. Die verwendeten Google-News-Embeddings umfassen insgesamt 3 Millionen unterschiedliche Wörter und Phrasen, deren Wort-Vektoren 300 Dimensionen aufweisen.

Topic-Modellbildung:
Eine weitere Methode zur Feature-Generierung in dieser Arbeit stellt das Topic-Modell *LDA*, das in Kapitel 2.2.5 beschrieben wird, dar. Als einziges *unsupervised* Verfahren der vorgestellten ML-Modelle wird es in dieser Arbeit nicht direkt zur Branchen-Klassifikation eingesetzt, sondern im Rahmen des Feature-Engineerings zur Extraktion neuer Features mit gleichzeitiger Reduktion der Feature-Dimensionalität. Dies bedeutet, dass aus den Texten hoher Dimensionalität, deren Worte die Features darstellen, eine kleinere Anzahl neuer Features generiert werden, die sich aus dem Text ableiten. Diese Features sind im LDA-Modell die *Themen*. Beim Training des Modells muss die Anzahl der *verborgenen Themen* ausgewählt werden, um die Verteilung der Terme in den Themen und die Verteilung der Themen innerhalb der Dokumente berechnen zu können. Für diese Arbeit wird die Themenanzahl 23 gewählt, da diese der Anzahl an Klassen bzw. Branchen des Klassifikationsproblems entspricht, die in den Trainings- und Testdaten enthalten sind (siehe Kapitel 6.2). Als Input für das Modell dienen als Korpus die gesamten extrahierten und vorverarbeiteten Daten. Die einzelnen Dokumente im Korpus werden von den zuvor erzeugten BOW-Vektoren dargestellt. Nachdem das Modell trainiert ist, wird für jedes Dokument eine Wahrscheinlichkeitsverteilung der vertretenen Themen bestimmt. Diese Wahrscheinlichkeitsverteilung wird als 23-dimensionaler Vektor aufbereitet, in dem jedes Thema vertreten ist und der jeweilige Eintrag die Wahrscheinlichkeit angibt, dass im Dokument das zugehörige Thema repräsentiert ist (siehe Abbildung 4.6). Der Vektor wird in dieser Arbeit alleine und in Kombination der bereits vorgestellten Vektoren als Feature-Vektor bei der Modellbildung eingesetzt.

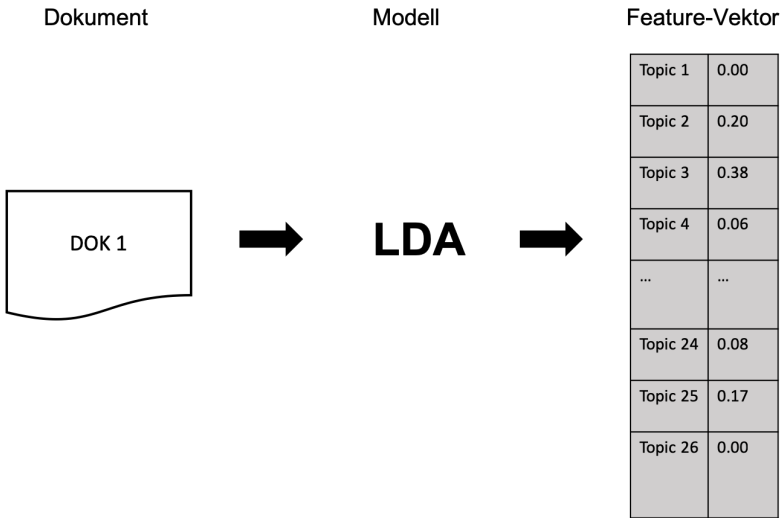


Abbildung 4.6: Grafische Darstellung der Feature-Generierung mit LDA.

4.4.5 Modell-Training

Zum Lösen des gegebenen Multi-Label Klassifikationsproblems werden in dieser Arbeit bei der Modellbildung zwei Ansätze verfolgt. Der erste Ansatz fällt unter den Begriff *Problem Transformation Method*, bei dem das Multi-Label Problem auf l unabhängige Klassifikationsprobleme transformiert wird, wobei l für die Anzahl der verschiedenen Label bzw. Klassen steht. Unter den Begriffen *Binary Relevance*, *One-vs-the-rest (OvR) classification* oder *One-vs-all approach* wird für jedes Label ein eigenes binäres Klassifikationsmodell erstellt [BLSB04], um die An- oder Abwesenheit der Klasse im gegebenen Dokument zu schätzen. Dabei wird die Annahme getroffen, dass die Label voneinander unabhängig sind. Zum Bestimmen des Multi-Label Targets werden alle l Modelle auf die zu klassifizierenden Dokumente angewendet. Mit den vorhergesagten Ergebnissen wird der Target-Vektor aufgebaut, der l Dimensionen aufweist und für jedes Label binär codiert angibt, ob dieses im Dokument an- oder abwesend ist (siehe Abbildung 4.7). Die Methode *Binary Relevance* wird in dieser Arbeit mit verschiedenen Klassifikationsmodellen angewendet. Als erstes Modell kommt *Naive Bayes* (siehe Kapitel 2.2.1) zum Einsatz, um zu sehen, wie ein einfaches Modell bezüglich der Unabhängigkeits-Annahmen im vorgestellten Klassifikationsproblem performt. Das zweite Modell, das eingesetzt wird, ist die logistische Regression (siehe Kapitel 2.2.2), um das lineare Regressionsmodell als Problemlöser zu testen. Mit dem letzten Modell im Zusammenspiel mit der Binary-Relevance-Methode wird ein diskriminativer Ansatz gewählt, in dem die Klassifikation durch das Separieren der Daten

anhand der Features stattfindet. Eines der beliebtesten Modelle, die diesen Ansatz verfolgen, sind die linearen SVMs (siehe Kapitel 2.2.3), die aus diesem Grund in dieser Arbeit verwendet werden.

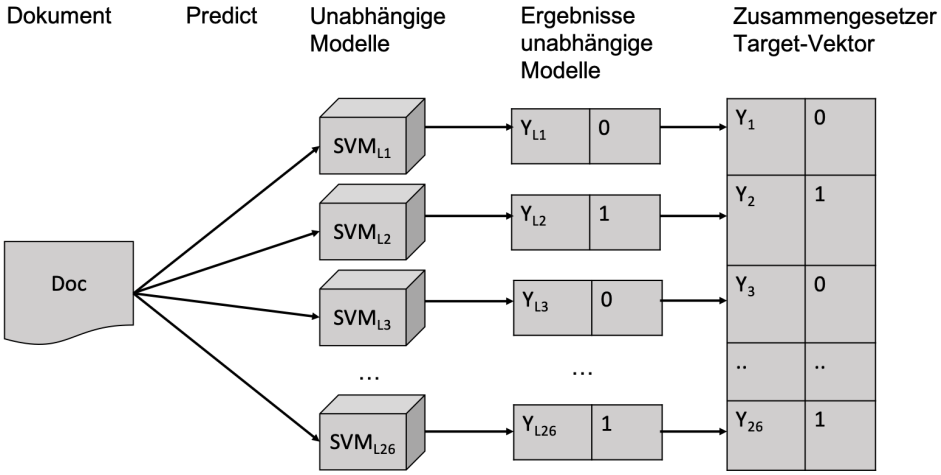


Abbildung 4.7: Grafische Darstellung der Klassifikation der Binary-Relevance-Methode.

Der zweite Ansatz in dieser Arbeit zum Lösen des Multi-Label Klassifikationsproblems ist die Anwendung eines Modells, das ohne Problemtransfomation eine solche Klassifikation durchführen kann. Ein Modell, auf das dies zutrifft, sind die Entscheidungsbäume (siehe Kapitel 2.2.4), die aus diesem Grund in dieser Arbeit verwendet werden. Damit im Gegensatz zum ersten Ansatz nicht nur ein einzelnes Modell vertreten ist, und außerdem die in der aktuellen Forschung verwendeten Neuronalen Netze berücksichtigt werden, wird als letztes Modell in dieser Arbeit das im Zusammenhang mit der Multi-Label Klassifikation häufig eingesetzte CNN-Modell (siehe Kapitel 2.2.6) gewählt. Dabei werden die gleichen Word-Embeddings und die meisten Parameter, wie im originalen Paper von Kim [Kim14], verwendet. Es werden jedoch drei entscheidende Änderungen vorgenommen, damit das Neuronale Netz mit dem Multi-Label Klassifikationsproblem umgehen kann.

1.) Änderung der Aktivierungsfunktion im *Fully-Connected-Layer*: Anstatt einer *Softmax*-Aktivierungsfunktion wird eine *Sigmoid* Aktivierungsfunktion verwendet. Grund dafür ist, dass die Wahrscheinlichkeiten der Label im Multi-Label Klassifikationsproblem unabhängig voneinander bestimmt werden sollen.

2.) Änderung der Verlustfunktion: Anstatt einer *categorical crossentropy*-Verlustfunktion wird eine *binary crossentropy*-Verlustfunktion eingesetzt, da das Target binär codiert ist, und jeder Eintrag des Targets unabhängig penaltisiert werden soll.

3.) Der Regularisierer wird herausgenommen, da diese zusätzliche Maßnahme gegen ein mögliches Overfitting bei ersten Versuchen zu einer extremen Verschlechterung der Ergebnisse geführt hat.

Alle Parameter des originalen CNN von Kim und des in dieser Arbeit eingesetzten CNNs sind im Anhang in Tabelle B.2 aufgeführt.

Die Modellbildung wird iterativ aufgebaut, indem zunächst ein Modell ohne weitere Anpassungen oder Parameter generiert wird und neue Modelle anhand der gewonnenen Erkenntnisse aus den Ergebnissen trainiert werden.

4.5 Evaluationsverfahren

Bei der Interpretation der Ergebnisse der Modelle auf den Testdaten gibt es verschiedene Ansätze. Der einfachste Ansatz, der in dieser Arbeit verwendet wird, ist es, jede Klasse c bzw. jedes Label λ aus dem geschätzten Target-Vektor Z einzeln zu betrachten. Dies wird erreicht, indem jedes binäre Klassifikationsmodell des Binary-Relevance-Ansatzes (siehe Abschnitt 4.4.5) einzeln bewertet wird. Die Bewertung erfolgt mit den folgenden Metriken:

- Accuracy: $ACC = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$
- Precision: $PRE = \frac{TP}{TP+FP}$
- Recall: $REC = \frac{TP}{FN+TP}$
- F1-Score: $F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE+REC}$

Die Ergebnisse der Mulit-Label-Klassifikation lassen sich auch zusammenhängend als solche betrachten. Dabei wird zwischen zwei Ansätzen unterschieden [TKV09][Sor10]. Der erste Ansatz ist die Label-basierte Evaluation, bei der die Ergebnisse aus dem Binary-Relevance-Ansatz zuerst einzeln bewertet werden, und im Anschluss der Durchschnitt über alle Label gebildet wird. Der zweite Ansatz ist die Beispiel-basierte Evaluation, bei der die Ergebnisse der Klassifikation ganzheitlich betrachtet werden, indem die Ähnlichkeit der geschätzten Target-Vektoren Z_i mit den tatsächlichen Target-Vektoren Y_i ermittelt und bewertet werden. Beide Ansätze werden in dieser Arbeit betrachtet. Daher werden sie zusammen mit deren Metriken im Folgenden ausführlicher vorgestellt.

Label-basierte Evaluation

Bei der Label-basierten Evaluation wird auf die gleichen Metriken, wie bei der binären Klassifikation (siehe oben), zurückgegriffen. Allerdings werden diese auf alle Label, anstatt auf einzelne bezogen. Dies kann über zwei verschiedene Durchschnittsberechnungen erreicht werden: dem *macro-averaging* und dem *micro-averaging* [Yan99]. Beim *macro-averaging* wird die jeweilige Metrik zuerst für jede Klasse bzw. jedes Label bestimmt und anschließend über alle Klassen gemittelt:

$$\lambda\text{-Precision, } P^{\lambda}_{\text{macro}}(Y,Z) = \frac{\sum_{i=1}^{N} Y_i^{\lambda} Z_i^{\lambda}}{\sum_{i=1}^{N} Z_i^{\lambda}} \qquad \text{Precision, } P_{macro} = \frac{1}{L}\sum_{i=1}^{L} P^{\lambda}_{\text{macro}}(Y,Z) \tag{4.5.1}$$

$$\lambda\text{-Recall, } R^{\lambda}_{\text{macro}}(Y,Z) = \frac{\sum_{i=1}^{N} Y_i^{\lambda} Z_i^{\lambda}}{\sum_{i=1}^{N} Y_i^{\lambda}} \qquad \text{Recall, } R_{macro} = \frac{1}{L}\sum_{i=1}^{L} R^{\lambda}_{\text{macro}}(Y,Z) \tag{4.5.2}$$

Im Gegensatz dazu wird beim *micro-averaging* die jeweilige Metrik global über alle Klassen und alle Testdaten gemittelt:

$$\text{Precision, } P_{micro} = \frac{\sum_{j=1}^{L}\sum_{i=1}^{N} Y_i^j Z_i^j}{\sum_{j=1}^{L}\sum_{i=1}^{N} Z_i^j} \tag{4.5.3}$$

$$\text{Recall, } R_{micro} = \frac{\sum_{j=1}^{L}\sum_{i=1}^{N} Y_i^j Z_i^j}{\sum_{j=1}^{L}\sum_{i=1}^{N} Y_i^j} \tag{4.5.4}$$

Beispiel-basierte Evaluation

Die konservativste Metrik der Beispiel-basierten Evaluation ist die *Classification Accuracy* [ZJXG05], auch *Exact Match Ratio* [Sor10] genannt. In dieser Metrik wird der vorhergesagte Target-Vektor nur dann als korrekt angesehen, wenn dieser vollständig mit dem tatsächlichen Target-Vektor übereinstimmt. Eine teilweise Übereinstimmung, durch die korrekte Klassifizierung einzelner Label, wird dabei ignoriert. Die Metrik wird per Indikatorfunktion $I()$ berechnet, die dann den Wert 1 liefert, wenn alle Label des geschätzten Target-Vektoren Z_i mit den tatsächlichen Target-Vektoren Y_i übereinstimmen:

$$\text{ClassificationAccuracy}(Y,Z) = \frac{1}{N}\sum_{i=0}^{N-1} I(Z_i = Y_i) \tag{4.5.5}$$

Zur Berücksichtigung der teilweisen Übereinstimmung werden die Metriken der binären Klassifikation auf die Multi-Label Klassifikation übertragen, indem die Übereinstimmung der gesetzten Label (1er in den Target-Vektoren) ermittelt und anschließen über alle Test-Daten gemittelt wird. Die daraus resultierenden Metriken sind folgendermaßen definiert [GS04]:

- Multi-Label-Accuracy bzw. Hamming Score: $\text{Accuracy}(Y, Z) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|Z_i \cap Y_i|}{|Z_i \cup Y_i|}$
- Multi-Label-Precision: $\text{Precision}(Y, Z) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|Z_i \cap Y_i|}{|Z_i|}$
- Multi-Label-Recall: $\text{Recall}(Y, Z) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|Z_i \cap Y_i|}{|Y_i|}$

Eine weitere Metrik der Beispiel-basierten Evaluation stellt der *Hamming Loss* dar [SS00]. Diese Metrik beschreibt die symmetrische Differenz zwischen dem Schätzer und dem wahren Target-Vektor, indem der Anteil der Label, die falsch vorhergesagt werden, bestimmt wird.

$$\text{Hamming Loss}(Y, \mathbb{Z}) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{\oplus(Z_i, Y_i)}{L}$$

5 Implementierung

In diesem Kapitel wird die Implementierung der Datenverarbeitungs- und Modellbildungspipeline aus dem vorherigen Kapitel beschrieben. Dabei wird zunächst in Abschnitt 5.1 ein Überblick über die verwendeten Technologien gegeben. Im Anschluss wird in Abschnitt 5.2 auf die Implementierung der Datenaufbereitung und in Abschnitt 5.3 auf die der Modellbildung eingegangen. Es wird dabei aufgezeigt, für welche Aufgabe welche Technologien zum Einsatz kommt und die Wahl der Parameter.

5.1 Verwendete Software

Als Programmiersprache wird in jedem Schritt der Pipeline *Python 3.6.6*[1] verwendet, allerdings wird für einzelne Aufgaben eine abweichende Version eingesetzt, da manche Pakete nur in bestimmten Python-Releases verwendet werden können. Im Falle einer abweichenden Python-Version, wird diese im weiteren Verlauf dieser Arbeit kenntlich gemacht.

Die Umsetzung findet auf einem Server des *Big Data Competence Centers* der Hochschule Darmstadt [2] statt. Es handelt sich hierbei um einen *Dell PowerEdge C6220* Server mit 4 Rechenkernen und 64 GB RAM. Als Betriebssystem läuft *Ubuntu 14.04.5 LTS* auf dem Server.

Die wichtigsten Technologien für die Aufbereitung der Daten und die Umsetzung des Klassifikationsproblems sind in Abbildung 5.1 zu sehen. Diese teilen sich auf in Java-Anwendungen und Python-Bibliotheken, die lokal auf einem Hochschul-Server verwendet werden. Die Python-Bibliotheken lassen sich aufteilen in solche, die für die Datenaufbereitung und -exploration verwendetet werden, und die Bibliotheken, die für die Modell-

[1] `https://www.python.org/downloads/release/python-366/`

[2] `https://fbi.h-da.de/forschung/arbeitsgruppen/big-data-competence-center/big-data-cluster/`

bildung zum Einsatz kommen. Dabei ist kenntlich gemacht, welche Python-Bibliothek mit welcher Technologie auf dem lokalen Server oder mit externen Diensten kommuniziert bzw. Daten austauscht. Einzelne Technologien werden in den folgenden Abschnitten genauer beschrieben.

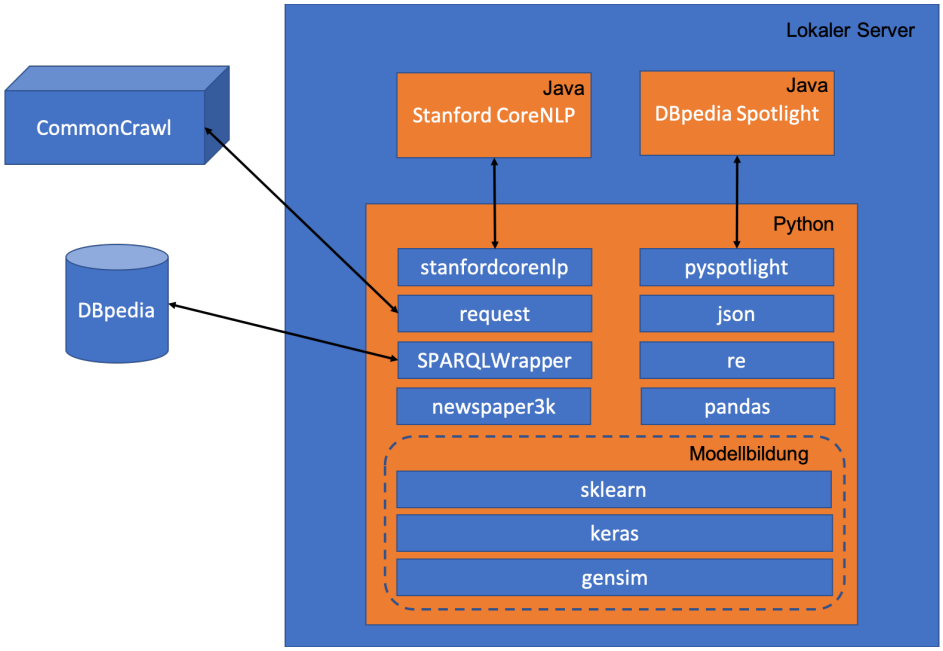


Abbildung 5.1: Übersicht der wichtigsten verwendeten Technologien und deren Zusammenspiel.

5.1.1 Stanford CoreNLP

Im weiteren Verlauf dieser Arbeit wird für die Aufgabe der NER das *Stanford CoreNLP* Toolkit eingesetzt [Sta19]. Es handelt sich hierbei um ein Open Source Java Annotations-Pipeline-Framework, das sowohl in der Forschung, als auch in der Industrie und in staatlichen Einrichtungen weit verbreitet ist. Der Zugriff auf *CoreNLP* kann per Command-Line oder per API erfolgen. Über verschiedene Wrapper können dabei neben Java viele weitere Programmiersprachen verwendet werden, wie beispielsweise Python, das im Rahmen dieser Arbeit eingesetzt wird. Eine weitere Einsatzmöglichkeit von CoreNLP ist die Verwendung in Szenarien auf verteilten Systemen, wie Apache Hadoop oder Apache Spark, wie es in Kapitel 3.1.3 der Fall ist. Der Nutzer kann mit dem Tool eine Pipeline aus verschiedenen Annotatoren aufbauen, um Texte aufzubereiten und so Informationen zu gewinnen.

Dabei sind folgende Annotatoren implementiert, die den großen Umfang des Tools widerspiegeln:

- **tokenize**: Teilt den Text in eine Sequenz von Token auf.
- **cleanxml**: Entfernt alle XML-Tags aus einem Dokument.
- **ssplit**: Teilt eine Token-Sequenz in Sätze auf.
- **truecase**: Identifiziert die wahre Groß- oder Kleinschreibung eines Tokens.
- **pos**: Fügt die Part-of-Speech Tags zu den Tokens hinzu.
- **lemma**: Generiert die Grundform der Tokens.
- **gender**: Fügt Gender-Informationen zu Namen hinzu.
- **ner**: Führt eine NER durch.
- **regexner**: Führt eine NER basierend auf regulären Ausdrücken durch.
- **parse**: Führt eine syntaktische Analyse aus.
- **sentiment**: Führt eine Sentimentanalyse durch.
- **decoref**: Implementiert eine Mention Detection, die Erwähnungen auf Entitäten bezieht.

Die NER, die im Annotator *ner* implementiert ist, verwendet für die Identifikation von Personen, Orten und Organisationen ein CRF-Modell. Das Modell [Sta18] nutzt eine Variante des Viterbi-Algorithmus [For73], um die verdeckten Variablen zu identifizieren, ähnlich, wie es bei der Modell-Variante „local+Viterbi“ von Finkel et al. [FGM05] der Fall ist. Als Anpassung werden Features eingesetzt, die auf einer Clustering-Methode mit der Bezeichnung *distributional similarity* basieren. Das Modell für die englische Sprache wurde auf verschiedenen Named-Entity-Korpora[3] trainiert, um es robust gegenüber verschiedenen Domänen zu machen.

[3] verwendete Korpora: CoNLL, MUC-6, MUC-7 und ACE

5.1.2 DBpedia Spotlight

In dieser Arbeit wird für das EL das Java Tool *DBpedia Spotlight* [Ins18b][Ins18a] verwendet. Es handelt sich dabei um ein Software-Projekt, das im Zusammenhang mit der Knowledge Base DBpedia (siehe Kapitel 3.2) entwickelt wurde, die in dieser Arbeit zum Einsatz kommt. Von *Spotlight* existieren aktuell zwei verschiedene Ausführungen. Die zuerst entwickelte Ausführung trägt den Namen *Lucene* [MJGsB11], während die neu entwickelte Ausführung unter dem Namen *Model* [DJHM13] läuft. Im weiteren Verlauf dieser Arbeit bezieht sich die Erwähnung und Verwendung von *Spotlight* auf die Entwicklung *Model*. Spotlight kann genau wie CoreNLP über Wrapper mit verschiedenen Programmiersprachen verwendetet und auch in verteilten Szenarien eingesetzt werden (siehe Kapitel 3.1.3).

Das EL findet bei Spotlight in zwei Stufen statt [DJHM13]. Die erste Stufe ist das *Phrase Spotting*, in dem sowohl das Spotting, als auch die Candidate Selection stattfindet. Im ersten Schritt des Phrase Spottings werden mögliche Kandidaten für das EL generiert. Dafür wird auf drei Ansätze zurückgegriffen: 1. Identifikation aller großgeschriebenen Wort-Sequenzen, 2. Identifikation aller Nominalsätze, Präpositionalsätze und Mehr-Wort-Einheiten, 3. Identifikation aller Named Entities. Für die beiden letzten Ansätze wird *Apache OpenNLP*[4] und dessen Modelle verwendet. Im zweiten Schritt werden die besten Kandidaten ermittelt, indem für jeden Kandidaten ein Score berechnet wird. Dieser Score ist eine Linearkombination aus verschiedenen Features. Eines dieser Features ist beispielsweise die Annotationswahrscheinlichkeit $P(\text{annotations}|s) = \sum_e \frac{\text{count}(e,s)}{\text{count}(s)}$ für eine DBpedia-Resource e (Entity) unter Verwendung des Ankertextes s (Spot). Die Informationen zum Ermitteln der Features stammen aus einem Wikipedia Datensatz, der aus Artikel-Links e mit deren verwendeten Ankertexten s und dem textuellen Kontext c besteht. Überschneidungen von Kandidaten und Kandidaten, die unter einen festgelegten Score fallen, werden entfernt. Der zweite Schritt ist der *Disambiguation*-Schritt. Dabei wird ein generatives Wahrscheinlichkeitsmodell verwendet, das einen Score auf der Basis einer Kombination der Wahrscheinlichkeiten $\text{P}(e)$, $\text{P}(s|e)$ und $\text{P}(e|e)$ berechnet. Bei der Ermittlung der Wahrscheinlichkeiten wird der Wikipedia Datensatz und ein Maximum Likelihood Ansatz verwendet. Die Scores werden im Anschluss per Softmax-Funktion normalisiert, damit sie zwischen 0.0 und 1.0 liegen. Weiterhin wird eine *NIL*-Entität eingeführt, um die Möglichkeit zu berechnen, dass die Entität nicht bekannt ist, also nicht in

[4] https://opennlp.apache.org/

DBpedia vorkommt. Alle Kandidaten, die einen kleineren Score haben als die NIL-Entität, werden verworfen. Die Entität mit dem höchsten Score wird als passender Kandidat für das Linking-Problem angesehen.

5.1.3 Gensim

Das LDA-Topic-Modeling wird mit der Python-Bibliothek *gensim* umgesetzt, die die Parameter der LDA mit einem „Online-variationalen-Inferenz-Algorithmus“ berechnet, eine Methode, die von Hoffman et al. [HBB10] beschrieben wurde. Der Algorithmus arbeitet auf dem geglätteten LDA-Modell mit der eingeführten variationalen Inferenz, wie es in Kapitel 2.2.5 beschrieben ist. Zum Approximieren der Parameter verwendet der Algorithmus im *Online LDA*-Modell wie im *Standard-LDA-Modell* den EM-Algorithmus. Die Besonderheit in der Online Variante ist, dass der Korpus in mehrere Teile, sogenannte *Mini-Batches*, aufgeteilt wird. Im *E-Schritt* werden die variationalen Parameter für die einzelnen *Mini-Batches* unabhängig voneinander berechnet, indem sie für jedes Dokument im jeweiligen *Mini-Batch* angepasst werden, bis sie konvergieren. Im *M-Schritt* werden gemittelte variationale Parameter aus den *Mini-Batches* berechnet, die anschließend dazu verwendet werden, die variationalen Parameter der Themenverteilung des gesamten Korpus anzupassen. Der Algorithmus hat den Vorteil, dass der E-Schritt parallel auf den *Mini-Batches* durchgeführt werden kann, und er schnell auf großen Datensätzen konvergiert.

5.2 Datenaufbereitung

Die Datenaufbereitung findet in der Verarbeitungspipeline an verschiedenen Stellen statt und benötigt je nach Verarbeitungsschritt verschiedene Methoden und Tools. Im Folgenden wird die technische Umsetzung der einzelnen Verarbeitungsschritte aufgezeigt.

News Extraktion

Die Extraktion der Nachrichtenartikel findet, wie in Kapitel 4.3 beschrieben, über die CC-Indexsuche statt. Dazu wird für die verwendeten Nachrichtenseiten jeweils ein URL-String der CC-Indexsuche aufgebaut und per Request der Python-Bibliothek *requests* abgefragt.

Folgende Parameter werden in die URL mit aufgenommen:

- *mime:text/html*: Das WARC-Record soll eine HTML-Datei enthalten.
- *status:200*: Der HTTP-Status-Code der gecrawlten Seite soll *200* sein, was bedeutet, dass auf die Seite ohne auftretende Fehler zugegriffen wurde.
- *output:json*: Die Response der Indexsuche soll im JSON-Format sein.

Als Response wird eine zeilenweise Auflistung mit allen Records im durchsuchten Monatscrawl zurückgegeben, die auf die Such-URL zutreffen. Ebenfalls per Request werden die einzelnen Records per HTTP-Zugriff (siehe Kapitel 3.1.2) von CC angefragt. Neben der Angabe des Dateinamen, in dem das gesuchte Record liegt, steckt in der Abfrage die Reichweite an Bytes, die heruntergeladen werden sollen. Die Response beinhaltet die Bytes, die den komprimierten Record darstellen. Per *BytesIO*-Objekt der Python-Bibliothek *io* werden die Records mit der Bibliothek *gzip* dekomprimiert. Die Extraktion des Nachrichtentextes aus den WARC-Records findet anschließend per Python-Bibliothek *newspaper3k* statt.

Zum Verringern der Laufzeit werden die Records der einzelnen Nachrichtenseiten parallel heruntergeladen. Dies wird mit der Bibliothek *concurrent* erreicht, indem jede Such-URL in einem einzelner Prozess verarbeitet wird.

Named Entity Recognition und Entity Linking
Die NER und das EL finden nacheinander im gleichen Prozessschritt statt.

Für die Named Entity Recognition wird in dieser Arbeit das Tool *Stanford CoreNLP 3.9.2* eingesetzt (siehe Abschnitt 5.1.1). Damit das Tool ausreichende Ressourcen zur Verfügung hat, werden beim Starten des Tools per Java-Parameter *mx4g* 4GB RAM allokiert.

Der Zugriff auf den lokalen CoreNLP-Server findet per Python-Bibliothek *stanfordcorenlp* statt. Beim Annotieren der Texte mit Stanford CoreNLP werden folgende Parameter gesetzt:

- *annotators: ner*: Es soll eine NER mit allen dazu benötigten Schritten durchgeführt werden.
- *outputFormat : json*: Ergebnisse sollen im JSON-Format zurückgegeben werden.

- *ner.applyFineGrained: false*: Es soll keine feinere Granularität der Entitäts-Klassen vorgenommen werden, wie beispielsweise die Klassifikation einer *Location*-Entität als *City*.
- *ner.model: edu/stanford/nlp/models/ner/english.all.3class.distsim.crf.ser.gz*: Es soll das Modell verwendet werden, das Entitäten in die drei Klassen *LOCATION*, *ORGANIZATION* und *PERSON* einteilt.
- *ner.applyNumericClassifiers: false*: Es soll keine Klassifikation von Zahlensequenzen vorgenommen werden.
- *ner.useSUTime: false*: Es soll keine Klassifikation von Datumsangaben vorgenommen werden.

Durch das Setzen der Parameter werden nur die für das gegebene Szenario relevanten Schritte ausgeführt und Entitäten gesucht. Somit wird die Verarbeitungsgeschwindigkeit erhöht.

Das EL wird im Szenario dieser Arbeit mit dem Tool *DBpedia Spotlight 1.0.0* (siehe Kapitel 5.1.2) ausgeführt. Als Sprach-Modell wird die aktuellste Version des englischen Modells mit der Bezeichnung *2016-10* verwendet. Der Zugriff auf den lokalen Spotlight-Server erfolgt per Python-Bibliothek *pyspotlight 0.7.2*. Aus Kompatibilitätsgründen wird *Python 3.5* im Zusammenhang mit *pyspotlight* verwendet. Beim Annotieren der Texte mit DBpedia Spotlight werden folgende Parameter gesetzt:

- *policy: whitelist*: Gibt alle Entitäten zurück, die den selben Typ aufweisen, wie mit dem Parameter *types* angegeben (siehe nächster Punkt).
- *types: DBpedia:Company*: Es sollen nur die Entitäten zurückgegeben werden, die als *dbo:Company* in DBpedia aufgeführt sind.
- *coreferenceResolution: false*: Parameter, der angibt, ob der zu annotierende Text auf Koreferenzen untersucht werden soll, um daraus die verschiedenen Surface-Formen abzuleiten. Falls auf *true* gesetzt, könnten keine anderen Filter angewendet werden.
- *confidence: 0.5/0.8*: Gibt die Konfidenz des Disambiguation-Schritts an. Der Parameter nimmt Werte zwischen 0.0 und 1.0 an. Je höher der Parameter gesetzt wird, desto weniger Fehler sollen eingegangen werden. Dafür wird allerdings der Verlust von korrekten Zuordnungen in Kauf genommen. Es werden nur Entitäten gelinkt, die

im Disambiguation-Schritt einen Mehrdeutigkeits-Score $< 1 - \textit{confidence}$ aufweisen. Der Mehrdeutigkeits-Score gibt die Ähnlichkeit zwischen den zwei wahrscheinlichsten Linking-Kandidaten an und ist umso größer, je ähnlicher sich die Kandidaten sind [MJGsB11].

- *support: 0*: Gibt an, wie häufig die Entität in Wikipedia vertreten sein muss, damit sie als Linking-Entität in Frage kommt. Gemeint ist dabei die Anzahl an Verlinkungen in Wikipedia von Wikipedia-Artikeln anderer Entitäten zu der betrachteten Entität [MJGsB11].

Durch die angegebenen Parameter werden nur Firmen aus DBpedia gelinkt. Dabei soll es keine Rolle spielen, wie prominent diese Firmen sind, also wie oft sie in Wikipedia vertreten sind. Durch das Variieren der Konfidenz werden zwei verschiedene Datensätze erzeugt. Der erste Datensatz ist ein liberaler Datensatz, in dem mehr Entitäten gelinkt, dabei aber mehr Fehler eingegangen werden. Der zweite Datensatz ist ein konservativer Datensatz, in dem weniger Fehler eingegangen, dafür aber weniger Entitäten gelinkt werden.

Zum Verringern der Laufzeit werden die Daten ebenfalls parallel pro Nachrichtenseite in einem separaten Prozess verarbeitet, in dem nacheinander die NER und das EL durchgeführt werden, und die Ergebnisse mit Hilfe der Python-Bibliothek *json* im JSON-Format zusammengetragen werden.

Datensatz Erstellung

Das Klassen-Mapping bei der Erstellung des Datensatzes (siehe Kapitel 4.4.1) findet über SPARQL-Abfragen an den DBpedia SPARQL Web Service (Kapitel 3.2.2) statt. Per Python-Bibliothek *SPARQLWrapper* werden die Branchen der zuvor gelinkten Entitäten abgefragt und dann per Mapping-Tabelle, die mit der Bibliothek *csv* importiert wird, normalisiert. Die Ergebnisse werden wie zuvor mit der Python-Bibliothek *json* zusammengetragen und aufbereitet.

5.3 Modellbildung

Für die Modellbildung und die Visualisierung der Daten wird die Entwicklungsumgebung *Jupyter Notebook*[5] verwendet.

Zur weiteren Verarbeitung werden die Daten aus den erstellten JSON-Dateien in DataFrames der Python-Bibliothek *pandas* geladen. Die DataFrames dienen als einheitliche Schnittstelle zu anderen Python-Bibliotheken zur Datenverarbeitung und Auswertung.

Die Umsetzung der Modellbildung orientiert sich an den Prozessschritten aus Kapitel 4.4.

Exploration
Aufgrund der verschiedenen Datensätze, die während der Verarbeitungspipeline anfallen, wird ein Teil der Datenexploration auf einer *MongoDB*-Datenbank ausgeführt. Dazu werden die JSON-Dateien in die Datenbank importiert und manuell abgefragt. Folgende Abfragen werden auf *MongoDB* ausgeführt:

- Anzahl der gespotteten Entitäten (NER)
- Anzahl der gelinkten Entitäten (EL)
- Schnittmenge zwischen gespotteten und gelinkten Entitäten
- Anzahl der Entitäten, die nur gespottet werden
- Anzahl der Entitäten, die nur gelinkt werden
- Anzahl der unterschiedlichen gelinkten Entitäten (distinct)

Weitere Statistiken werden durch die Aggregation der Daten in den DataFrames berechnet. Zu diesen gehören die Verteilung der Branchen und die Verteilung der Anzahl der Firmen pro Dokument. Visualisiert werden die Statistiken mit der Bibliothek *matplotlib*. Weiterhin wird die paarweise Korrelation der Branchen mit der Funktion *matthews_corrcoef* der Klasse *sklearn.metrics* berechnet, die mit dem Phi-Koeffizienten übereinstimmt. Visualisiert werden die Korrelationen als Korrelationstabelle mit einer Heatmap der Bibliothek *seaborn*.

[5] `https://jupyter.org/`

Vorverarbeitung

Für die Vorverarbeitung der Daten werden sowohl vordefinierte als auch selbst implementierte Funktionen verwendet. Das Entfernen der leeren Daten wird mit den pandas-DataFrame-Funktionen *loc* und *iloc* umgesetzt, wie in Listing 5.1 zu sehen ist.

```
data = data_raw.loc[(data_raw.iloc[:,:26].sum(axis=1) != 0),:]
null_columns = [col for col in data_pre_preprocessed.columns[:26] \
                    if data_raw.loc[:,col].sum() == 0]
data = data[data_pre_preprocessed.columns.difference(null_columns)]
```

Listing 5.1: Entfernen von leereen Daten

Die Kleinschreibung wird durch die String-Funktion *lower()* erzielt. Selbst implementierte Funktionen werden bei dem Entfernen von Satzzeichen, Zahlen und Stoppwörtern verwendet (siehe Listing 5.2). In den ersten beiden Fällen werden reguläre Ausdrücke unter Verwendung der Bibliothek *re* eingesetzt, während bei der Entfernung der Stoppwörter eine Liste der betroffenen Wörter mit der Natural-Language-Processing-Bibliothek *nltk* geladen und mit den einzelnen Wörtern in den Daten abgeglichen wird.

```
def removePunctuation(text):
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',text)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
    cleaned = cleaned.strip()
    cleaned = cleaned.replace("\n"," ")
    return cleaned

def removeNumbers(text):
    cleaned = ""
    for word in text.split():
        cleaned_word = re.sub('[^a-z A-Z]+', ' ', word)
        cleaned += cleaned_word
        cleaned += " "
    cleaned = cleaned.strip()
    return cleaned

def removeStopwords(text):
    cleaned = ""
    for word in text.split():
        if word not in stop_words:
            cleaned += word
            cleaned += " "
    cleaned = cleaned.strip()
    return cleaned
```

Listing 5.2: Funktionen zur Vorverarbeitung der Text-Daten

Das Aufteilen in Trainings- und Testdaten wird mit der Funktion *train_test_split* der *sklearn*-Klasse *model_selection* durchgeführt, wie es in Listing 5.3 zu sehen ist. Der Anteil der Testdaten beträgt 20%. Mit dem Parameter *shuffle=True* wird angegeben, dass die Daten vor der Aufteilung zufällig gemischt werden und *random_state=42* setzt dabei einen Seed, damit die Ergebnisse reproduziert werden können.

```
1 train, test = train_test_split(data,
2                                random_state=42,
3                                test_size=0.20,
4                                shuffle=True)
```

Listing 5.3: Aufteilung in Trainings- und Testdaten

Feature Extraction

Bei der Erstellung der BOW- und Tf-idf-Vektoren werden die Funktionen der Python-Bibliothek *sklearn* eingesetzt. Für die Erstellung der Tf-idf-Vektoren wird hierfür die Klasse *TfidfVectorizer* und für die BOW-Vektoren die Klasse *CountVectorizer* verwendet. In beiden Fällen wird der Parameter *ngram_range* auf Uni- und Bigramme festgelegt.

Die vortrainierten Word Embeddings des Google-News-Datensatzes (siehe Kapitel 4.4.4) werden mit der Python-Bibliothek *gensim* geladen. Anschließend wird aus den vorverarbeiteten Trainings- und Testdaten ein Index erstellt, der alle in den Daten auftretende Wörter enthält. Mit dem Index wird eine Embedding-Matrix erstellt, in der die Zeilen die Worte aus dem Wort-Index widerspiegeln und die Spalten die zu den Wörtern passenden 300 Dimensionen aus den Google-News-Word-Embeddings. Ist ein Wort aus dem Wort-Index nicht in den vortrainierten Word Embeddings enthalten, so werden alle Dimensionen für dieses Wort auf den Wert 0 gesetzt. Die Texte der Trainings- und Testdaten werden für das Training des CNNs mit der Funktion *pad_sequences* aus der Python-Bibliothek *keras* in gleich lange Wort-Sequenzen umgewandelt.

Für das Topic-Modelling mit *gensim* werden zunächst die Daten, wie zuvor beschrieben, vorverarbeitet. Anschließend wird die Funktion *word_tokenize()* der *nltk*-Bibliothek dazu verwendet, um alle extrahierten Nachrichtentexte in Token, in diesem Fall Unigramme, zu zerlegen. Die Token werden dazu verwendet, ein Dictionary mit der gleichnamigen Gensim-Klasse zu erstellen. Das Dictionary wiederum wird dazu verwendet, den Korpus der Nachrichtenartikel per *doc2bow*-Gensim-Funktion als Korpus aus BOW-Vektoren darzustellen. Die beiden erstellten Datenstrukturen dienen als Input für das LDA-Modell, dessen Erstellung mit den verwendeten Parametern in Listing 5.4 zu sehen ist. Zur Erstel-

lung des Modells wird die Klasse *LdaMulticore* eingesetzt, da diese parallelisiert arbeitet und somit die Laufzeit verringert. Die Anzahl der Topics wird auf 23 gesetzt, da in den Trainings- und Testdaten nur 23 Branchen vertreten sind, wie in Kapitel 6.2 zu sehen ist. Das Modell wird auf die BOW-Vektoren der vorverarbeiteten Texte aus den Trainings- und Testdaten angewandt, um die LDA-Features zu generieren.

```
1 ''' Train LDA model.'''
2 from gensim.models.ldamulticore import LdaMulticore
3
4 ''' Set training parameters.'''
5 num_topics = 23
6
7 model = LdaMulticore(corpus=corpus, id2word=dictionary, \
8                      num_topics=num_topics)
```

Listing 5.4: Erstellung des LDA-Modells

Modelltraining

Die Modellbildung wird mit der Ausnahme des CNNs mit der Python-Bibliothek *scikit-learn* umgesetzt. Dabei werden für die Modelle die Klassen *MultinomialNB* (Naive Bayes), *LogisticRegression*, *SVC* (Classification Support Vector Machine) und *tree* verwendet. Die Umsetzung als Multi-Label Klassifikation mit dem Ansatz Binary Relevance (siehe Kapitel 4.4.5) wird mit der Klasse *OneVsRestClassifier* erreicht. Ein Beispiel für die Umsetzung des Modells Support Vector Machine als Multi-Label Klassifikator ist in Listing 5.5 zu sehen.

```
1 from sklearn.svm import LinearSVC
2 from sklearn.multiclass import OneVsRestClassifier
3
4 ''' Run classifier '''
5 classifier_svc = OneVsRestClassifier(LinearSVC())
6 classifier_svc.fit(x_train, y_train)
```

Listing 5.5: Erstellung SVM als Multi-Label Klassifikator in der Python-Bibliothek *scikit-learn*

Die Umsetzung des CNNs findet mit der Bibliothek *keras* statt, wie in Listing 5.6 zu sehen ist.

```
''' model parameters '''
num_filters = 100
embed_dim = 300

''' training params '''
batch_size = 50
num_epochs = 25

''' CNN architecture '''
model = Sequential()
model.add(Embedding(nb_words, embed_dim,
          weights=[embedding_matrix], input_length=max_seq_len))
model.add(Conv1D(num_filters, 3, activation='relu',
         border_mode='same'))
model.add(Dropout(0.5))
model.add(MaxPooling1D(2))
model.add(Flatten())
model.add(Dense(num_classes, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
         optimizer=optimizers.Adam(lr=0.001),
         metrics=[categorical_accuracy])

''' model training '''
hist = model.fit(word_seq_train, y_train,
         batch_size=batch_size, epochs=num_epochs,
         validation_data=(word_seq_test, y_test), verbose=2)
```

Listing 5.6: Umsetzung CNN in der Bibliothek *keras*

Auswertung

Für die Auswertung der Modelle werden ebenfalls Funktionen der Bibliothek *scikit-learn* eingesetzt, die zu der Klasse *sklearn.metrics* gehören. Je nach Input und Wahl des Parameters *average* werden für die Funktionen *accuracy_score*, *precision_score* und *recall_score* verschiedene Metriken berechnet. Eine Übersicht hierzu ist in Tabelle 5.1 zu sehen.

Funktion	Input	average	Metrik
accuracy_score	binär	-	Accuracy
	multi-label	-	Exact Match Ratio
precision_score/ recall_score	binär	binary	Precision/ Recall
	multi-label	samples	Mulit-Label-Precision/ Mulit-Label-Recall
	multi-label	micro	Micro-Precision/ Micro-Recall
	multi-label	macro	Macro-Precision/ Macro-Recall
	multi-label	weighted	Macro-Precision/ Macro-Recall gewichtet nach Klassenverteilung

Tabelle 5.1: Verwendete Metriken aus *scikit-learn* nach Input und Parameter *average*

Die Multi-Label-Accuracy bzw. Hamming Score ist nicht in einer *scikit-learn*-Funktion umgesetzt. Hierfür wird die Implementation aus Listing 5.7 verwendet. Für die Hamming-Loss-Metrik wird die Funktion *hamming_loss* der Klasse *sklearn.metrics* genutzt.

```
def hamming_score(y_true, y_pred, normalize=True, sample_weight=None):
    '''
    Compute the Hamming score (a.k.a. label-based accuracy) for
    the multi-label case http://stackoverflow.com/q/32239577/395857
    '''
    acc_list = []
    for i in range(y_true.shape[0]):
        set_true = set( np.where(y_true[i])[0] )
        set_pred = set( np.where(y_pred[i])[0] )
        tmp_a = None
        if len(set_true) == 0 and len(set_pred) == 0:
            tmp_a = 1
        else:
            tmp_a = len(set_true.intersection(set_pred))/\
                    float( len(set_true.union(set_pred)) )
        acc_list.append(tmp_a)
    return np.mean(acc_list)
```

Listing 5.7: Implementation Hamming Score[6]

[6] https://stackoverflow.com/questions/32239577/getting-the-accuracy-for-multi-label-\prediction-in-scikit-learn

6 Auswertung

In diesem Kapitel folgt die Auswertung der Daten und Modelle aus der Modellbildungspipeline. Dazu wird in Abschnitt 6.1 auf die extrahierten Daten aus der Datenaufbereitung eingegangen. Anschließend werden in Abschnitt 6.2 die erstellten Modelle unter Verwendung verschiedener Parameter und Variationen untersucht.

6.1 Exploration der extrahierten Daten

Bei der Extraktion der Nachrichtenartikel aus Common Crawl werden in dieser Arbeit 125.048 Records heruntergeladen. Die genaue Verteilung der Records auf die verschiedenen Nachrichtenseiten ist im Anhang in Tabelle B.1 zu sehen. In der Datenaufbereitung (siehe Kapitel 4.3) wird eine NER, gefolgt von einem EL, durchgeführt, wobei beim EL mit dem Parameter *confidence* (siehe Kapitel 5.2) variiert, sodass zwei unterschiedliche Extraktionen entstehen. Die Ergebnisse der Schicht sind in Tabelle 6.1 zusammengefasst. Abweichende Zahlen in der Anzahl der Records sind auf fehlerhafte Datensätze oder auftretende Fehler bei der Aufbereitung zurückzuführen. Zu erkennen ist, dass mehr als doppelt so viele Entitäten bei der NER gefunden werden, als beim EL gelinkt. Bei den Daten auf höherer Konfidenz wird der Unterschied noch einmal größer. Grund dafür ist, dass bei der NER viele Firmen entdeckt werden, die beim EL nicht gelinkt werden können, da sie nicht in DBpedia erfasst sind. Bei den Daten auf höherer Konfindenz werden weniger Entitäten gelinkt, um Fehler zu vermeiden. Auch die niedrige Anzahl unterschiedlicher Entitäten im EL im Gegensatz zur NER unterstützt diese Aussagen. Die Schnittmenge zwischen NER und EL ist geringer, als die Anzahl der gelinkten Entitäten. Dies kann sowohl auf Fehlklassifikationen als auch auf nicht entdeckte Entitäten in beiden verwendeten Tools zurückzuführen sein.

Kennzahl	***confidence 0.5***	***confidence 0.8***
Anzahl Records	125.001	125.028
Anzahl Entitäten NER	1.313.300	1.313.300
Anzahl Entitäten EL	555.191	442.913
Schnittmenge NER und EL	323.083	277.967
Anzahl unterschiedliche Entitäten NER	203.910	203.910
Anzahl unterschiedliche Entitäten EL	13.093	10.700

Tabelle 6.1: Ergebnisse der Datenaufbereitung für die beiden unterschiedlichen Konfidenz-Level.

Bei der Erstellung des Datensatzes für die Modellbildung werden die Daten, wie in den Kapiteln 4.3 und 5.2 beschrieben, aufbereitet. Dabei entstehen zwei Extraktionen pro Konfidenzlevel, aus denen sich jeweils zwei Datensätze ergeben, in denen die Daten dokumentenweise oder paragraphenweise dargestellt sind. Die Datensätze enthalten nach der Aufbereitung leere Dokumente und Paragraphen oder solche, die keine Entitäten enthalten. Diese werden im Zuge der Vorverarbeitung entfernt, wie in Kapitel 5.3 in Listing 5.1 zu sehen ist. In Tabelle 6.2 kann die Größe der einzelnen Datensätze vor und nach der Entfernung der leeren Daten und Entity-abwesenden Datensätze entnommen werden. Außerdem wird hier die durchschnittliche Anzahl der Wörter (Median) in den Texten nach der Bereinigung aufgelistet. Die Anzahl der Tupel erhöht sich bei der Aufbereitung in Paragraphen um den Faktor 3 bzw. um den Faktor $2,6$ (hohe Konfidenz), während sich die Anzahl der Wörter pro Tupel um den Faktor 14 verringert.

Datensatz	**Anzahl Tupel (vor Entfernung)**	**Anzahl Tupel (nach Entfernung)**	**Median Anzahl Wörter (nach Entfernung)**
Dokumentenweise (confidence = 0.5)	125.001	84.075	612
Dokumentenweise (confidence = 0.8)	125.026	76.566	611
Paragraphenweise (confidence = 0.5)	376.358	335.432	44
Paragraphenweise (confidence = 0.8)	326.178	277.716	44

Tabelle 6.2: Anzahl der Tupel vor der Modellbildung; vor und nach der Entfernung von leeren Tupeln und Tupeln ohne Entitäten.

Die Verteilung der unterschiedlichen Entitäten pro Dokument und pro Paragraph in den bereinigten Datensätzen sind in den Abbildungen 6.1 und 6.2 zu sehen. Dabei ist zu erkennen, dass die Dokumente und Paragraphen mit nur einer Entität deutlich dominieren. Vereinzelte Paragraphen können bis zu 31 verschiedene Entitäten beinhalten. Bei den Dokumenten enthalten einzelne sogar über 50 verschiedene Entitäten.

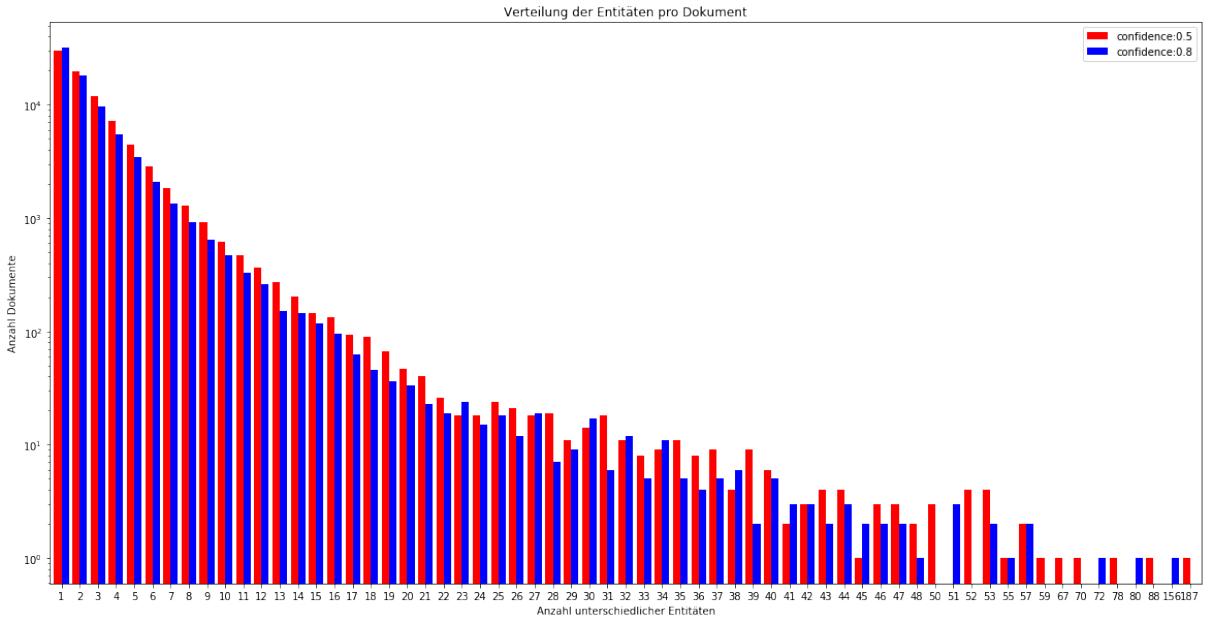


Abbildung 6.1: Verteilung der unterschiedlichen Entitäten pro Dokument; Die y-Achse ist logarithmiert skaliert.

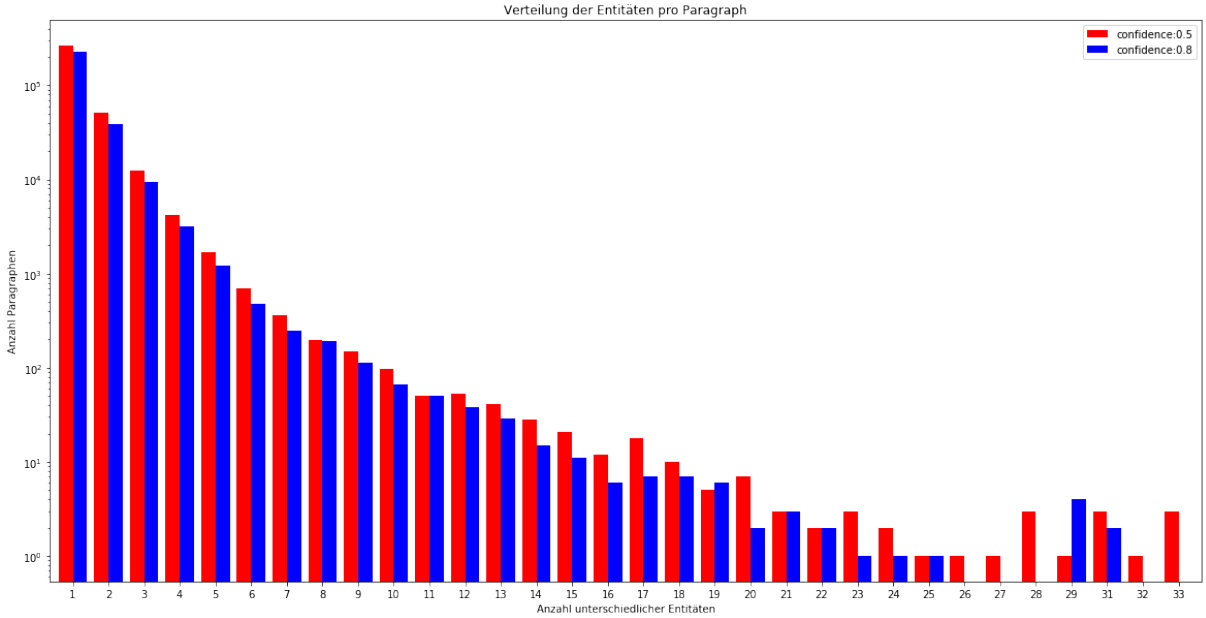


Abbildung 6.2: Verteilung der unterschiedlichen Entitäten pro Paragraph; Die y-Achse ist logarithmiert skaliert.

Resultierend aus den Verteilungen der Entitäten pro Dokument/Paragraph und der Verteilung der Branchen pro Entität (siehe Kapitel 3.2.3, Abbildung 3.4) ergeben sich die Verteilungen der Branchen pro Dokument/Paragraph, die in den Abbildungen 6.3 und 6.4 zu sehen sind. Während in den Dokumenten-Daten ca. 40% der Tupel zwei oder mehr Branchen aufweisen, wird ca. 84% der Tupel der Paragraph-Daten nur eine einzelne Entität zugeordnet.

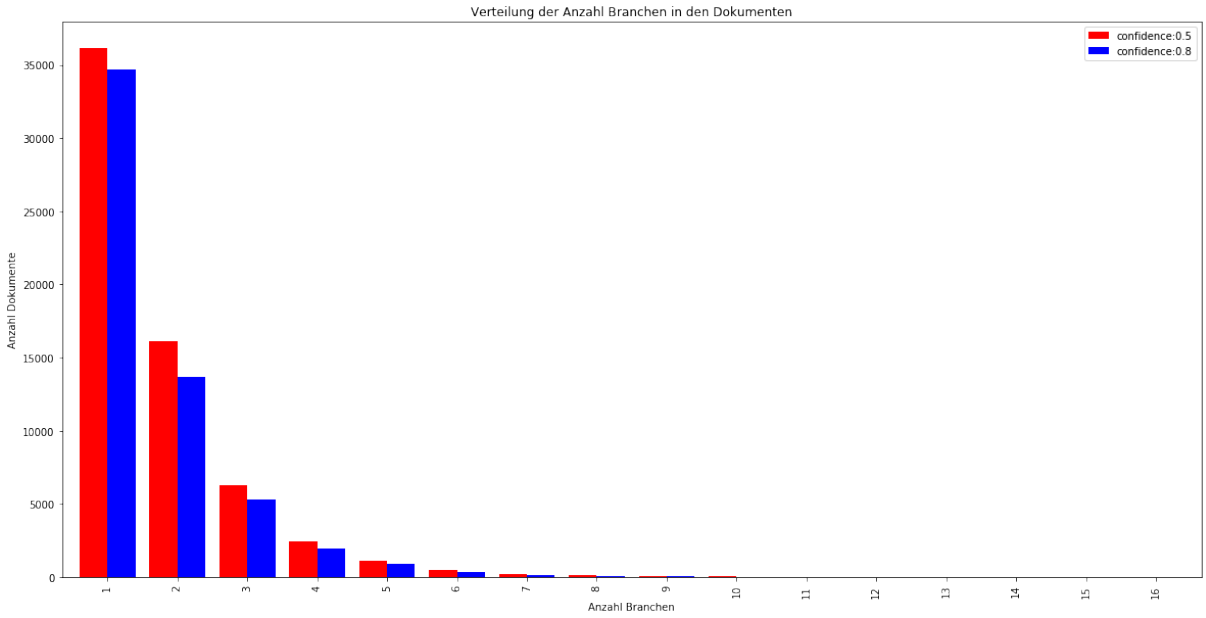


Abbildung 6.3: Häufigkeitsverteilung der Branchen in den Dokumenten.

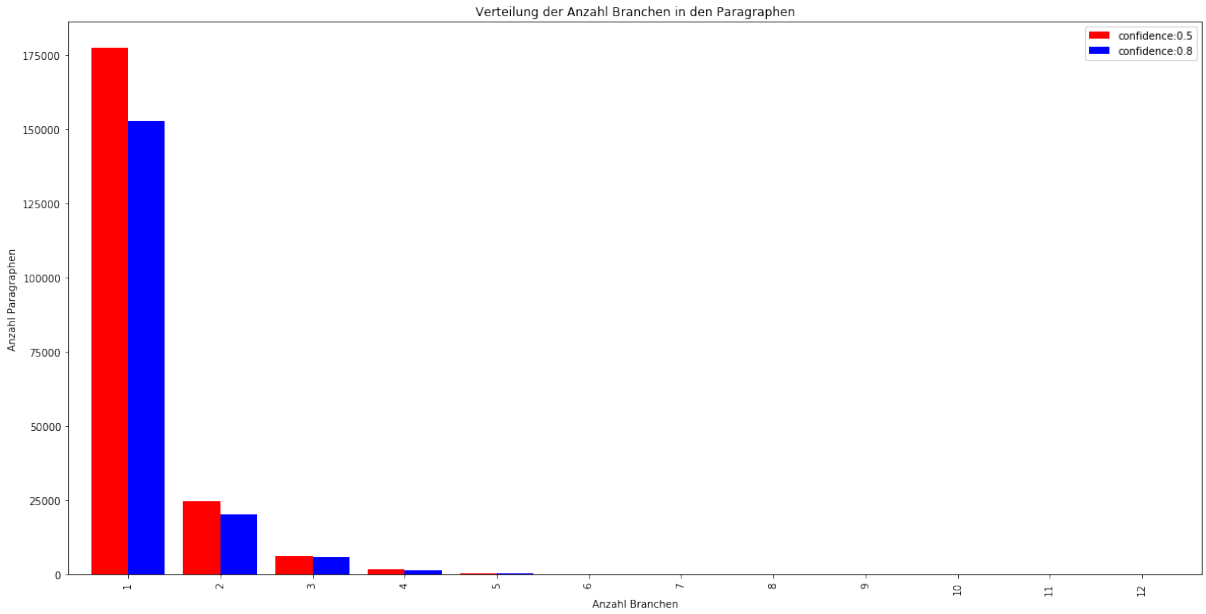


Abbildung 6.4: Häufigkeitsverteilung der Branchen in den Paragraphen.

Die möglichen Korrelationen zwischen den Branchen untereinander, wie sie mit den Heatmaps auf den Daten der niedrigen Konfidenz (Konfidenz-Parameter beim EL: 0,5) in Abbildung 6.5 und 6.6 dargestellt sind, geben keine Hinweise auf starke Korrelationen ($0{,}5 \leq$ Phi-Koeffizient). Leichte Korrelationen ($0{,}2 \leq$ Phi-Koeffizient $\leq 0{,}5$) werden zwischen den Branchen *Transportation and Logistics* und *Aerospace and Defense* festgestellt sowie in den Dokumenten-Daten zwischen den beiden Branchen *Pharmaceuticals and Biotech* und *Chemicals*. Die beobachteten Werte geben einen Hinweis darauf, dass die Abgrenzung der Branchen, wie sie durch das Mapping erreicht werden soll (siehe Kapitel 4.4.1), erfolgreich ist.

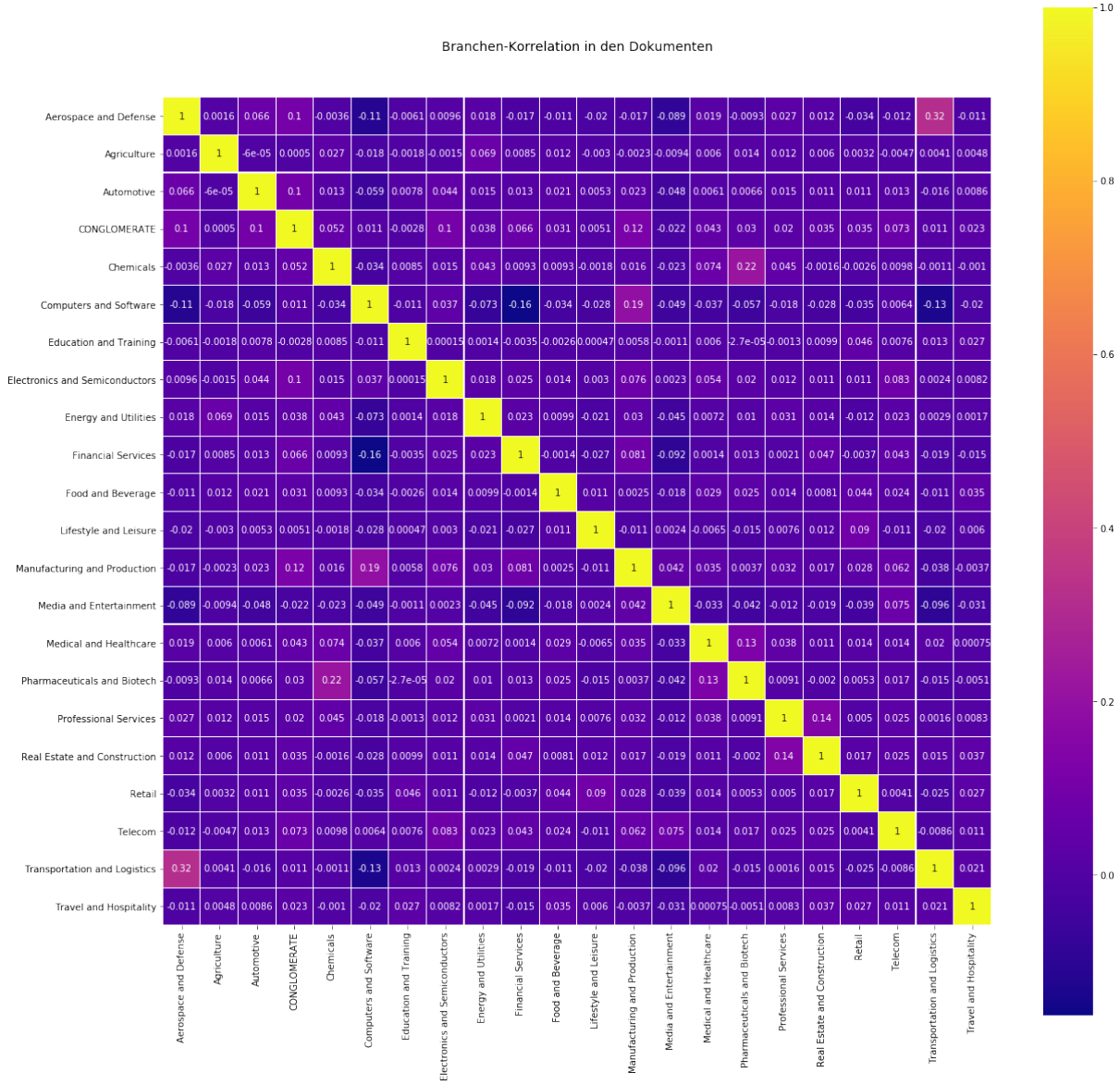


Abbildung 6.5: Korrelation der Branchen in den Dokumenten (niedrige Konfidenz), ermittelt durch den Phi-Koeffizienten.

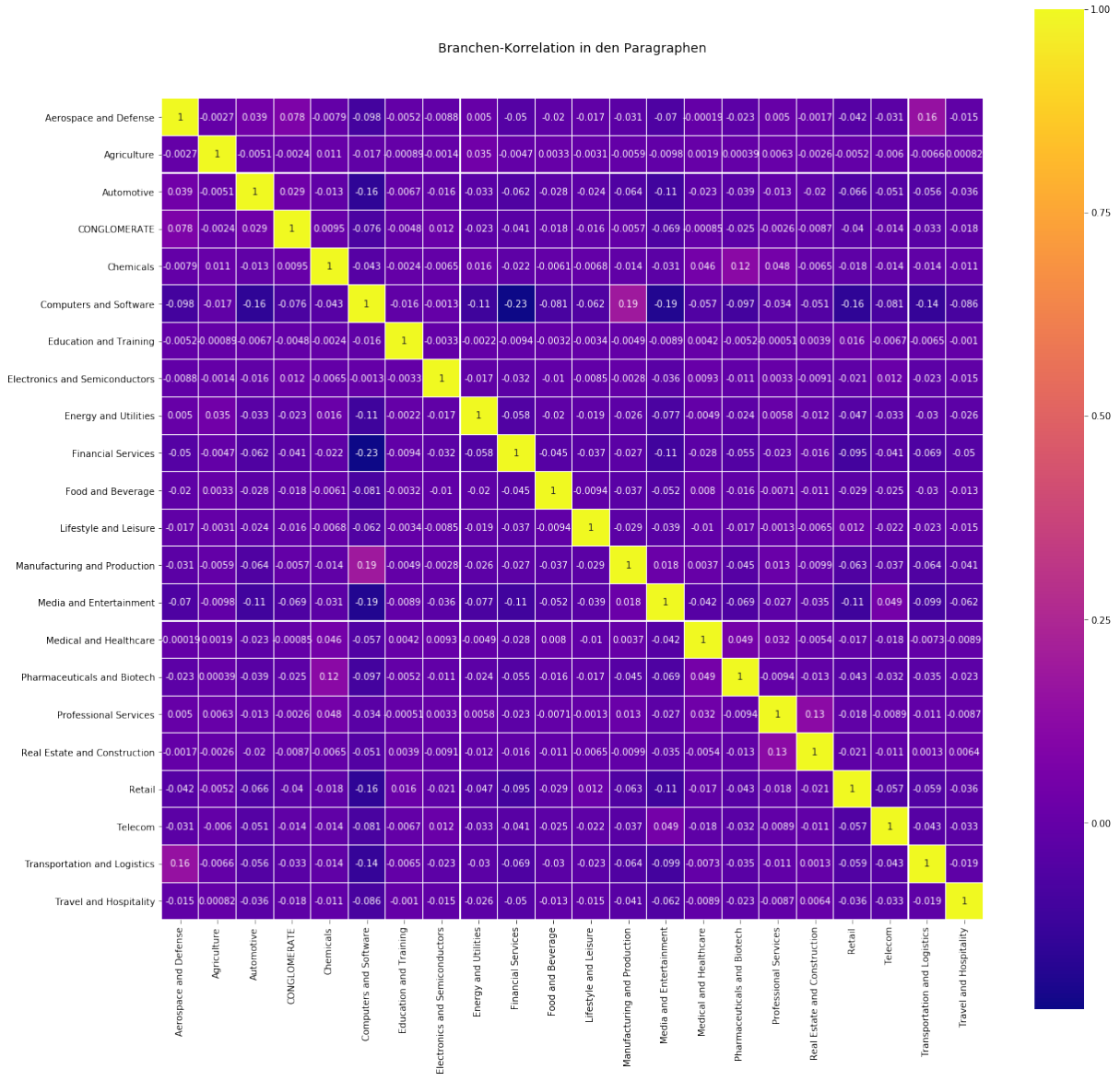


Abbildung 6.6: Korrelation der Branchen in den Paragraphen (niedrige Konfidenz), ermittelt durch den Phi-Koeffizienten.

6.2 Modellauswertung

Bevor die verschiedenen Modelle trainiert und ausgewertet werden, wird die Verteilung der Klassen betrachtet. In Abbildung 6.7 und 6.8 ist zu sehen, dass die Branchen unabhängig vom Konfidenzlevel, sowohl in den Dokumenten als auch in den Paragraphen, ähnlich verteilt sind. Die genauen Zahlen zur Verteilung der Branchen sind im Anhang

in den Tabellen C.1 und C.2 zu finden. Drei der 26 Branchen sind nicht in den Daten vertreten, aus diesem Grund werden die Branchen *Environment*, *Government* und *Sports* im Rahmen der Vorverarbeitung (siehe Kapitel 4.4.3) aus den Daten entfernt. Dadurch reduziert sich die Anzahl der betrachteten Klassen auf 23. Insgesamt sind die Branchen sehr unbalanciert, wobei die Branche *MISC* (Erstellung der Klasse in Kapitel 4.4.1 beschrieben) deutlich häufiger auftritt, als alle anderen Branchen. Modelle könnten dadurch auf die Klassen *MISC* oder *Computers and Software* übertrainiert werden. Auf selten vertretenen Klassen, wie beispielsweise *Agricultur*, zu denen wenige Informationen vorhanden sind, könnten sich Modelle hingegen nur schlecht anpassen.

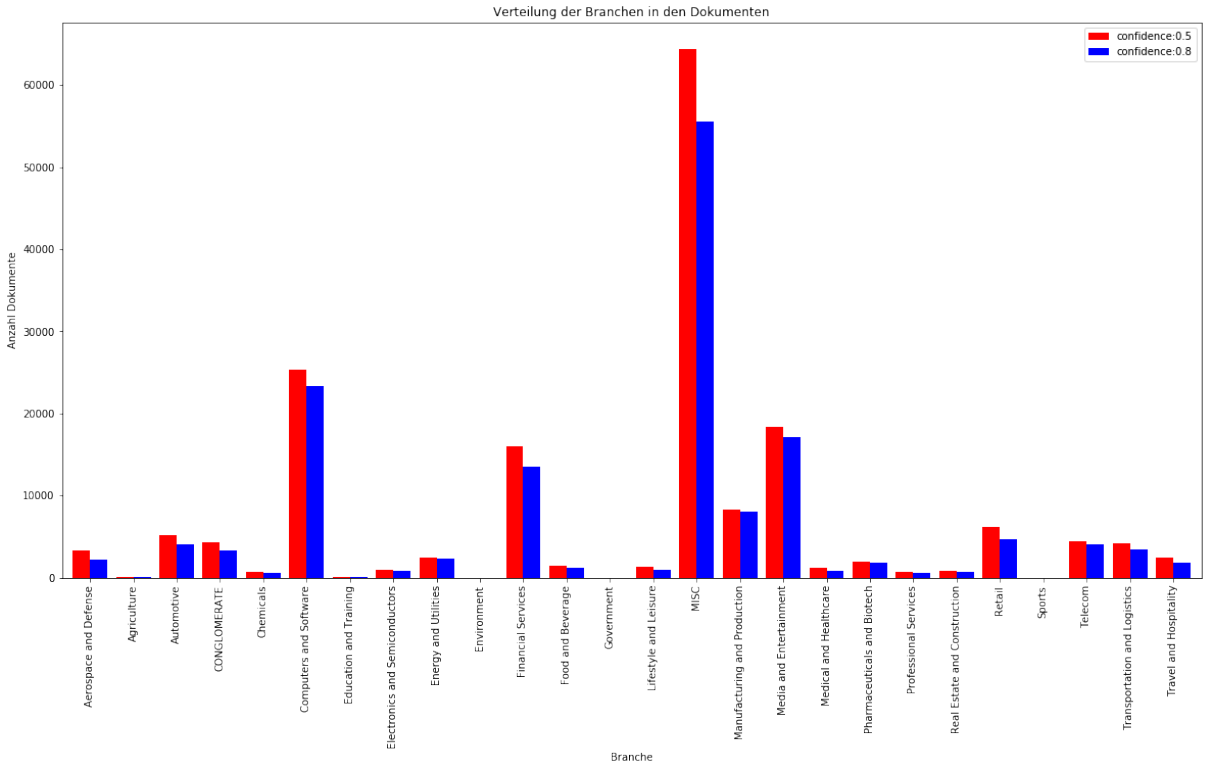


Abbildung 6.7: Verteilung der Branchen in den Dokumenten.

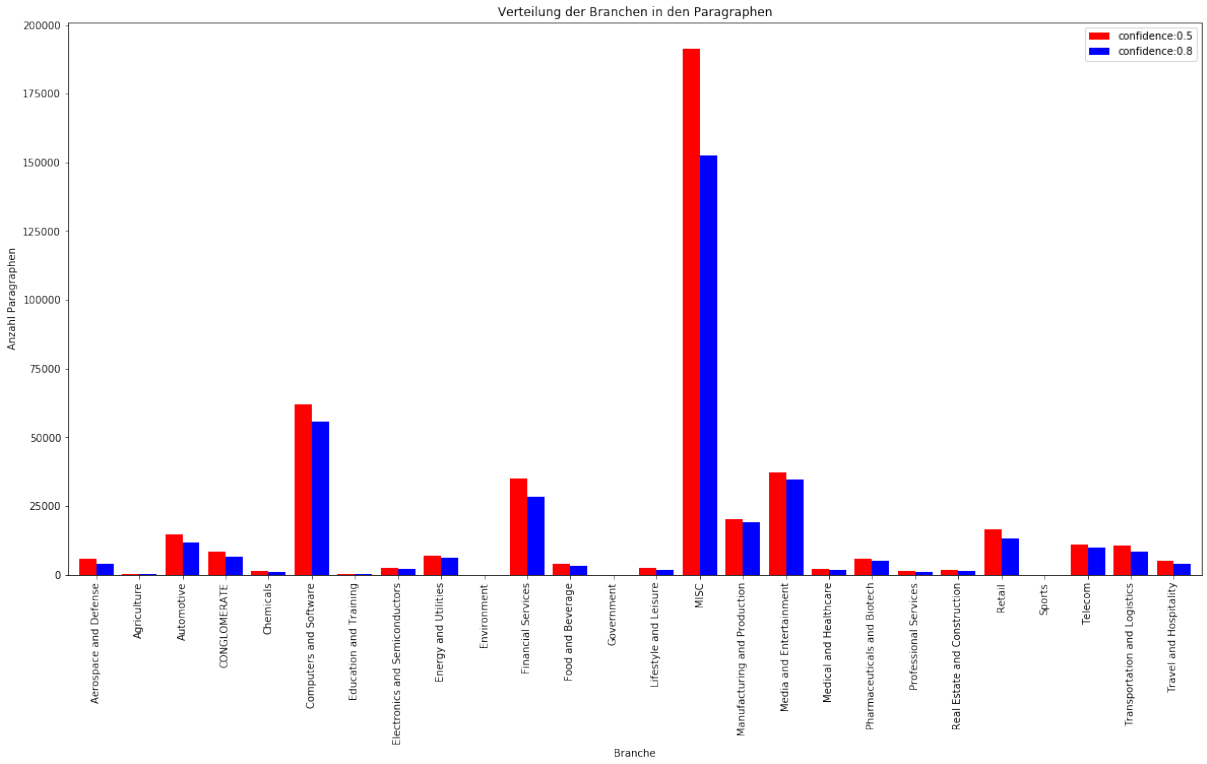


Abbildung 6.8: Verteilung der Branchen in den Paragraphen.

Die ersten Modelle, die trainiert werden, werden auf den Daten mit der niedrigen Konfidenz (Konfidenz-Parameter bei EL: 0.5) trainiert. Dabei werden die Modelle Naive Bayes, logistische Regression, lineare SVMs und Entscheidungsbäume verwendet. Als Features werden die BOW-, Tf-idf- und LDA-Vektoren unter Berücksichtigung von Uni- und Bigrammen verwendet, deren Dimensionalität in Tabelle 6.3 zu sehen ist.

Feature-Vektor	**Dimensionen**
BOW:	3.624.698
Tf-idf:	3.624.698
LDA:	23

Tabelle 6.3: Dimensionen der Feature-Vektoren aus den Dokumenten-Daten (niedrige Konfidenz).

In Tabelle 6.4 sind die Ergebnisse der Modelle mit den Metriken aus Kapitel 4.5 aufgeführt. Auffällig ist der niedrige Wert für die Exact Match Ratio, die für alle Modelle unter 60% liegt, und damit aussagt, dass eine große Menge der vorhergesagten Target-Vektoren nicht mit den tatsächlichen Vektoren übereinstimmen. Bei den Modellen, die mit den LDA-Vektoren trainiert werden, und bei den Naive Bayes Modellen liegen die Werte für die Exact Match Ratio sogar unter 30%. Die Werte für die Metrik Hamming Loss geben jedoch einen Hinweis darauf, dass die falschen Vorhersagen nur wenige Label innerhalb der Feature-Vektoren betreffen. Dies ist auch an den hohen Werten bei den verschiedenen Precision-Metriken zu sehen, die bis zu 0.9327 (Lineare SVM auf Tf-idf-Vektoren) erreichen. Diese sagen aus, dass der Großteil der vorhergesagten Branchen richtig vorhergesagt wird. Auch Werte für den Micro-Recall von bis zu 0.8275 (Lineare SVM auf BOW-Vektoren) deuten auf ein positives Ergebnis hin, denn diese sagen aus, dass der Großteil der Branchen bei der Vorhersage entdeckt wird. Es ist festzustellen, dass die Modelle auf den BOW-Vektoren eine bessere Performance aufweisen, als die Modelle auf den Tf-idf-Vektoren, gefolgt von den Modellen auf den LDA-Vektoren, die die schlechteste Performance aufweisen. Feature-übergreifend fällt das Naive Bayes Modell mit der schlechtesten Performance auf. Bei genauer Betrachtung der einzelnen Klassen, wie sie im Anhang C in Abbildung C.1 für das Modell der logistischen Regression auf BOW-Features zu sehen sind, sind die positiven Ergebnisse der Multi-Label- und Micro-Metriken auf die Unbalanciertheit der Branchen zurückzuführen. Die dominante Klasse *MISC* und andere häufig vertretene Klassen werden von den erstellten Modellen sehr gut erkannt und richtig klassifiziert, während die selten vertretenen Klassen, wie beispielsweise *Education and Training*, nicht erkannt und daher falsch klassifiziert werden. Da für die Ergebnisse der Multi-Label- und Micro-Metriken die Label nach ihrer Verteilung ins Gewicht fallen, fallen diese Metriken so positiv aus, obwohl das Modell manche Klassen nicht oder nur schlecht klassifiziert. Das Problem der Unbalanciertheit spiegelt sich jedoch in den niedrigen Werten der Macro-Recall-Metrik wider. Diese mittelt die Recall-Werte der einzelnen Label, ohne Rücksicht auf den Anteil der Label an der Gesamtmenge der Daten zu nehmen. Es werden alle Klassen unabhängig von ihrer Größe gleich gewichtet, sodass es auffällt, wenn einzelne Klassen nicht erkannt werden. Aus diesem Grund wird die Metrik im weiteren Vorgehen genauer beobachtet. Trotz hoher Werte bei der Precision oder dem Micro-Recall kann aus den genannten Gründen nicht von zufriedenstellenden Modellen in Hinblick auf das aufgebaute Szenario gesprochen werden, da diese überwiegend auf die Dummy-Klasse trainiert sind. Die Klasse *MISC* ist für das Szenario nicht relevant, da diese auf keine Branchenzugehörigkeit deutet, und darüber hinaus die Modelle durch

ihr häufiges Auftreten negativ beeinflusst. Sie wird deshalb im nächsten Schritt von der Modellbildung ausgeschlossen.

Features	Modell	EM	HS	HL	Precision (ML)	Recall (ML)	Precision (micro)	Recall (micro)	Precision (macro)	Recall (macro)
BOW	Naiv Bayes	0.1619	0.4091	0.1603	0.4466	0.7177	0.3258	0.7307	0.2339	0.5824
	Log. Regression	**0.5976**	**0.7729**	**0.0272**	0.8706	0.8192	0.8968	**0.7886**	0.8490	0.5677
	Lineare SVM	0.5944	0.7705	0.0279	0.8568	**0.8275**	0.8769	0.8025	0.8005	**0.6145**
	Decision Tree	0.5540	0.7360	0.0369	0.8460	0.7972	0.8219	0.7527	0.6698	0.5109
Tf-idf	Naiv Bayes	0.2683	0.4740	0.0644	0.7835	0.4782	0.7854	0.3411	0.3309	0.0520
	Log. Regression	0.4155	0.6261	0.0445	0.8376	0.6460	0.8861	0.5802	0.8388	0.2027
	Lineare SVM	0.5815	0.7457	0.0299	**0.8843**	0.7667	**0.9327**	0.7195	**0.9621**	0.4702
	Decision Tree	0.5450	0.7327	0.0374	0.8457	0.7964	0.8198	0.7497	0.6348	0.5029
LDA	Naiv Bayes	x	x	x	x	x	x	x	x	x
	Log. Regression	0.2554	0.4825	0.0663	0.7673	0.5146	0.7461	0.4275	0.2108	0.0716
	Lineare SVM	0.2574	0.4768	0.0665	0.7701	0.4984	0.7561	0.4133	0.1614	0.0590
	Decision Tree	0.2815	0.4754	0.0838	0.5932	0.6022	0.5450	0.5608	0.2891	0.3115

Tabelle 6.4: Ergebnisse der Modellbildung auf den Dokumenten.

Nach der Entfernung der Klasse *MISC* reduziert sich die Datenmenge weiter auf 63.048 Tupel. Die Ergebnisse der Modellbildung auf den reduzierten Daten sind in Tabelle 6.5 dargestellt. Die meisten Modelle weisen im Vergleich zu den ersten Modellen verschlechterte Ergebnisse auf. In nur wenigen Modellen ist die Performance für einzelne Metriken leicht gestiegen oder gleich geblieben, wie es beispielsweise bei der linearen SVM auf den Tf-idf-Vektoren bei der Micro- und Macro-Precision der Fall ist. Die Modelle auf den LDA-Vektoren und die Naive-Bayes-Modelle liegen wie schon zuvor hinter den restlichen Modellen. Auffällig ist die hohe Precision der meisten Modelle im Gegensatz zu deren Recall-Werten. Der Grund dafür ist erneut in der Unbalanciertheit der einzelnen Klassen zu finden, wie sie im Anhang C in Abbildung C.2 in den Metriken der linearen SVM auf BOW-Features für die einzelnen Klassen zu sehen ist. Besonders gut ist das Problem der Unbalanciertheit wie zuvor in den niedrigen Macro-Recall-Werten zu erkennen.

Features	Modell	EM	HS	HL	Precision (ML)	Recall (ML)	Precision (micro)	Recall (micro)	Precision (macro)	Recall (macro)
BOW	Naiv Bayes	0.1909	0.4278	0.1622	0.4485	0.8728	0.3063	0.8562	0.2342	0.6297
	Log. Regression	**0.6021**	0.7247	**0.0284**	**0.8068**	0.7569	0.8916	0.7252	0.8560	0.5576
	Lineare SVM	0.6013	**0.7340**	0.0297	0.8063	**0.7839**	0.8508	**0.7535**	0.7915	**0.6005**
	Decision Tree	0.5653	0.6955	0.0408	0.7799	0.7426	0.7629	0.6955	0.6095	0.5038
Tf-idf	Naiv Bayes	0.1481	0.1981	0.0674	0.2646	0.1984	0.9052	0.1559	0.3025	0.0410
	Log. Regression	0.3474	0.4513	0.0480	0.5737	0.4551	0.9507	0.4084	0.8362	0.1934
	Lineare SVM	0.5505	0.6491	0.0322	0.7478	0.6562	**0.9613**	0.6137	**0.9662**	0.4678
	Decision Tree	0.5457	0.6789	0.0425	0.7660	0.7276	0.7509	0.6851	0.6161	0.4928
LDA	Naiv Bayes	x	x	x	x	x	x	x	x	x
	Log. Regression	0.1777	0.2558	0.0734	0.3292	0.2680	0.6125	0.2170	0.2461	0.0701
	Lineare SVM	0.1699	0.2401	0.0735	0.3111	0.2499	0.6197	0.2018	0.1989	0.0583
	Decision Tree	0.2860	0.4024	0.0895	0.4711	0.4793	0.4408	0.4573	0.3018	0.3149

Tabelle 6.5: Ergebnisse der Modellbildung auf den Dokumenten ohne der Klasse *MISC*.

Das Größte Problem der bisher erstellten Modelle ist die schlechte Erkennung unterrepräsentierter Klassen. Im nächsten Schritt wird geklärt, ob dieses Problem bei den Modellen auftritt, die ohne die Klasse *MISC* auf den Paragraphen-Daten trainiert werden. Diese Daten unterscheiden sich von den Daten auf Dokumentenebene unter anderem dadurch, dass sie weniger Branchen pro Tupel aufweisen (siehe Abbildung 6.4), was bedeutet, dass die einzelnen Label eindeutiger voneinander abgegrenzt sind. Durch das Entfernen der Dummy-Klasse reduziert sich die Datenmenge auf 210.395 Tupel. Die Ergebnisse der Modellbildung auf diesen Daten ist in Tabelle 6.6 zu sehen. Es fällt auf, dass der Recall in allen Modellen, die nicht auf den LDA-Vektoren trainiert werden, deutlich gestiegen ist. Dennoch liegt der Macro-Recall bis auf eine Ausnahme unter 80% und bei der logisitischen Regression auf den Tf-idf-Vektoren sogar unter 35%. Bei der Betrachtung der einzelnen Klassen der logistischen Regression auf Tf-idf-Vektoren, wie sie im Anhang C in Abbildung C.3 zu sehen ist, führt auch in den Paragraph-Daten die Unbalanciertheit der Klassen zu unbalancierten Modellen. Insgesamt ist festzustellen, dass die Modelle, die auf den BOW-Vektoren trainiert werden, besser abschneiden, als die auf den Tf-idf-Vektoren. Sie weisen deutlich höhere Werte in den Recall-Metriken auf und gleichzeitig eine geringere Varianz zwischen den Precision- und Recall-Metriken. Insbesondere die lineare SVM sticht durch ihre besonders gute Performance hervor. Aufgrund der durchweg unzureichenden Performance für das erstellte Szenario werden im Folgenden keine Modelle mehr auf den LDA-Vektoren betrachtet. Die LDA-Features werden jedoch im späteren Verlauf mit anderen Features kombiniert, um festzustellen, ob dadurch ein Mehrwert bei

der Klassifikation entstehen kann.

Features	Modell	EM	HS	HL	Precision (ML)	Recall (ML)	Precision (micro)	Recall (micro)	Precision (macro)	Recall (macro)
BOW	Naiv Bayes	0.4953	0.6511	0.0807	0.6614	0.9311	0.3980	0.9194	0.2941	0.7460
	Log. Regression	0.8618	0.8876	0.0075	0.9030	0.8961	0.9725	0.8889	0.9576	0.7732
	Lineare SVM	**0.8951**	**0.9226**	**0.0058**	**0.9344**	**0.9361**	0.9645	**0.9284**	0.9529	**0.8638**
	Decision Tree	0.8426	0.8702	0.0130	0.8897	0.8779	0.8944	0.8659	0.8422	0.7648
Tf-idf	Naiv Bayes	0.3458	0.3733	0.0365	0.4043	0.3739	0.9737	0.3444	0.7196	0.1173
	Log. Regression	0.5693	0.5986	0.0232	0.6293	0.5994	**0.9902**	0.5834	0.8974	0.3237
	Lineare SVM	0.8133	0.8354	0.0099	0.8554	0.8373	0.9911	0.8268	**0.9901**	0.7013
	Decision Tree	0.8355	0.8636	0.0135	0.8843	0.8709	0.8912	0.8584	0.8402	0.7549
LDA	Naiv Bayes	x	x	x	x	x	x	x	x	x
	Log. Regression	0.1745	0.1877	0.0515	0.1907	0.1760	0.6769	0.1618	0.2194	0.0426
	Lineare SVM	0.1536	0.1643	0.0515	0.1640	0.1522	0.7156	0.1391	0.1377	0.0326
	Decision Tree	0.3675	0.4028	0.0638	0.4211	0.4202	0.4299	0.4139	0.3059	0.2844

Tabelle 6.6: Ergebnisse der Modellbildung auf den Paragraphen ohne der Klasse *MISC*.

Um dem Problem der Unbalanciertheit der Klassen entgegenzuwirken, werden im nächsten Schritt die Modelle unter Berücksichtigung der Klassenverteilung trainiert. Dafür werden die Regularisierungsparameter der Modelle mit der Klassenverteilung gewichtet, sodass selten auftretende Klassen beim Trainieren stärker ins Gewicht fallen und umgekehrt. Im Naive-Bayes-Modell ist eine solche Gewichtung nicht möglich. Die Ergebnisse der gewichteten Modelle auf den Parameter-Daten ohne die der Klasse *MISC* sind in Tabelle 6.7 zu sehen. Bei den Modellen auf den Tf-idf-Vektoren können durch die Gewichtung höhere Recall-Werte erreicht werden und gleichzeitig die Varianz zwischen den Precision- und Recall-Metriken verringert werden. Auch die logistische Regression auf den BOW-Vektoren profitiert von der Gewichtung. Die lineare SVM auf den BOW-Vektoren hat sich hingegen geringfügig verschlechtert. Auch nach der Gewichtung bleiben die Modelle auf den Tf-idf-Vektoren mit niedrigeren Recall-Werten und einer höheren Varianz in den Metriken hinter den Modellen auf den BOW Vektoren zurück. Aus diesem Grund werden die weiteren Modelle nur noch auf den BOW-Vektoren trainiert.

Features	Modell	EM	HS	HL	Precision (ML)	Recall (ML)	Precision (micro)	Recall (micro)	Precision (macro)	Recall (macro)
BOW	Naiv Bayes	x	x	x	x	x	x	x	x	x
	Log. Regression	**0.8859**	**0.9212**	**0.0064**	**0.9317**	**0.9437**	0.9455	**0.9366**	0.8707	**0.8838**
	Lineare SVM	0.8806	0.9147	0.0068	0.9282	0.9332	0.9512	0.9234	0.8845	0.8431
	Decision Tree	0.7905	0.8239	0.0181	0.8431	0.8373	0.8384	0.8310	0.7452	0.7829
Tf-idf	Naiv Bayes	x	x	x	x	x	x	x	x	x
	Log. Regression	0.8208	0.8611	0.0100	0.8777	0.8824	0.9406	0.8736	0.8996	0.8089
	Lineare SVM	0.8572	0.8816	0.0076	0.8980	0.8882	**0.9794**	0.8793	**0.9595**	0.7891
	Decision Tree	0.7742	0.8095	0.0199	0.8295	0.8250	0.8175	0.8194	0.7192	0.7582

Tabelle 6.7: Ergebnisse der gewichteten Modellbildung auf den Paragraphen ohne die der Klasse *MISC*.

Im nächsten Schritt wird untersucht, ob die bisherigen Ergebnisse, die auf den Daten mit dem kleineren Konfidenzwert beim EL generiert werden, verbessert werden können, indem beim Generieren der Daten eine höhere Konfidenz verwendet wird. Dazu werden Modelle auf den Paragraphen-Daten der höheren Konfidenz trainiert, ebenfalls ohne die Klasse *MISC*. Die Daten umfassen 180.695 Tupel. Als Modelle werden hierfür diejenigen verwendet, die bei den bisherigen Ergebnissen die beste Performance gezeigt haben. Diese Modelle sind: lineare SVM, gewichtete logistische Regression und gewichtete lineare SVM. Alle Modelle werden auf den BOW-Features trainiert. Die Ergebnisse sind in Tabelle 6.8 zu sehen. Alle Modelle wurden im Durchschnitt der Metriken durch die höhere Konfidenz der Daten leicht verbessert.

Features	Modell	EM	HS	HL	Precision (ML)	Recall (ML)	Precision (micro)	Recall (micro)	Precision (macro)	Recall (macro)
BOW	Lineare SVM	**0.9179**	**0.9401**	**0.0045**	**0.9501**	0.9507	**0.9735**	0.9431	**0.9479**	0.8602
	Log. Regression (gewichtet)	0.9051	0.9351	0.0053	0.9443	**0.9535**	0.9562	**0.9462**	0.8584	**0.8683**
	Lineare SVM (gewichtet)	0.9049	0.9331	0.0054	0.9444	0.9485	0.9615	0.9392	0.8767	0.8435

Tabelle 6.8: Ergebnisse der Modellbildung auf den Paragraphen auf hoher Konfidenz und ohne die Klasse *MISC*.

Eine weitere Variationsmöglichkeit bei der Erstellung der Modelle stellt die Kombination aus verschiedenen Features dar. Es wird untersucht, ob die LDA-Features, die einzeln zu keinen positiven Ergebnissen geführt haben, in Kombination mit anderen Features

einen Mehrwert bringen. Dazu werden die bisher besten Modelle, lineare SVM, gewichtete logistische Regression und gewichtete lineare SVM mit kombinierten BOW-LDA-Features, auf den Paragraphen-Daten der hohen Konfidenz trainiert. Die Ergebnisse, die in Tabelle 6.9 zu sehen sind, deuten auf geringe Änderungen im Gegensatz zu den zuvor erstellten Modellen hin. Das Modell lineare SVM hat sich geringfügig verschlechtert, während sich die Modelle gewichtete logistische Regression und gewichtete lineare SVM geringfügig verbessert haben.

Features	**Modell**	**EM**	**HS**	**HL**	**Precision (ML)**	**Recall (ML)**	**Precision (micro)**	**Recall (micro)**	**Precision (macro)**	**Recall (macro)**
BOW + LDA	Lineare SVM	**0.9172**	**0.9395**	**0.0046**	**0.9496**	0.9501	**0.9734**	0.9424	**0.9402**	0.8580
	Log. Regression (gewichtet)	0.9071	0.9363	0.0052	0.9457	**0.9540**	0.9579	**0.9468**	0.8736	**0.8687**
	Lineare SVM (gewichtet)	0.9051	0.9329	0.0054	0.9442	0.9478	0.9625	0.9384	0.8874	0.8444

Tabelle 6.9: Ergebnisse der Modellbildung auf den Paragraphen mit kombinierten BOW+LDA Features auf hoher Konfidenz und ohne die Klasse *MISC*.

Abschließend wird auf den vier verschiedenen Datensätzen ohne der Klasse *MISC* jeweils ein CNN trainiert. In den Ergebnissen in Tabelle 6.10 ist zu erkennen, dass wie zuvor die anderen Modelle, die CNNs auf den Dokumenten-Datensätzen keine positiven Ergebnisse erreichen. Insbsondere in der Macro-Recall-Metrik ist dies zu erkennen. Auf den Paragraph-Daten schneiden die CNNs gut ab. Analog zu den vorher trainierten Modellen ist das beste CNN-Modell das, das auf den Paragraph-Daten der hohen Konfidenz trainiert wird.

Daten	EM	HS	HL	Precision (ML)	Recall (ML)	Precision (micro)	Recall (micro)	Precision (macro)	Recall (macro)
Doc: Ohne MISC	0.5952	0.7167	0.0298	0.8043	0.7478	0.8966	0.7004	0.8910	0.4882
Doc: Ohne MISC; Hohe Konfidenz	0.6267	0.7431	0.0268	0.8223	0.7777	0.8957	0.7319	0.8884	0.5014
Paragraph: Ohne MISC	0.8842	0.9159	0.0066	0.9310	0.9307	0.9589	0.9194	**0.9535**	**0.8260**
Paragraph: Ohne MISC; Hohe Konfidenz	**0.9079**	**0.9338**	**0.0053**	**0.9459**	**0.9462**	**0.9665**	**0.9358**	0.9397	0.8207

Tabelle 6.10: Ergebnisse Convolutional Neural Networks auf den Dokumenten- und Paragraphen Daten, jeweils ohne die Klasse *MISC*.

Zum Vergleich und zur Übersicht über die trainierten Modelle werden die besten Modelle gegenübergestellt. Diese sind die lineare SVM auf den BOW-Vektoren, die gewichtete logistische Regression und die gewichtete lineare SVM auf den kombinierten BOW-LDA-Vektoren und das CNN auf den Paragraph-Daten. Alle Modelle werden auf den Paragraph-Daten der hohen Konfidenz trainiert. Zur besseren Vergleichbarkeit wird neben den Metriken *Exact Match* und *Hamming Score* jeweils der F1-Wert der Multi-Label-, Micro- und Macro-Metriken gebildet. Die Ergebnisse sind in Tabelle 6.11 gegenübergestellt. Das beste Modell ist die lineare SVM auf den BOW-Vektoren, die insbesondere in den Macro-Metriken mit Abstand die besten Ergebnisse erzielt. Insgesamt liegen die Modelle sehr eng beieinander und unterscheiden sich in den Werten der Metriken erst an der dritten Nachkommastelle voneinander. Bezüglich des Aufwandes zum Erstellen der Modelle liegt ebenfalls das lineare SVM-Modell auf den BOW-Features vorne, da hier nicht noch ein zusätzliches LDA-Modell trainiert werden muss, und die Trainingsdauer deutlich hinter der eines neuronalen Netzwerkes liegt.

Daten	Features	Modell	EM	HS	F1 (ML)	F1 (micro)	F1 (macro)
Paragraph: Ohne MISC; Hohe Konfidenz	BOW	Lin. SVM	**0.9179**	**0.9401**	**0.9504**	**0.9581**	**0.9019**
	BOW + LDA	Log. Regression (gewichtet)	0.9071	0.9363	0.9498	0.9523	0.8711
	BOW + LDA	Lin. SVM (gewichtet)	0.9051	0.9329	0.9460	0.9503	0.8654
	Word Embeddings	CNN	0.9079	0.9338	0.9460	0.9509	0.8762

Tabelle 6.11: Ergebnisse der besten Modelle gegenübergestellt mit F1-Metrik.

Die Metriken für die einzelnen Branchen des besten Modells (lineare SVM auf BOW-Vektoren und Paragraph-Datensatz mit hoher Konfidenz) werden betrachtet, um das Ergebnis besser einordnen zu können. Wie in Abbildung 6.9 zu sehen ist, schwanken die Klassifikationsergebnisse für die einzelnen Branchen teils stark. Die Branche *Agriculture* weist mit einem F1-Score von 0,50 das schlechteste Ergebnis auf. Gleichzeitig ist die Branche aber mit 10 Beobachtungen auch diejenige, die am wenigsten in den Test-Daten vertreten ist. Die Branche *Computers and Software*, die am häufigsten in den Test-Daten vertreten ist, erreicht sogar einen F1-Score von 0,99. Es zeichnet sich ab, dass Branchen, die häufiger vertreten sind, besser klassifiziert werden. Insgesamt sind die Ergebnisse der einzelnen Klassen bis auf die beiden Klassen *Agriculture* und *Education and Training*, die weniger als 50 Mal vertreten sind, als positiv zu bewerten.

	precision	recall	f1-score	support
Aerospace and Defense	0.92	0.87	0.89	767
Agriculture	0.67	0.40	0.50	10
Automotive	0.97	0.95	0.96	2235
CONGLOMERATE	0.92	0.87	0.90	1203
Chemicals	0.97	0.89	0.93	221
Computers and Software	0.99	0.98	0.99	11188
Education and Training	0.88	0.60	0.71	25
Electronics and Semiconductors	0.99	0.85	0.92	413
Energy and Utilities	0.97	0.94	0.96	1248
Financial Services	0.97	0.95	0.96	5637
Food and Beverage	0.98	0.92	0.95	652
Lifestyle and Leisure	0.95	0.83	0.89	345
Manufacturing and Production	0.97	0.94	0.96	3774
Media and Entertainment	0.97	0.95	0.96	6919
Medical and Healthcare	0.95	0.85	0.89	308
Pharmaceuticals and Biotech	0.98	0.95	0.97	970
Professional Services	0.98	0.68	0.80	201
Real Estate and Construction	0.96	0.83	0.89	281
Retail	0.97	0.92	0.95	2686
Telecom	0.95	0.89	0.92	2050
Transportation and Logistics	0.96	0.94	0.95	1728
Travel and Hospitality	0.96	0.91	0.94	833

Abbildung 6.9: Übersicht der Metriken für die einzelnen Klassen aus dem Modell lineare SVM auf den Paragraph-Daten (hohe Konfidenz) ohne der Klasse *MISC* unter Verwendung der BOW-Vektoren.

Um festzustellen, wie die Ergebnisse im Gegensatz zu zufälligen Klassifkationen einzuordnen sind, werden zwei Zufallssimulationen durchgeführt. In der ersten Simulation werden die einzelnen Label mit der Wahrscheinlichkeit ihres Vorkommens in den Testdaten gesetzt, in der zweiten Simulation wird mit einer Wahrscheinlichkeit von 50% für jedes Label ein Münzwurf simuliert. Als Test-Datensatz dient der Paragraphen-Datensatz auf hoher Konfidenz. Beide Simulationen klassifizieren die gesamten Test-Daten 1.000 Mal. Nach jedem Durchlauf über den gesamten Test-Datensatz werden die zuvor verwendeten Metriken Precision, Recall und F1-Score gebildet. Die Ergebnisse in Abbildung 6.10 sind die gemittelten Metriken der 1.000 Durchläufe. Die Ergebnisse aus der Simulation nach der Verteilung der Branchen zeigen, dass sich die Metriken der Verteilung annähern, während sich bei der Münzwurf-Simulation die Precision-Werte an die Verteilung annähern und die Recall-Werte den 50% aus dem Münzwurf. Die Simulationen zeigen, dass die Ergebnisse der zuvor erstellten ML-Modelle einen klaren Mehrwert gegenüber einer Zufallsklassifikation bringen und nicht alleine von der Verteilung der Branchen in den Daten abhängen.

Branche	Precision	Recall	F1
Aerospace and Defense	0.021418	0.021394	0.021399
Agriculture	0.000502	0.000400	0.000442
Automotive	0.061961	0.061946	0.061947
CONGLOMERATE	0.033548	0.033576	0.033555
Chemicals	0.005876	0.005837	0.005850
Computers and Software	0.309687	0.309839	0.309758
Education and Training	0.000641	0.000680	0.000656
Electronics and Semiconductors	0.011173	0.011186	0.011173
Energy and Utilities	0.034940	0.034947	0.034937
Financial Services	0.156083	0.156111	0.156091
Food and Beverage	0.018006	0.017982	0.017987
Lifestyle and Leisure	0.009603	0.009594	0.009591
Manufacturing and Production	0.104466	0.104495	0.104474
Media and Entertainment	0.191654	0.191601	0.191622
Medical and Healthcare	0.008732	0.008731	0.008724
Pharmaceuticals and Biotech	0.026898	0.026894	0.026889
Professional Services	0.005496	0.005607	0.005544
Real Estate and Construction	0.007896	0.007875	0.007878
Retail	0.074298	0.074347	0.074316
Telecom	0.056618	0.056608	0.056607
Transportation and Logistics	0.047987	0.048019	0.047996
Travel and Hospitality	0.023019	0.023028	0.023016

(a)

Branche	Precision	Recall	F1
Aerospace and Defense	0.021196	0.499411	0.040666
Agriculture	0.000277	0.499800	0.000553
Automotive	0.061837	0.499890	0.110059
CONGLOMERATE	0.033332	0.500681	0.062503
Chemicals	0.006115	0.499937	0.012082
Computers and Software	0.309694	0.500162	0.382527
Education and Training	0.000688	0.497520	0.001375
Electronics and Semiconductors	0.011395	0.498702	0.022281
Energy and Utilities	0.034569	0.500490	0.064671
Financial Services	0.155924	0.499907	0.237705
Food and Beverage	0.018057	0.500451	0.034856
Lifestyle and Leisure	0.009552	0.500359	0.018746
Manufacturing and Production	0.104405	0.499732	0.172724
Media and Entertainment	0.191331	0.499630	0.276699
Medical and Healthcare	0.008522	0.499951	0.016759
Pharmaceuticals and Biotech	0.026856	0.500163	0.050976
Professional Services	0.005561	0.499925	0.011000
Real Estate and Construction	0.007770	0.499722	0.015303
Retail	0.074309	0.499984	0.129388
Telecom	0.056736	0.500039	0.101908
Transportation and Logistics	0.047818	0.500044	0.087289
Travel and Hospitality	0.023048	0.500050	0.044064

(b)

Abbildung 6.10: Übersicht der Metriken für die einzelnen Klassen aus (a) einer simulierten Klassifikation nach der Verteilung der Klassen in den Test-Daten und (b) einer simulierten Zufalls-Klassifikation.

7 Schlussfolgerung

Im abschließenden Kapitel dieser Arbeit werden in Abschnitt 7.1 die Ergebnisse und Erkenntnisse nochmals zusammengefasst. Im Anschluss wird in Abschnitt 7.2 ein Überblick darüber gegeben, wie auf den Ergebnissen aufgebaut werden kann, und wie der aktuelle Stand der Forschung zu verschiedenen aufgegriffenen Themen ist.

7.1 Zusammenfassung

Ziel dieser Arbeit war es, aufzuzeigen, wie aus frei zugänglichen Web-Texten mit Hilfe von Methoden aus der *Information Extraction* unter Hinzunahme von Informationen aus *Knowledge Bases* neues Branchenwissen zu Firmen generiert werden kann. Dieses Ziel wird durch die Umsetzung der beschriebenen Verarbeitungs- und Klassifikations-Pipeline (Kapitel 4) erreicht. In dieser werden Nachrichten von Webseiten zusammen mit Informationen der Knowledge Base DBpedia aufbereitet und daraus ein *Multi-Label Klassifikationsproblem* aufgebaut, das Nachrichtentexte nach Branchen klassifiziert.

Aus der Frage, wie sich Nachrichtentexte aus dem CC-Datensatz extrahieren lassen, ergeben sich zwei Möglichkeiten. Eine Möglichkeit, die in dieser Arbeit bei der Erstellung der Trainings- und Testdaten angewendet wird, ist die Extraktion bestimmter Nachrichtenseiten über die CC-Indexsuche (Kapitel 4.3). Die zweite Möglichkeit kommt in dieser Arbeit bei der Beantwortung der Frage nach dem Potential des CC-Datensatzes für das Klassifikationsproblem zum Einsatz. Hierbei werden die Dateien des Datensatzes durchlaufen und jeder Record auf seine Relevanz hin untersucht (Kapitel 3.1.3).

Das Ergebnis der Frage nach den potentiellen Texten im CC-Datensatz ist, dass dieser bezüglich des beschriebenen Klassifikationsproblems eine große Menge relevanter Texte beinhaltet (Kapitel 3.1.3). In $39,6\%$ der untersuchten Records werden per NER Firmen entdeckt, und in $31,5\%$ der Records können Firmen mit DBpedia gelinkt werden. Hochgerechnet auf den gesamten November-2018-Crawl bedeutet dies, dass schätzungsweise in

986.720.000 Records Informationen zu Firmen enthalten sind und 784.616.000 der Records mit Firmen in DBpedia in Verbindung gebracht werden können. Dabei ist jedoch nicht die Struktur und Qualität der Daten geklärt und es muss weiterhin beachtet werden, dass bei der NER und dem EL Fehler auftreten können.

Die Identifikation der Firmen in den Nachrichtentexten findet mit dem Tool Stanford CoreNLP statt, das eine NER auf den Daten durchführt. Mit dem Tool DBpedia Spotlight wird hingegen ein EL durchgeführt, das gefundene Firmen mit den Einträgen der Knowlege Base DBpedia verknüpft (Kapitel 5.1). Mit den in DBpedia hinterlegten Branchen zu Firmen lassen sich die Texte mit einem Branchenbezug anreichern (Kapitel 5.2). In DBpedia sind jedoch nur 109.625 Firmen weltweit enthalten, und für nur etwa die Hälfte der Firmen sind eine oder mehrere Branchen hinterlegt. Dies hat zur Folge, dass beim EL weniger Firmen entdeckt werden, als in den Nachrichtentexten enthalten sind. Die NER deutet auf deutlich mehr Firmen hin (Kapitel 6.1). Mehr als doppelt so viele Entitäten werden in der NER gespottet als mit dem EL gelinkt. Außerdem werden bei der NER eine größere Anzahl unterschiedlicher Terme als Entität entdeckt, als im EL unterschiedliche Entitäten. Die Schnittmenge zwischen NER und EL deutet auf falsche oder fehlende Klassifikationen in den beiden Tools hin, da diese nicht mit der Menge der gelinkten Entitäten übereinstimmt.

Bei der Modellauswertung (Kapitel 6.2) wird ersichtlich, welche Einflussfaktoren sich bei der durchgeführten Branchenklassifikation auf die Ergebnisse auswirken. Ein großer Einflussfaktor, der sich durch die gesamte Modellbildung zieht, ist die Verteilung der Branchen. Diese ist stark unbalanciert. Von den 26 definierten Branchen sind nur 23 vertreten. Die dominanteste Klasse ist dabei die Klasse *MISC*, die stellvertretend für die Branchen von Unternehmen steht, die in der Knowledge Base nicht angegeben sind, oder die Branchen, die nicht zugeordnet werden können, da sie nicht unter den berücksichtigten Branchen auftreten. Da die Klasse *MISC* lediglich die Branchen vertritt, die aus den genannten Gründen nicht zugeordnet werden können, wird sie von der Klassifikation ausgeschlossen. Zwischen den Branchen können keine starken Korrelationen festgestellt werden. Die Unbalanciertheit der Klassen führt dazu, dass in manchen Modellen selten vertretene Branchen schlechter erkannt werden, als stark vertretene. Dies ist insbesondere an den Werten der Macro-Recall-Metrik zu erkennen. Eine Gewichtung im Training dieser Modelle nach der Verteilung der Branchen kann dabei zu einer Verbesserung führen.

Einen weiteren großen Einfluss auf die Modelle hat die Art der Aufbereitung der Daten. In den Paragraphen-Daten sind die Texte klarer nach den Branchen getrennt. In 84% der Paragraphen ist nur eine einzelne Branche vertreten, während in den Daten auf Dokumentenebene ca. 40% der Daten mehr als eine Branche aufweisen. Die Modelle, die auf den Paragraphen trainiert sind, schneiden deutlich besser ab, als die, die auf den ganzen Dokumenten trainiert werden. In der Macro-Recall-Metrik sind Verbesserungen von bis zu 0.2631 (lineare SVM) möglich.

Auch die Wahl der Features hat im durchgeführten Klassifikationsproblem einen starken Einfluss auf die Güte der Modelle. Die Modelle auf den BOW-Vektoren erzeugen dabei die besten und stabilsten Ergebnisse, da hier sowohl die Precision-, als auch die Recall-Metriken hohe Werte aufweisen. Bei den Modellen auf den Tf-idf-Vektoren werden meist hohe Werte in den Precision-Metriken erzielt, die Macro-Recall-Werte liegen jedoch deutlich unter diesen. Die Modelle auf den Feature-Vektoren, die mit dem LDA-Modell erzeugt werden, schneiden mit Abstand am schlechtesten ab. Dabei muss jedoch berücksichtigt werden, dass, wie auch bei den anderen Modellen, im LDA-Modell keine Optimierungen durchgeführt werden. Ein geringer Mehrwert kann durch die Kombination der LDA-Features mit anderen Features, wie den BOW-Vekoren, erzielt werden.

Schon beim EL für die Erstellung der Trainingsdaten kann durch die Wahl des Konfidenz-Parameters Einfluss auf die Klassifikatoren genommen werden. Mit der Wahl einer höheren Konfidenz reduziert sich die Anzahl der gelinkten Entitäten und somit die Menge der Daten für das Training der Modelle. Jedoch werden dadurch weniger Entitäten falsch gelinkt, was zu einer besseren Qualität der Daten führt. Die Modelle, die auf den Daten mit höherer Konfidenz trainiert werden, weisen eine verbesserte Performance im Klassifikationsproblem auf.

Abschließend wird die Frage beantwortet, welche Methoden sich zum Lösen des Multi-Label-Klassifikationsproblems auf den gegeben Daten eignen. Bei den verwendeten Modellen zur Klassifizierung zeigt sich, dass das lineare SVM-Modell auf den Paragraphen-Daten der hohen Konfidenz unter Verwendung der BOW-Vektoren am besten abschneidet (Kapitel 6.2). Die gewichteten Modelle der logistischen Regression und der linearen SVM auf den kombinierten BOW-LDA-Vektoren und dem gleichen Datensatz schneiden ähnlich gut ab. Jedoch muss für diese durch das Trainieren des LDA-Modells ein höherer Aufwand betrieben werden. Das Entscheidungsbaum-Modell liegt hinter den beiden bereits

genannten Modellen, weist jedoch auf den Paragraph-Daten und bei den BOW- und Tf-idf-Vektoren Werte für die Metriken $> 75\%$ auf. Auf den LDA-Vektoren performen die Entscheidungsbäume am besten, jedoch sind die Ergebnisse, wie auch die anderen Modelle auf den LDA-Features, ungenügend. Die Naive-Bayes-Modelle erzeugen im Klassifikationsproblem die schlechtesten Ergebnisse. Der Großteil der Metriken liegt hier (teilweise deutlich) unter 50%. Die erstellten neuronalen Netze in Form von CNNs erreichen auf den Paragraphen-Daten mit hoher Konfidenz ähnlich gute Ergebnisse wie das „Sieger"-Modell, das lineare SVM-Modell auf den BOW-Features. Allerdings weisen sie eine deutlich höhere Trainingszeit auf.

Die positiven Ergebnisse der Modelle zeigen, dass diese auf den aufbereiteten Daten und dem erstellten Klassifikationsproblem dazu geeignet sind, Branchen für die gegebenen Nachrichtentexte vorherzusagen. Daraus ergibt sich die Möglichkeit mit Nachrichtentexten, in denen sich unbekannte Entitäten befinden, einen Rückschluss auf die Branchen der Entitäten zu erhalten. Dadurch lassen sich Knowledge Bases (im Fall dieser Arbeit DBpedia) um weitere Informationen erweitern. Die Richtigkeit der Branchenzuordnung in DBpedia ist dabei nicht garantiert. Weiterhin muss berücksichtigt werden, dass durch die Aufbereitung der Daten ein großer Teil der Daten und Branchen aus dem Klassifikationsproblem ausgeschlossen wird. Es handelt sich dabei um sehr „reine" Daten. Eine Aussage zur Performance der Modelle auf beliebigen Daten und weiteren *DBpedia*-Branchen kann dadurch nicht getroffen werden.

7.2 Ausblick

Durch die Erkenntnisse, die im vorherigen Abschnitt aufgezeigt wurden, entstehen viele Möglichkeiten und Ansatzpunkte, auf den Ergebnissen dieser Arbeit aufzubauen. Der erste Ansatzpunkt ist der Umfang der verwendeten Daten. Wie bei der Untersuchung auf relevante Daten (siehe Kapitel 3.1.3) zu sehen ist, bietet CC das Potential auf einer viel größeren Datenmenge zu arbeiten. Anstatt Nachrichten gezielt nach vorselektierten Nachrichtenseiten zu extrahieren, kann der CC-Datensatz durchlaufen und alle englischen Texte mit enthaltenen Entitäten extrahiert werden. Des Weiteren können auch die anderen Schritte der Datenaufbereitungs- und Modellbildungspipline, für die dortige parallele Ausführung, auf ein Spark-Cluster übertragen werden. Die ML-Modelle Naive Bayes, logistische Regression, SVMs und Entscheidungsbäume sind in *MLlib*, der ML-Bibliothek von Spark, implementiert und für die Verarbeitung im Cluster optimiert. Auch LDA ist in

MLlib implementiert, es besteht jedoch auch die Möglichkeit, *gensim* auf einem Cluster auszuführen. Die Daten aus DBpedia können frei zugänglich heruntergeladen und anschließend in einer Graphdatenbank bereitgestellt werden. Neuronale Netze, die mit *keras* implementiert werden, lassen sich mit *Elephas*[1] auf *Spark* übertragen.

In dieser Arbeit wird die Knowledge Base DBpedia eingesetzt. Da diese jedoch nicht die einzige frei zugängliche Knowledge Base ist, die Firmen enthält, könnte eine Weiterführung dieser Arbeit darin bestehen, Informationen aus verschiedenen Knowledge Bases einzubeziehen. Die Knowledge Base *Yago*[2] vom *Max-Planck-Institut für Informatik*, die ihre Informationen aus *Wikipedia*, *WordNet* und *GeoNames* ableitet, enthält ebenfalls Firmen und wird häufig für Forschungszwecke eingesetzt. Sie ist daher gut für das beschriebene Multi-Label Klassifikationsproblem geeignet. Eine weitere potentiell relevante Knowledge Base stellt *Open PermID*[3] des Medienkonzerns *Thomson Reuters* dar. *Open PermID* ist auf einen wirtschaftlichen Kontext ausgerichtet und enthält unter anderem Informationen und Metadaten zu Organisationen, Fonds und Personen weltweit.

Das vorgestellte Multi-Label Klassifikationsproblem berücksichtigt nur einen Teil der Branchen, die sich in DBpedia befinden (siehe Kapitel 4.4.1). Zudem werden die Branchen manuell auf eine kleinere Klassenstruktur gemappt. Um das volle Potential von DBpedia auszunutzen, können alle enthaltenen Informationen einbezogen, also alle dort enthaltenen Branchen, verwendet werden. Das Klassifikationsproblem, das dadurch entsteht, ist aufgrund der 12.586 möglichen Branchen (Summe der *dbo:industry*- und *dbr:industry*-Branchen, siehe Kapitel 3.2.3) nicht nur ein Multi-Label Klassifikationsproblem, sondern eines aus dem Bereich *Extreme multi-label text classification (XMTC)* [LCWY17]. Methoden aus diesem Bereich basieren auf *Target Embeddings*, bei denen versucht wird, niedriger dimensionale Embeddings für die Target Vektoren zu finden, Baum-basierte Ensemble-Methoden, in denen der Target-Raum partitioniert wird und Deep Learning Methoden. Ein Deep Learning Verfahren wird von Liu et al. [LCWY17] vorgestellt und basiert auf dem CNN-Modell von Kim, das auch in dieser Arbeit verwendet wird (siehe Kapitel 2.2.6).

[1] `https://github.com/maxpumperla/elephas/blob/master/README.md`

[2] `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/`

[3] `https://permid.org/`

Ein weiterer Ansatzpunkt, der auch bei der Multi-Label Klassifikation mit vielen Branchen relevant ist, ist das Auftreten von Korrelationen zwischen den Labeln bzw. Branchen. Diese werden in dieser Arbeit nicht thematisieriert, da der Binary-Relevance-Ansatz diese nicht berücksichtigt, sondern für jedes Label unabhängig ein Modell trainiert. Außerdem werden die Branchen durch das manuelle Mapping ohne Auftreten stärkerer Korrelationen voneinander getrennt. Bei auftretenden Korrelationen gibt es verschiedene Vorgehensweisen, über die Zhang et al. [ZZ14] einen Überblick gibt. Eine weit verbreitete Möglichkeit ist es, anstatt der Binary-Relecance-Methode, die *Classifier Chain* Methode [RPHF11] zu verwenden. Dabei werden ähnlich zur Binary Relevance einzelne Modelle trainiert, allerdings erfolgt die Erstellung der Modelle nacheinander und bezieht jeweils die vorherigen Modelle als Feature mit ins Training ein.

Das LDA-Modell, das in dieser Arbeit als unsupervised Methode zur Reduktion von Dimensionen eingesetzt wird, bietet noch weitere Möglichkeiten, in der Multi-Label Klassifikation eingesetzt zu werden. Eine dieser Möglichkeiten stellt die Methode *Labeled LDA* von Ramage et al. [RHNM09] dar. Es handelt sich hierbei um einen *supervised* Modell, in dem die Topics in einer 1-zu-1-Beziehung den Labeln zugeordnet werden. Sowohl die Wort-Label-Verteilung, als auch die Dokument-Label-Verteilung können gleichzeitig auf einem gegebenen Mulit-Label-Korpus gelernt werden. Aufbauend auf dem Standard-LDA-Modell von Blei et al. [BNJ03] und den Labeled-LDA-Modell von Ramage et al. [RHNM09] stellen Rubin et al. [RCSS12] drei verschiedene LDA-Modelle für die Multi-Label Klassifikation vor. Darunter ein *unsupervised* Modell, ein *supervised* Modell und ein *supervised*, das Abhängigkeiten zwischen den Labeln berücksichtigt. Für alle genannten LDA-Modelle existieren noch keine Implementierungen in *Spark* oder den verwendeten Python-Bibliotheken.

Im Bereich Neuronale Netze gibt es vielseitige Möglichkeiten, an die Problemstellung dieser Arbeit heranzugehen. Neben der Optimierung und Weiterentwicklung der CNNs bieten sich weitere Netz-Architekturen an, die für ein Multi-Label Klassifikationsproblem geeignet sind. Dazu gehören *Recurrent Neural Network (RNNs)* [Elm90][LXLZ15], in denen Rückkopplungen modelliert werden und *Long Short Term Memory networks (LSTMs)* [HS97], in denen Informationen gemerkt oder vergessen werden können. Die verschiedenen Architekturen werden nicht nur getrennt voneinander behandelt, sondern können miteinander kombiniert werden [CYX+17][DCP+18].

Das vorgestellte Multi-Label Brachenklassifikationsproblem bietet ein großes Potential, das World Wide Web zum Schließen von Lücken in Knowledge Bases zu nutzen und neues Wissen zu generieren. Durch die Anwendung, Erweiterung und Skalierung der Methoden aus dem Information Retrieval und dem Machine Learning ist es möglich, die großen Mengen des unerschlossenen und verborgenen Wissens, das täglich weiter wächst, nutzbar zu machen.

Literaturverzeichnis

[Ama19] Amazon: *Amazon Web Services Dokumentiation.* `https://docs.aws.amazon.com/index.html#lang/de_de`. Version: 2019, Abruf: 14.02.2019

[Apa19] Apache: *Apache Spark 2.1.0 Dokumentation.* `https://spark.apache.org/docs/2.1.0/`. Version: 2019, Abruf: 14.02.2019

[AZ12] Aggarwal, Charu C. ; Zhai, ChengXiang: *Mining Text Data.* Springer, 2012

[BLSB04] Boutell, Matthew R. ; Luo, Jiebo ; Shen, Xipeng ; Brown, Christopher M.: Learning multi-label scene classification. In: *Pattern Recognition* 37 (2004), Nr. 9, S. 1757–1771

[BNJ03] Blei, David M. ; Ng, Andrew Y. ; Jordan, Michael I.: Latent Dirichlet Allocation. In: *Journal of Machine Learning Research* 3 (2003), S. 993–1022

[Com19] Common Crawl: *Common Crawl Webseite.* `http://commoncrawl.org/`. Version: 2019, Abruf: 14.02.2019

[Cra46] Cramér, Harald: *Mathematical Methods of Statistics.* Princeton University Press, 1946

[CYX+17] Chen, Guibin ; Ye, Deheng ; Xing, Zhenchang ; Chen, Jieshan ; Cambria, Erik: Ensemble Application of Convolutional and Recurrent Neural Networks for Multi-label Text Categorization. In: *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)* (2017), S. 2377–2383

[DBp19] DBpedia: *DBpedia Webseite.* `https://wiki.dbpedia.org/`. Version: 2019, Abruf: 14.02.2019

[DCP+18] Du, Jingcheng ; Chen, Qingyu ; Peng, Yifan ; Xiang, Yang ; Tao, Cui ; Lu, Zhiyong: ML-Net: multi-label classification of biomedical texts with deep neural networks. In: *CoRR* (2018)

[DJHM13] Daiber, Joachim ; Jakob, Max ; Hokamp, Chris ; Mendes, Pablo N.: Improving Efficiency and Accuracy in Multilingual Entity Extraction Categories and Subject Descriptors. In: *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics 2013)* (2013), S. 121–124

[Elm90] Elman, Jeffreyl: Finding Structure in Time. In: *Cognitive Science* 14 (1990), S. 179–211

[Fay96] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: The KDD Process for Extracting Useful Knowledge from Volumes of Data. In: *Communications of the ACM* 39 (1996), Nr. 11, S. 27–34

[FGM05] Finkel, Jenny R. ; Grenager, Trond ; Manning, Christopher: Incorporating non-local information into information extraction systems by gibbs sampling. In: *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)* (2005), S. 363 – 370

[For73] Forney, G. D.: The Viterbi Algorithm. In: *Proceedings of the IEEE* 61 (1973), Nr. 3, S. 268–278

[FS07] Feldman, Ronen ; Sanger, James: *The text mining handbook: advanced approaches in analyzing unstructured data.* Cambridge University Press, 2007

[GH14] Gandomi, Amir ; Haider, Murtaza: Beyond the hype : Big data concepts , methods , and analytics. In: *International Journal of Information Management* 35 (2014), Nr. 2, S. 137–144

[Goo13] Google: *word2vec.* `https://code.google.com/archive/p/word2vec/`. Version: 2013, Abruf: 14.02.2019

[GR11] Gantz, By J. ; Reinsel, David: Extracting Value from Chaos. In: *IDC iView* June (2011)

[Gri08] Grimes, Seth: Unstructured Data and the 80 Percent Rule. In: *Clarabridge, Breakthrough Analysis - Bridgepoints* (2008)

[GS04] Godbole, Shantanu ; Sarawagi, Sunita: Discriminative Methods for Multi-labeled. In: *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)* (2004), S. 22–30

[HBB10] Hoffman, MD ; Blei, DM ; Bach, Francis: Online learning for latent dirichlet allocation. In: *advances in neural information processing systems*. 2010, S. 856–864

[HS97] Hochreiter, Sepp ; Schmidhuber, Jürgen: LONG SHORT-TERM MEMORY. In: *Neural Computation* Bd. 9. 1997, S. 1735–1780

[HTF08] Hastie, Trevor ; Tibshirani, Robert ; Friedman, Jerome: *The Elements of Statistical Learning*. Second. Springer, 2008

[HYB+11] Hoffart, Johannes ; Yosef, Mohamed A. ; Bordino, Ilaria ; Fürstenau, Hagen ; Pinkal, Manfred ; Spaniol, Marc ; Taneva, Bilyana ; Thater, Stefan ; Weikum, Gerhard: Robust Disambiguation of Named Entities in Text. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (2011), S. 782–792

[Ins18a] Institut für Angewandte Informatik e.V.: *DBpedia Spotlight Github*. `https://github.com/dbpedia-spotlight/dbpedia-spotlight-model`. Version: 2018, Abruf: 14.02.2019

[Ins18b] Institut für Angewandte Informatik e.V.: *DBpedia Spotlight Homepage*. `https://www.dbpedia-spotlight.org/`. Version: 2018, Abruf: 14.02.2019

[JWHT13] James, Gareth ; Witten, Daniela ; Hastie, Trevor ; Tibshirani, Robert: *An Introduction to Statistical Learning*. Springer, 2013

[Kim14] Kim, Yoon: Convolutional Neural Networks for Sentence Classification. (2014)

[Kre15a] Kreymer, Ilya: *Common Crawl Index*. `http://commoncrawl.org/2015/04/announcing-the-common-crawl-index/`. Version: 2015, Abruf: 14.02.2019

[Kre15b] Kreymer, Ilya: *Common Crawl Index API Dokumentation*. `https://github.com/webrecorder/pywb/wiki/CDX-Server-API#api-reference`. Version: 2015, Abruf: 14.02.2019

[LBS+16] Lample, Guillaume ; Ballesteros, Miguel ; Subramanian, Sandeep ; Kawakami, Kazuya ; Dyer, Chris: Neural Architectures for Named Entity Recognition. (2016)

[LCWY17] Liu, Jingzhou ; Chang, Wei-Cheng ; Wu, Yuexin ; Yang, Yiming: Deep Learning for Extreme Multi-label Text Classification. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017), S. 115–124

[LIJ+15] Lehmann, Jens ; Isele, Robert ; Jakob, Max ; Jentzsch, Anja ; Kontokostas, Dimitris ; Mendes, Pablo N. ; Hellmann, Sebastian ; Morsey, Mohamed ; Van Kleef, Patrick ; Auer, Sören ; Bizer, Christian: DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. In: *Semantic Web* 6 (2015), Nr. 2, S. 167–195

[LMP01] Lafferty, John ; McCallum, Andrew ; Pereira, Fernando C N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the Eighteenth International Conference on Machine Learning* 8 (2001), Nr. June, S. 282–289

[LXLZ15] Lai, Siwei ; Xu, Liheng ; Liu, Kang ; Zhao, Jun: Recurrent Convolutional Neural Networks for Text Classification. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015), S. 2267–2273

[MCCD13] Mikolov, Tomas ; Corrado, Greg ; Chen, Kai ; Dean, Jeffrey: Efficient Estimation of Word Representations in Vector Space. In: *ICLR Workshop Papers* (2013)

[MJGsB11] Mendes, Pablo N. ; Jakob, Max ; García-silva, Andrés ; Bizer, Christian: DBpedia Spotlight : Shedding Light on the Web of Documents. In: *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics).* 95 (2011), S. 1–8

[MMMK03] Matsugu, Masakazu ; Mori, Katsuhiko ; Mitari, Yusuke ; Kaneda, Yuji: Subject independent facial expression recognition with robust face detection using a convolutional neural network. In: *Neural Networks* 16 (2003), Nr. 5-6, S. 555–559

[MRS09] Manning, Christopher D. ; Ragahvan, Prabhakar ; Schutze, Hinrich: *An Introduction to Information Retrieval*. Cambridge University Press, 2009

[MSC+13] Mikolov, Tomas ; Sutskever, Ilya ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: Distributed Representations of Words and Phrases and their Compositionality. In: *Advances in neural information processing systems*. 2013, S. 3111–3119

[Pyw19] Pywb: *Common Crawl Index Server*. `http://index.commoncrawl.org/`. Version: 2019, Abruf: 14.02.2019

[RCSS12] Rubin, Timothy N. ; Chambers, America ; Smyth, Padhraic ; Steyvers, Mark: Statistical topic models for multi-label document classification. In: *Machine Learning* 88 (2012), S. 157–208

[RHNM09] Ramage, Daniel ; Hall, David ; Nallapati, Ramesh ; Manning, Christopher D.: Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* (2009), Nr. August, S. 248–256

[RPHF11] Read, Jesse ; Pfahringer, Bernhard ; Holmes, Geoff ; Frank, Eibe: Classifier chains for multi-label classification. In: *Machine Learning* 85 (2011), S. 333–359

[She00] Shearer, Colin: The CRISP-DM Model: The New Blueprint for Data Mining. In: *Journal of Data Warehousing* 5 (2000), Nr. 4, S. 13–22

[Sor10] Sorower, Mohammad: A Literature Survey on Algorithms for Multi-label Learning. In: *Oregon State University, Corvallis* 18 (2010)

[SS00] Schapire, Robert E. ; Singer, Yoram: BoosTexter : A Boosting-based System for Text Categorization. In: *Machine Learning* 39 (2000), S. 135–168

[Sta18] Stanford: *Stanford Named Entity Recognizer Dokumentation*. `https://nlp.stanford.edu/software/CRF-NER.html`. Version: 2018, Abruf: 14.02.2019

[Sta19] Stanford: *Stanford CoreNLP Homepage*. `https://stanfordnlp.github.io/CoreNLP/index.html`. Version: 2019, Abruf: 14.02.2019

[SWH15] Shen, Wei ; Wang, Jianyong ; Han, Jiawei: Entity linking with a knowledge base: Issues, techniques, and solutions. In: *IEEE Transactions on Knowledge and Data Engineering* 27 (2015), Nr. 2, S. 443–460

[TKV09] Tsoumakas, Grigorios ; Katakis, Ioannis ; Vlahavas, Ioannis: Mining Multi-label Data. In: *Data mining and knowledge discovery handbook*. Boston : Springer, 2009, S. 667–685

[Uni18] United States Census Bureau: *2016 SUSB Annual Data Tables by Establishment Industry*. `https://www.census.gov/data/tables/2016/econ/susb/2016-susb-annual.html`. Version: 2018, Abruf: 14.02.2019

[W3C13] W3C: *SPARQL Dokumentation*. `https://www.w3.org/TR/sparql11-query/`. Version: 2013, Abruf: 14.02.2019

[W3C14] W3C: *RDF Dokumentation*. `https://www.w3.org/TR/rdf11-concepts/`. Version: 2014, Abruf: 14.02.2019

[Yan99] Yang, Yiming: An Evaluation of Statistical Approaches to Text Categorization. In: *Journal of Information Retrieval* 1 (1999), Nr. 1-2, S. 69–90

[ZC18] Zheng, Alice ; Casari, Amanda: *Feature Engineering for Machine Learning*. O'Reilly, 2018

[ZJXG05] Zhu, Shenghuo ; Ji, Xiang ; Xu, Wei ; Gong, Yihong: Multi-labelled Classification Using Maximum Entropy Method. In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in Information Retrieval* (2005), S. 274–281

[Zur92] Zurada, J.M.: Introduction to artificial neural systems. (1992)

[ZZ14] Zhang, Min-ling ; Zhou, Zhi-hua: A Review on Multi-Label Learning Algorithms. In: *IEEE Transactions on Knowledge and Data Engineering* 26 (2014), Nr. 8, S. 1819–1837

A Übersicht der Daten

In diesem Anhang befinden sich alle Ergänzungen des Kapitels 3. Dazu gehören Ausschnitte der CC-Rohformate in Abschnitt A.1, die Verteilungen der DBpedia-Branchen in Abschnitt A.2 und die Browser-Darstellung einer Firma in DBpedia in Abschnitt A.3.

A.1 Common-Crawl-Rohformate

Ein Ausschnitt aus den CC-Rohformaten WARC, WAT und WET, sind in den Abbildungen A.1, A.2 und A.3 zu sehen. Sie dienen als Ergänzungen der Beschreibungen aus Kaptiel 3.1.1.

```
WARC/1.0
WARC-Type: response
WARC-Date: 2014-08-02T09:52:13Z
WARC-Record-ID:
Content-Length: 43428
Content-Type: application/http; msgtype=response
WARC-Warcinfo-ID:
WARC-Concurrent-To:
WARC-IP-Address: 212.58.244.61
WARC-Target-URI: http://news.bbc.co.uk/2/hi/africa/3414345.stm
WARC-Payload-Digest: sha1:M63W6MNGFDWXDSLTHF7GWUPCJUH4JK3J
WARC-Block-Digest: sha1:YHKQUSBOS4CLYFEKQDVGJ457OAPD6IJO
WARC-Truncated: length

HTTP/1.1 200 OK
Server: Apache
Vary: X-CDN
Cache-Control: max-age=0
Content-Type: text/html
Date: Sat, 02 Aug 2014 09:52:13 GMT
Expires: Sat, 02 Aug 2014 09:52:13 GMT
Connection: close
Set-Cookie: BBC-UID=...; expires=Sun, 02-Aug-15 09:52:13 GMT; path=/; domain=bbc.co.uk;

<!doctype html public "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
<title>
        BBC NEWS | Africa | Namibia braces for Nujoma exit
</title>
...
```

Abbildung A.1: Ausschnitt des WARC-Datenformates [Com19].

```
Envelope
 WARC-Header-Metadata
  WARC-Target-URI [string]
  WARC-Type [string]
  WARC-Date [datetime string]
  ...
 Payload-Metadata
  HTTP-Response-Metadata
   Headers
    Content-Language
    Content-Encoding
    ...
   HTML-Metadata
    Head
     Title [string]
     Link [list]
     Metas [list]
    Links [list]
   Headers-Length [int]
   Entity-Length [int]
   ...
  ...
 ...
Container
 Gzip-Metadata [object]
 Compressed [boolean]
 Offset [int]
```

Abbildung A.2: Grundgerüst des WAT-Datenformates [Com19].

```
WARC/1.0
WARC-Type: conversion
WARC-Target-URI: http://news.bbc.co.uk/2/hi/africa/3414345.stm
WARC-Date: 2014-08-02T09:52:13Z
WARC-Record-ID:
WARC-Refers-To:
WARC-Block-Digest: sha1:JROHLCS5SKMBR6XY46WXREW7RXM64EJC
Content-Type: text/plain
Content-Length: 6724

BBC NEWS | Africa | Namibia braces for Nujoma exit
...
President Sam Nujoma works in very pleasant surroundings in the small but beautiful old
State House...
```

Abbildung A.3: Ausschnitt des WET-Datenformates [Com19].

A.2 Verteilungen der DBpedia-Branchen

Die in diesem Abschnitt enthaltenen Abbildungen und Tabellen ergänzen die Exploration der DBpedia-Daten aus Kapitel 3.2.3.

Branche	Anzahl
Air_Transport	3927
Retail	1467
Financial_services	1235
Video_game_industry	1029
Aerospace	721
Financial_Services	707
Telecommunications	703
Software	682
Banking	653
Telecommunication	639
Automotive_industry	589
Manufacturing	521
Computer_software	485
Internet	472
Entertainment	421
Mass_media	392
Alcoholic_beverage	370
Automotive	361
Computer_and_video_game_industry	358
Insurance	347
Construction	343
Restaurant	325
Finance	320
Publishing	306
Conglomerate_(company)	298
Engineering	282
Electronics	279
Interactive_entertainment	247
Biotechnology	242
Mining	236
Transport	234
Fashion	230
Shipbuilding	228
Technology	220
Real_estate	218
Finance_and_Insurance	210
Oil_and_gas_industry	206
Retailing	205
Automobile	200
Consumer_electronics	200
Advertising	192
Film_industry	192
Hospitality	187
Healthcare	185
Private_equity	182

Branche	Anzahl
Computer_hardware	180
Arms_industry	174
Restaurants	174
Food	170
Information_technology	166
Motion_picture	161
Film	161
Television	158
Tourism	158
Education	157
Transportation	156
Energy	156
Rail_transport	149
Private_Equity	148
Music	146
Information_Technology	143
Health_care	143
Shipping	139
Logistics	134
Marketing	134
Musical_instruments	133
Food_processing	133
Real_Estate	133
Hotel	128
Petroleum_industry	126
Pharmaceutical_industry	125
Animation	123
Retailer	120
Bank	120
Broadcasting	118
Travel	113
Investment	112
Architecture	110
E-commerce	109
Apparel	107
Clothing	105
Electricity	104
Investment_management	101
Chemical_industry	101
Agriculture	100

Abbildung A.4: Verteilung der verwendeten dbo:industry-Branchen.

Branche	Anzahl
Retail	328
Manufacturing	193
Financial services	153
Entertainment	135
Fashion	128
Telecommunications	123
Media	122
Internet	121
Software	88
Education	82
Healthcare	80
Energy	80
Shipping	77
Automotive	75
Technology	71
Banking	71
Food	69
Finance	67
Engineering	65
Restaurants	64
Mining	63
Transportation	62
Electronics	62
Film	62
Music	59
Publishing	59
Financial Services	55
Beverages	50

Branche	Anzahl
Hospitality	39
Aerospace	38
Clothing	38
Travel	38
Construction	37
Aviation	37
Steel	37
Transport	35
Computer software	34
Television	33
Retailing	32
Restaurant	32
Agriculture	31
Insurance	30
Computer Software	29
Passenger transportation	29
Broadcasting	27
Biotechnology	27
Communications	26
Housebuilding	26
Video games	24
Advertising	22
Shipbuilding	22
Information technology	22
Information Technology	21
Mixed martial arts promotion	20
Television production	20
Real Estate	20
Electricity	20

Abbildung A.5: Verteilung der verwendeten *dbp:industry*-Branchen.

Branche	Anzahl
Air_Transport	3927
Retail	1795
Financial_services	1235
Video_game_industry	1029
Telecommunications	826
Software	770
Aerospace	759
Banking	724
Manufacturing	714
Financial_Services	707
Telecommunication	639
Internet	593
Automotive_industry	589
Entertainment	556
Computer_software	485
Automotive	436
Mass_media	392
Finance	387
Construction	380
Insurance	377
Alcoholic_beverage	370
Publishing	365
Computer_and_video_game_industry	358
Fashion	358
Restaurant	357
Engineering	347
Electronics	341
Mining	299
Conglomerate_(company)	298
Technology	291
Transport	269
Biotechnology	269
Healthcare	265
Shipbuilding	250
Interactive_entertainment	247
Education	239
Food	239
Restaurants	238
Retailing	237
Energy	236
Hospitality	226
Film	223
Transportation	218
Real_estate	218
Shipping	216
Advertising	214
Finance_and_Insurance	210
Oil_and_gas_industry	206
Music	205
Consumer_electronics	200
Automobile	200

Branche	Anzahl
Film_industry	192
Television	191
Private_equity	182
Computer_hardware	180
Arms_industry	174
Information_technology	166
Motion_picture	161
Tourism	158
Financial services	153
Travel	151
Rail_transport	149
Private_Equity	148
Broadcasting	145
Clothing	143
Health_care	143
Information_Technology	143
Marketing	134
Logistics	134
Real_Estate	133
Musical_instruments	133
Food_processing	133
Agriculture	131
Hotel	128
Petroleum_industry	126
Pharmaceutical_industry	125
Electricity	124
Animation	123
Media	122
Retailer	120
Bank	120
Investment	112
Architecture	110
E-commerce	109
Apparel	107
Chemical_industry	101
Investment_management	101
Financial Services	55
Beverages	50
Aviation	37
Steel	37
Computer software	34
Passenger transportation	29
Computer Software	29
Communications	26
Housebuilding	26
Video games	24
Information technology	22
Information Technology	21
Television production	20
Real Estate	20
Mixed martial arts promotion	20

Abbildung A.6: Verteilung der verwendeten DBpedia-Branchen (*dbo:industry* und *dbp:industry* kombiniert).

# **Branchen**	***dbo:industry***	***dbp:industry***	***kombiniert***
1	34929	10846	**45775**
2	4877	197	**5074**
3	1292	65	**1357**
4	465	19	**484**
5	208	8	**216**
6	89	4	**93**
7	46	0	**46**
8	19	0	**19**
9	11	0	**11**
10	4	0	**4**
11	6	0	**6**
12	3	0	**3**
13	3	0	**3**
15	1	0	**1**
20	1	0	**1**

Tabelle A.1: Branchen pro Unternehmen gegliedert nach *dbo:industry*, *dbp:industry* und beide Properties kombiniert.

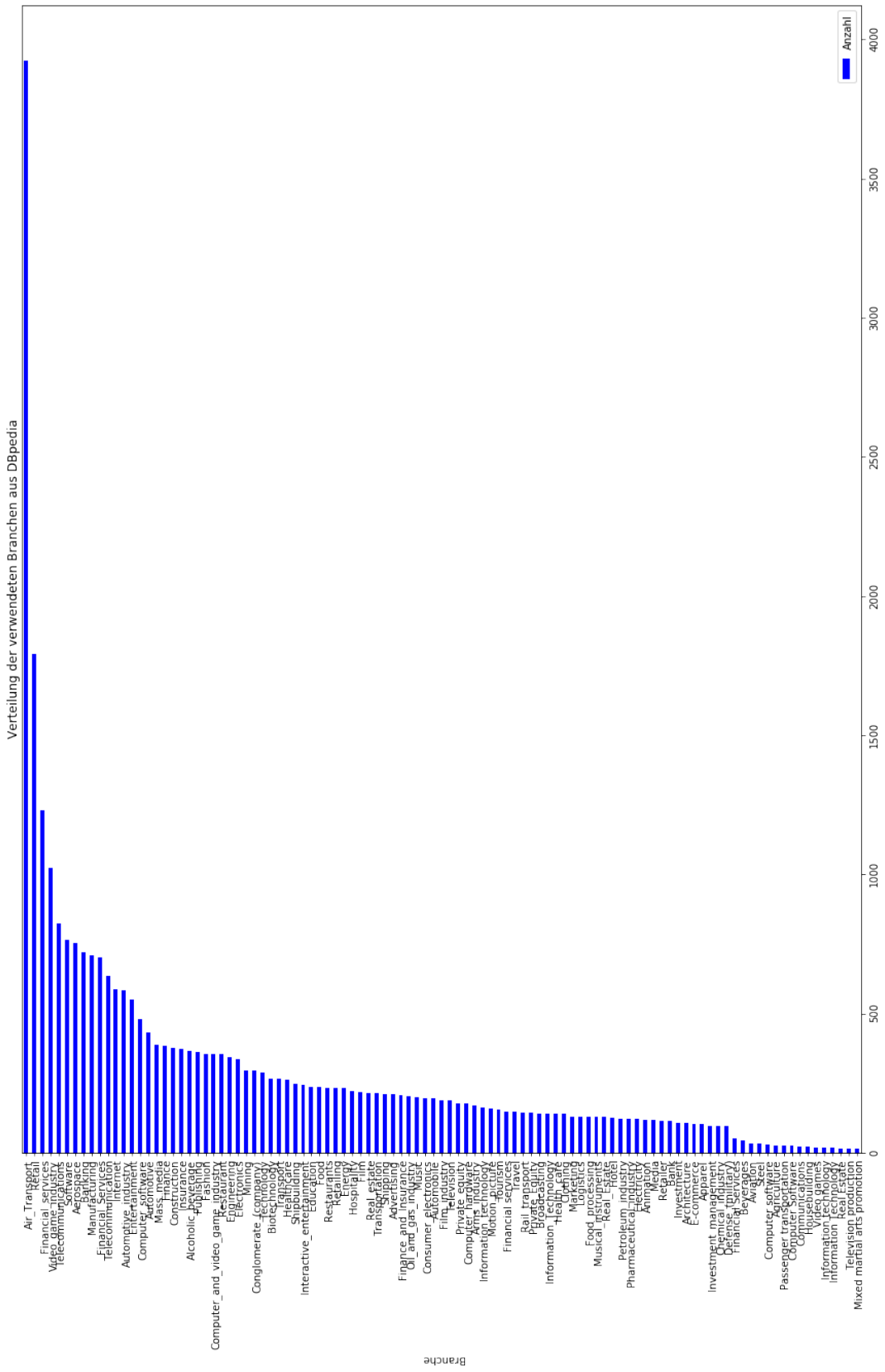
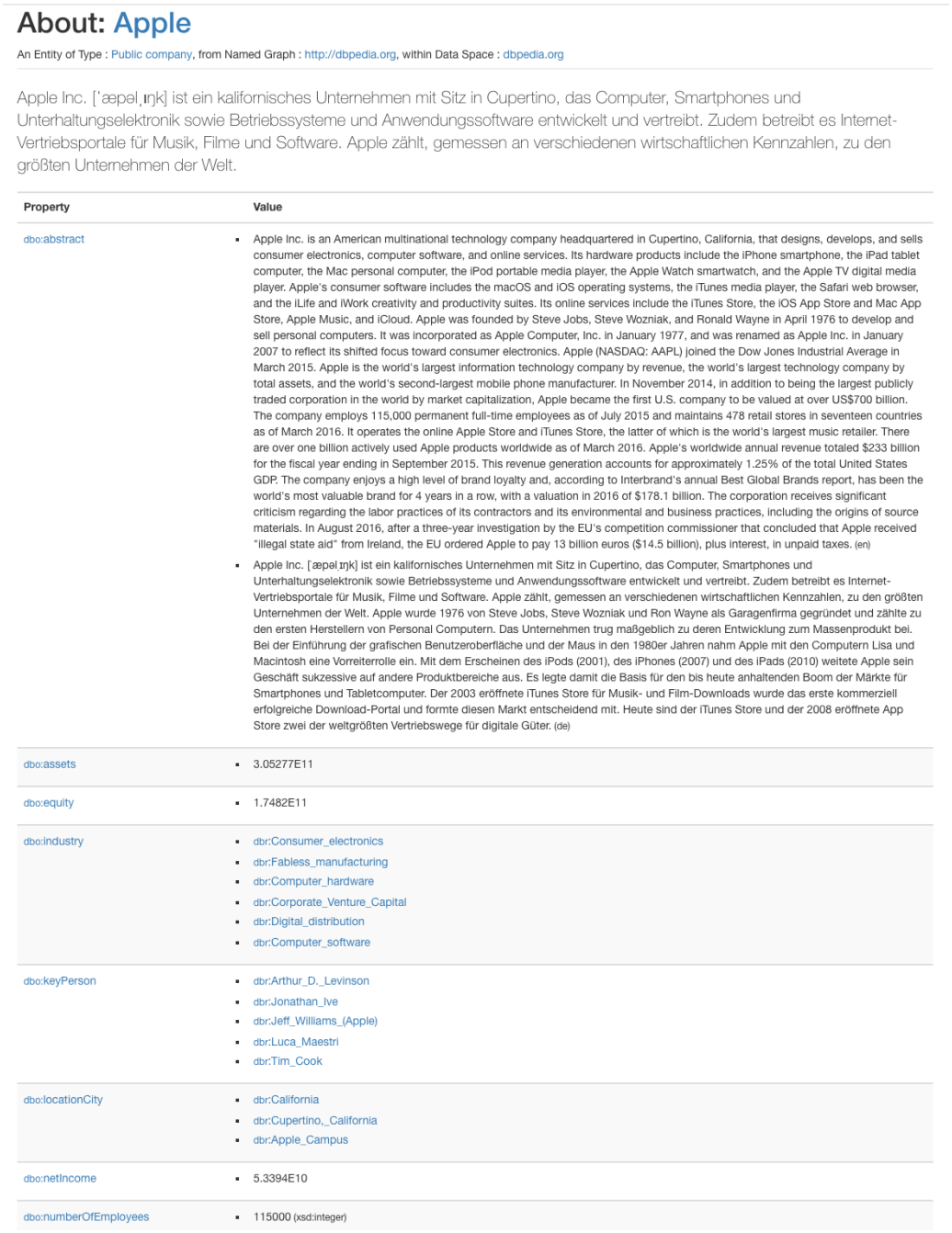


Abbildung A.7: Plot der Verteilung der verwendeten DBpedia-Branchen.

A.3 Darstellung Common-Crawl-Objekt

Die DBpedia-Darstellung der Firma *Apple Inc.* in einem HTML-Browser ist in Abbildung A.8 zu sehen. Es handelt sich dabei um eine der in Kapitel 3.2.2 genannten Zugriffsmöglichkeiten auf die Inhalte von DBpedia.



About: Apple

An Entity of Type : Public company, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

Apple Inc. ['æpəl ɪŋk] ist ein kalifornisches Unternehmen mit Sitz in Cupertino, das Computer, Smartphones und Unterhaltungselektronik sowie Betriebssysteme und Anwendungssoftware entwickelt und vertreibt. Zudem betreibt es Internet-Vertriebsportale für Musik, Filme und Software. Apple zählt, gemessen an verschiedenen wirtschaftlichen Kennzahlen, zu den größten Unternehmen der Welt.

Property	Value
dbo:abstract	▪ Apple Inc. is an American multinational technology company headquartered in Cupertino, California, that designs, develops, and sells consumer electronics, computer software, and online services. Its hardware products include the iPhone smartphone, the iPad tablet computer, the Mac personal computer, the iPod portable media player, the Apple Watch smartwatch, and the Apple TV digital media player. Apple's consumer software includes the macOS and iOS operating systems, the iTunes media player, the Safari web browser, and the iLife and iWork creativity and productivity suites. Its online services include the iTunes Store, the iOS App Store and Mac App Store, Apple Music, and iCloud. Apple was founded by Steve Jobs, Steve Wozniak, and Ronald Wayne in April 1976 to develop and sell personal computers. It was incorporated as Apple Computer, Inc. in January 1977, and was renamed as Apple Inc. in January 2007 to reflect its shifted focus toward consumer electronics. Apple (NASDAQ: AAPL) joined the Dow Jones Industrial Average in March 2015. Apple is the world's largest information technology company by revenue, the world's largest technology company by total assets, and the world's second-largest mobile phone manufacturer. In November 2014, in addition to being the largest publicly traded corporation in the world by market capitalization, Apple became the first U.S. company to be valued at over US$700 billion. The company employs 115,000 permanent full-time employees as of July 2015 and maintains 478 retail stores in seventeen countries as of March 2016. It operates the online Apple Store and iTunes Store, the latter of which is the world's largest music retailer. There are over one billion actively used Apple products worldwide as of March 2016. Apple's worldwide annual revenue totaled $233 billion for the fiscal year ending in September 2015. This revenue generation accounts for approximately 1.25% of the total United States GDP. The company enjoys a high level of brand loyalty and, according to Interbrand's annual Best Global Brands report, has been the world's most valuable brand for 4 years in a row, with a valuation in 2016 of $178.1 billion. The corporation receives significant criticism regarding the labor practices of its contractors and its environmental and business practices, including the origins of source materials. In August 2016, after a three-year investigation by the EU's competition commissioner that concluded that Apple received "illegal state aid" from Ireland, the EU ordered Apple to pay 13 billion euros ($14.5 billion), plus interest, in unpaid taxes. (en) ▪ Apple Inc. ['æpəl ɪŋk] ist ein kalifornisches Unternehmen mit Sitz in Cupertino, das Computer, Smartphones und Unterhaltungselektronik sowie Betriebssysteme und Anwendungssoftware entwickelt und vertreibt. Zudem betreibt es Internet-Vertriebsportale für Musik, Filme und Software. Apple zählt, gemessen an verschiedenen wirtschaftlichen Kennzahlen, zu den größten Unternehmen der Welt. Apple wurde 1976 von Steve Jobs, Steve Wozniak und Ron Wayne als Garagenfirma gegründet und zählte zu den ersten Herstellern von Personal Computern. Das Unternehmen trug maßgeblich zu deren Entwicklung zum Massenprodukt bei. Bei der Einführung der grafischen Benutzeroberfläche und der Maus in den 1980er Jahren nahm Apple mit den Computern Lisa und Macintosh eine Vorreiterrolle ein. Mit dem Erscheinen des iPods (2001), des iPhones (2007) und des iPads (2010) weitete Apple sein Geschäft sukzessive auf andere Produktbereiche aus. Es legte damit die Basis für den bis heute anhaltenden Boom der Märkte für Smartphones und Tabletcomputer. Der 2003 eröffnete iTunes Store für Musik- und Film-Downloads wurde das erste kommerziell erfolgreiche Download-Portal und formte diesen Markt entscheidend mit. Heute sind der iTunes Store und der 2008 eröffnete App Store zwei der weltgrößten Vertriebswege für digitale Güter. (de)
dbo:assets	▪ 3.05277E11
dbo:equity	▪ 1.7482E11
dbo:industry	▪ dbr:Consumer_electronics ▪ dbr:Fabless_manufacturing ▪ dbr:Computer_hardware ▪ dbr:Corporate_Venture_Capital ▪ dbr:Digital_distribution ▪ dbr:Computer_software
dbo:keyPerson	▪ dbr:Arthur_D._Levinson ▪ dbr:Jonathan_Ive ▪ dbr:Jeff_Williams_(Apple) ▪ dbr:Luca_Maestri ▪ dbr:Tim_Cook
dbo:locationCity	▪ dbr:California ▪ dbr:Cupertino,_California ▪ dbr:Apple_Campus
dbo:netIncome	▪ 5.3394E10
dbo:numberOfEmployees	▪ 115000 (xsd:integer)

Abbildung A.8: Ausschnitt der Darstellung von Apple in DBpedia (Zugriff per HTML-Browser über `http://dbpedia.org/page/Apple_Inc.`).

B Aufbau

In den folgenden Abschnitten befinden sich ergänzende Informationen zu den Inhalten aus Kapitel 4. Diese sind eine Übersicht zu den verwendeten Nachrichtenseiten aus dem CC-Datensatz, die sich in Abschnitt B.1 befindet. Des Weiteren ist in Abschnitt B.2 eine Übersicht zur Abbildung der berücksichtigten DBpedia-Klassen auf die verwendete Klassenstruktur des Klassifikationsproblems zu sehen. Zuletzt sind in Abschnitt B.3 die Parameter des verwendeten CNNs dem Referenz-CNN gegenübergestellt.

B.1 Verwendete Nachrichtenseiten aus Common Crawl

Domain-Such-String	**Anzahl Treffer in Crawl**
https://www.bloomberg.com/news/articles/*	14
http://www.marketwatch.com/story/*	12.285
https://www.stockwatch.com/News/*	613
https://www.thenewswire.com/archives/*	132
https://www.morningstar.com/articles/*	393
https://www.newsfilecorp.com/release/*	165
https://www.fscwire.com/newsrelease/*	49
http://www.marketwired.com/press-release/*	13.908
http://www.globenewswire.com/news-release/*	10.691
https://www.reuters.com/article/*	6
https://www.cnbc.com/2018/*	9.790
https://www.bbc.com/news/*	7.725
https://www.nytimes.com/2018/*	13.492
https://www.washingtonpost.com/national/*	4.138
https://www.washingtonpost.com/world/*	9.311
https://www.washingtonpost.com/business/*	4.081
https://www.washingtonpost.com/local/*	7.160
https://www.washingtonpost.com/entertainment/*	2.699
https://usatoday.com/story/*	10.653
https://www.wsj.com/articles/*	12.582
https://nypost.com/2018/*	5.161

Tabelle B.1: Verwendete Such-Domains mit Anzahl der Treffer in der Indexsuche für den November-2018-Crawl.

B.2 Klassenmapping

Das Mapping der 102 verwendeten DBpedia-Branchen auf die 24 Marketwired-Branchen ist in Abbildung B.1 zu sehen. Zusätzlich werden die beiden Klassen *CONGLOMERATE* und *MISC* eingeführt, wie in Kapitel 4.4.1 beschrieben.

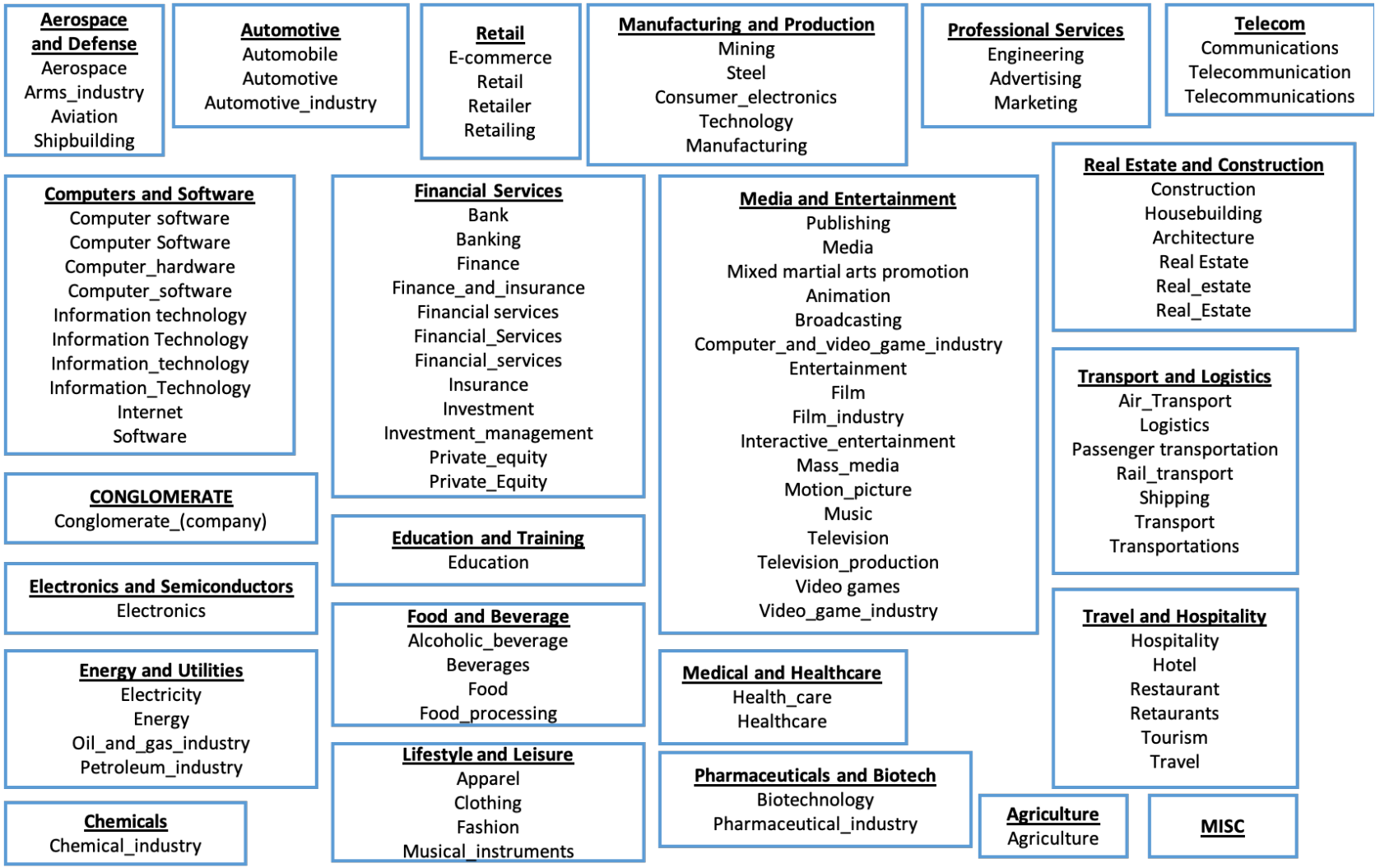


Abbildung B.1: Grafische Übersicht des Klassenmappings von den DBpedia-Branchen auf die angepassten Marketwired-Branchen.

B.3 Parameter Convolutional Neural Network

In Tabelle B.2 sind die Parameter des Referenz-CNNs zur Textklassifikation von Kim [Kim14], dem verwendeten CNN dieser Arbeit (siehe Kapitel 4.4.5, S. 63 und Kapitel 5.3, S. 79), gegenübergestellt.

Parameter	**CNN Kim**	**Verwendetes CNN**
Embeddings	Google-News-Embeddings	Google-News-Embeddings
Fensterbreite Filter	(3,4,5)	3
Anzahl Filter	100	100
Aktivierungsfunktion Convolutional Layer	ReLU	ReLU
Pooling	1-max pooling	2-max pooling
Dropout Rate	0.5	0.5
l2 Constraint	3.0	-
Batch Size	50	50
Aktivierungsfunktion Fully-Connected-Layer	softmax	sigmoid

Tabelle B.2: Verwendete Parameter des CNNs den Parametern von Kim's CNN [Kim14] gegenübergestellt.

C Auswertung

In diesem Anhang sind die Ergänzungen des Kapitels 6 enthalten. Zu diesen zählen die in Abschnitt C.1 aufgelisteten Tabellen der Branchenverteilungen in den Trainings- und Testdaten. In Abschnitt C.2 wird die Dimensionalität der Feature-Vektoren für die verschiedenen Datensätze ergänzt. Außerdem sind in Abschnitt C.3 die Ergebnisse einzelner Modelle aufgeführt, in denen die Metriken für die einzelnen Branchen zu sehen sind.

C.1 Branchenverteilung in Trainings- und Testdaten

In den Tabellen C.1 und C.2 sind die genauen Zahlen der Verteilungen der Branchen in den Trainings- und Testdaten aufgelistet. Die Visualisierung dieser Daten sind in Kapitel 6.2 in den Abbildungen 6.7 und 6.8 zu sehen.

Branche	Anzahl Dokumente (niedrige Konfidenz)	Anzahl Dokumente (hohe Konfidenz)
Aerospace and Defense	3302	2152
Agriculture	98	43
Automotive	5173	4069
CONGLOMERATE	4301	3377
Chemicals	694	621
Computers and Software	25281	23378
Education and Training	132	77
Electronics and Semiconductors	930	822
Energy and Utilities	2485	2309
Environment	0	0
Financial Services	15942	13465
Food and Beverage	1510	1259
Government	0	0
Lifestyle and Leisure	1339	905
MISC	64383	55605
Manufacturing and Production	8306	8096
Media and Entertainment	18372	17119
Medical and Healthcare	1236	869
Pharmaceuticals and Biotech	1917	1771
Professional Services	657	520
Real Estate and Construction	836	740
Retail	6183	4677
Sports	0	0
Telecom	4487	4014
Transportation and Logistics	4166	3425
Travel and Hospitality	2385	1880

Tabelle C.1: Verteilung der Branchen in den Dokumenten.

Branche	Anzahl Paragraphen (niedrige Konfidenz)	Anzahl Paragraphen (hohe Konfidenz)
Aerospace and Defense	5746	3781
Agriculture	171	67
Automotive	14498	11540
CONGLOMERATE	8443	6396
Chemicals	1281	1129
Computers and Software	61985	55835
Education and Training	204	136
Electronics and Semiconductors	2320	2045
Energy and Utilities	6794	6257
Environment	0	0
Financial Services	34938	28367
Food and Beverage	4039	3293
Government	0	0
Lifestyle and Leisure	2461	1701
MISC	191345	152610
Manufacturing and Production	20076	19165
Media and Entertainment	37356	34444
Medical and Healthcare	2258	1569
Pharmaceuticals and Biotech	5691	4936
Professional Services	1303	1039
Real Estate and Construction	1735	1468
Retail	16427	13129
Sports	0	0
Telecom	11108	9864
Transportation and Logistics	10604	8397
Travel and Hospitality	5215	4116

Tabelle C.2: Verteilung der Branchen in den Paragraphen.

C.2 Feature-Dimensionen

In Tabelle C.3 wird die Dimensionalität der Feature-Vektoren für die in Kapitel 6.2 verwendeten Datensätze ergänzt. Dabei ist zu sehen, dass mit der Reduzierung der Datenmenge eine gleichzeitige Reduzierung der Feature-Dimensionalität einhergeht. Die Texte werden durch das Ausschließen der Klasse *MISC*, die Entfernung irrelevanter Paragraphen und der Verwendung eines höheren Konfindenz-Parameters beim EL reduziert.

Feature-Vektor	**Dokumente**	**Dokumente ohne MISC**	**Paragraphen ohne MISC**	**Paragraphen ohne MISC (hohe Konfidenz)**
BOW:	3.624.698	2.927.096	810.349	709.480
Tf-idf:	3.624.698	2.927.096	810.349	709.480
LDA:	23	23	23	23

Tabelle C.3: Dimensionen der Feature-Vektoren aus den verwendeten Datensätzen.

C.3 Modelluntersuchung

Ausgewählte Modelle der Modellauswertung aus Kapitel 6.2 werden auf die Ergebnisse der einzelnen Branchen hin untersucht.

Das erste Modell in Abbildung C.1 ist das der logistischen Regression, das auf den Dokumenten-Daten der niedrigen Konfidenz (Konfidenzparameter: 0,5) unter Verwendung der BOW-Vektoren trainiert wird. Es fällt auf, dass die Branchen umso besser erkannt werden, je häufiger sie in den Trainingsdaten vorkommen. Die Klasse *MISC*, die am häufigsten in den Daten vertreten ist, wird am besten erkannt. Dies führt zu hohen Werten in den Micro-Metriken. Während die Precision für alle Branchen Werte $> 0{,}7$ annimmt, liegen die Recall-Werte, insbesondere in den selten vertretenen Branchen, unterhalb von 0,7 oder sogar unter 0,5. Daher erreicht die Macro-Recall-Metrik lediglich einen Wert von 0,57.

	precision	recall	f1-score	support
Aerospace and Defense	0.83	0.35	0.49	658
Agriculture	0.82	0.56	0.67	16
Automotive	0.83	0.50	0.63	1050
CONGLOMERATE	0.81	0.52	0.63	891
Chemicals	0.87	0.43	0.58	138
Computers and Software	0.95	0.87	0.91	5021
Education and Training	0.90	0.41	0.56	22
Electronics and Semiconductors	0.92	0.40	0.56	176
Energy and Utilities	0.87	0.63	0.73	481
Financial Services	0.85	0.73	0.79	3160
Food and Beverage	0.89	0.49	0.63	319
Lifestyle and Leisure	0.77	0.41	0.54	270
MISC	0.91	0.94	0.92	12950
Manufacturing and Production	0.89	0.70	0.78	1641
Media and Entertainment	0.91	0.80	0.86	3633
Medical and Healthcare	0.78	0.48	0.60	266
Pharmaceuticals and Biotech	0.89	0.65	0.75	400
Professional Services	0.74	0.32	0.45	125
Real Estate and Construction	0.78	0.44	0.56	163
Retail	0.85	0.60	0.70	1233
Telecom	0.80	0.56	0.66	880
Transportation and Logistics	0.89	0.72	0.79	859
Travel and Hospitality	0.79	0.53	0.63	459
micro avg	0.90	0.79	0.84	34811
macro avg	0.85	0.57	0.67	34811
weighted avg	0.89	0.79	0.83	34811
samples avg	0.87	0.82	0.82	34811

Abbildung C.1: Ergebnisse der logistischen Regression pro Label auf Dokumenten-Daten (niedrige Konfidenz) mit BOW-Features.

Die Ergebnisse der einzelnen Branchen des linearen SVM-Modells auf den Dokumenten-Daten der niedrigen Konfidenz, ohne der Klasse *MISC*, trainiert auf den BOW-Vektoren, sind in Abbildung C.2 zu sehen. Auch hier liegen die Recall-Werte bei der Mehrheit der Modelle teils deutlich hinter den Precision-Werte, wie bspw. für die Klasse *Education and Training*, deren Precision-Wert mehr als doppelt so hoch ist, wie der Recall-Wert. Tendenziell werden Klassen umso besser erkannt, je häufiger sie vertreten sind. Der niedrige Wert der Macro-Recall-Metrik zeigt, dass im Durchschnitt nur je 60% der Beobachtungen für die einzelnen Klassen erkannt werden.

	precision	recall	f1-score	support
Aerospace and Defense	0.79	0.68	0.73	618
Agriculture	0.86	0.50	0.63	24
Automotive	0.74	0.58	0.65	1025
CONGLOMERATE	0.77	0.59	0.67	862
Chemicals	0.74	0.43	0.55	129
Computers and Software	0.94	0.91	0.92	5029
Education and Training	0.75	0.36	0.49	25
Electronics and Semiconductors	0.82	0.46	0.59	189
Energy and Utilities	0.81	0.64	0.72	515
Financial Services	0.84	0.79	0.81	3192
Food and Beverage	0.83	0.57	0.68	299
Lifestyle and Leisure	0.64	0.46	0.54	266
Manufacturing and Production	0.83	0.75	0.79	1608
Media and Entertainment	0.90	0.84	0.86	3698
Medical and Healthcare	0.73	0.54	0.62	237
Pharmaceuticals and Biotech	0.85	0.71	0.77	399
Professional Services	0.69	0.35	0.46	127
Real Estate and Construction	0.83	0.49	0.62	177
Retail	0.76	0.65	0.70	1228
Telecom	0.74	0.60	0.67	861
Transportation and Logistics	0.83	0.73	0.77	798
Travel and Hospitality	0.74	0.57	0.65	431
micro avg	0.85	0.75	0.80	21737
macro avg	0.79	0.60	0.68	21737
weighted avg	0.85	0.75	0.79	21737
samples avg	0.81	0.78	0.78	21737

Abbildung C.2: Ergebnisse lineare SVM pro Label auf Dokumenten-Daten (niedrige Konfidenz) auf BOW-Features ohne der Klasse *MISC*.

Abbildung C.3 zeigt das lineare SVM-Modell auf den Paragraphen-Daten der niedrigen Konfidenz, das auf den Tf-idf-Vektoren trainiert wird. Die Klasse *MISC* wurde für die Modellbildung entfernt. Es fällt auf, dass alle Branchen außer *Agriculture* und *Education and Training* einen sehr hohen Precision-Wert $\geq 0{,}95$ aufweisen. Die Recall-Werte nehmen für 18 von 22 Klassen Werte $\leq 0{,}5$ an. Der deutlichste Unterschied zwischen Precision- und Recall-Werten ist mit 0,9 bei der Klasse *Chemicals* zu erkennen. Die beiden Klassen *Agriculture* und *Education and Training* werden vom Klassifikator nicht zugewiesen und somit nicht erkannt.

	precision	recall	f1-score	support
Aerospace and Defense	0.97	0.32	0.48	1112
Agriculture	0.00	0.00	0.00	37
Automotive	0.99	0.50	0.66	2865
CONGLOMERATE	0.95	0.26	0.41	1661
Chemicals	1.00	0.10	0.19	258
Computers and Software	1.00	0.84	0.91	12407
Education and Training	0.00	0.00	0.00	44
Electronics and Semiconductors	0.99	0.17	0.29	481
Energy and Utilities	0.98	0.34	0.51	1386
Financial Services	0.98	0.63	0.77	6970
Food and Beverage	1.00	0.24	0.39	817
Lifestyle and Leisure	1.00	0.11	0.19	498
Manufacturing and Production	0.98	0.55	0.70	3942
Media and Entertainment	0.99	0.71	0.83	7500
Medical and Healthcare	1.00	0.14	0.24	464
Pharmaceuticals and Biotech	0.99	0.38	0.55	1153
Professional Services	1.00	0.06	0.12	250
Real Estate and Construction	0.97	0.09	0.17	332
Retail	0.99	0.48	0.65	3241
Telecom	0.99	0.38	0.55	2201
Transportation and Logistics	0.99	0.51	0.67	2127
Travel and Hospitality	0.98	0.31	0.47	1037
micro avg	0.99	0.58	0.73	50783
macro avg	0.90	0.32	0.44	50783
weighted avg	0.99	0.58	0.71	50783
samples avg	0.63	0.60	0.61	50783

Abbildung C.3: Ergebnisse logistische Regression pro Label auf Paragraphen-Daten (niedrige Konfidenz) auf Tf-idf-Features ohne der Klasse *MISC*.