HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Andy Ludwig

Entwicklung und Evaluation eines wissensbasierten maschinellen Lernensembles zur Klassifikation von Buchklappentexten

2019

Fakultät **Angewandte Computer - und Biowissenschaften**

BACHELORARBEIT

Entwicklung und Evaluation eines wissensbasierten maschinellen Lernensembles zur Klassifikation von Buchklappentexten

Autor:
Andy Ludwig

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO16w2-B

Erstprüfer:
Prof. Dr. Dirk Labudde

Zweitprüfer:
M.Sc. Inf. Michael Spranger

Mittweida, August 2019

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Hintergrund der Arbeit

Der Ausgangspunkt dieser Arbeit war die Aufgabenstellung von „GermEval 2019 Task 1 – Shared task on hierarchical classification of blurbs.“[1] Die Organisatoren sind Rami Aly, Steffen Remus und Chris Biemann von der Language Technology Group der Universität Hamburg.

Jährlich werden dort neue Aufgaben veröffentlicht, die sich an Bearbeitende in Deutschland, Österreich und der Schweiz richten. Dadurch soll über Jahre hinweg die computergestützte Textverarbeitung der deutschen Sprache gefördert und verbessert werden.

Die aktuelle Aufgabe bezieht sich auf die Klassifikation von Buchklappentexten. Eine Besonderheit ist die Berücksichtigung einer Hierarchie. Jedes Buch soll in einen Zweig der Hierarchie sortiert werden, was die Aufgabe deutlich erschwert und sich somit von einer allgemeinen und flachen Textklassifikation unterscheidet.

Während der anfänglichen Auseinandersetzung mit der hierarchischen Klassifikation fiel der Fokus wiederholt auf die Verbesserung der ersten Klassifikationsebene. Die Auswahl möglicher Feature und die nötigen Vorverarbeitungsschritte wurden schnell so umfangreich, dass das Ziel der vorliegenden Arbeit die Verbesserung der flachen Klassifikation darstellte.

1.2 Motivation

Neben dem Ziel der allgemeinen Verbesserung der deutschsprachigen Textanalysen, findet die Methodik auch im forensischen Umfeld Anwendung. So sind automatisierte Analysen von sozialen Netzwerken ein verbreitetes Problem der Textanalyse. Dazu zählt beispielsweise das Finden von Leadern in Netzen und Gruppen, oder auch eine Bewertung von Stimmungen anhand von Nachrichten oder Posts. Da die Daten schnell einen enormen Umfang annehmen können, sind computergestützte Lösungen unumgänglich.

Um die Komplexität der Daten genauer strukturieren zu können, ist die hierarchische Textklassifikation auch im Feld der Forensik anwendbar. Im Zuge der Datensicherung eines Beschuldigten können die Daten so aufbereitet werden, dass hierarchische Strukturen erkennbar sind. Ein möglicher Anwendungsfall sind Firmenhierarchien, die nach einem Fall von Wirtschaftsbetrug aufgedeckt werden müssen. Bei der Analyse von Mailverkehr kann so die Datenmenge in beispielsweise Personen, Rechnungen etc. gruppiert werden und die jeweiligen Schichten anhand der Texte automatisiert bestimmt werden.

Obwohl die hier klassifizierten Daten Buchklappentexte sind, sind die Resultate nicht weniger relevant. Damit die Algorithmen und Vorverarbeitungsschritte auf Realdaten angewandt werden können, müssen sie an Testdaten erprobt und verbessert werden. Die hier klassifizierten Texte bieten einen großen Umfang und waren leicht zugänglich, was die Fokussierung auf die Modelle und Methoden vereinfacht.

Zusammenfassend ermöglicht ein fundiertes Wissen in der Textanalyse eine breite Anwendung in vielen akademischen, forensischen und auch wirtschaftlichen Belangen. Eine intensive Auseinandersetzung mit der Textverarbeitung bietet eine steile Lernkurve und bildet die

[1] Website zur Aufgabenstellung: `https://competitions.codalab.org/competitions/20139`, letzter Zugriff 12.08.2019

Grundlage für viele weitere Anwendungen, Verbesserungen und Neuerungen auf dem Gebiet der Datenanalyse.

2 Theorie

Dieses Kapitel gibt eine Übersicht über grundlegende theoretische Aspekte der Textklassifikation. Neben der allgemeinen Einführung in das Thema, folgt eine detailliertere Beschreibung der angewandten Modelle. Zusätzlich wird der Weg der Evaluation beschrieben.

2.1 Grundlagen der Textklassifikation und Feature Selection

In der folgenden Arbeit wird die Klassifikation von Texten betrachtet, dabei wird jedes Dokument einer vorher definierten Klasse zugeordnet. Bei den eingesetzten Modellen handelt es sich um überwachte (supervised) Lernverfahren. Im Gegensatz dazu steht das Clustering, bei dem die Klassen während der Modellierung anhand der Daten erstellt werden. Zu Beginn sind einige formale Regelungen zu treffen, um einen Rahmen für die Textverarbeitung zu schaffen. Jedes Dokument d_i besteht aus Wörtern w_i aus einem Vokabular $V = \{w_1, w_2, ..., w_n\}$. Ein Datensatz der mehrere Dokumente d_i beinhaltet, wird Collection $C = \{d_1, d_2, ..., d_m\}$ genannt. [1]

Die Trainingscollection besteht aus Dokumenten, die bereits den Klassen zugeordnet sind. Mit diesen Trainingsdaten wird später ein Klassifikationsmodell erstellt, mit dem anhand ausgewählter Feature ein Testdatensatz klassifiziert werden kann. [2]

Die Feature Selection stellt einen zentralen Bestandteil der Klassifikation dar. Damit die Dokumente den Klassen zugeordnet werden können, müssen hinreichend spezifische Merkmale ausgewählt werden. Im weiteren werden diese als Feature bezeichnet. Die grundlegenden Methoden, die auch in den späteren Umsetzungen Anwendung finden, sind die Kombination aus Bag-Of-Words (BOW) und Termfrequenz. Ein BOW repräsentiert das Dokument als eine Kombination der einzelnen Wörter (Terme), somit wird das Dokument als ein Vektor dargestellt, dessen Dimensionen die Terme sind. Dadurch wird die Klassenzugehörigkeit anhand des Auftretens einzelner Worte abgeschätzt. Pro Klasse besitzen die charakteristischsten Terme den höchsten Informationsgehalt.

Die einfachste Form ist ein Bitvektor. Möglich sind nur die Werte Null oder Eins. Das Feature repräsentiert somit nur eine binäre Entscheidung, ob ein Wort in einem Dokument auftritt oder nicht. Die Termfrequenz (TF) verfeinert diese Entscheidung, in dem die Anzahl der Wörter mit berücksichtig wird. Die Termfrequenz gibt an, wie oft ein Wort in einem Dokument vorkommt, formal in Gleichung (1) dargestellt. [1]

$$TF(w, d) = count(w, d) \tag{1}$$

Es existiert eine Vielzahl an weiteren, allgemeinen Features, wie zum Beispiel die Kombination aus Termfrequenz und inverser Dokumentenfrequenz (IDF). Dabei wird neben dem Vorkommen eines Wortes im Dokument, auch das Vorkommen in der gesamten Collection betrachtet. Je häufiger ein Wort in der Collection vorkommt, desto geringer kann der Einfluss auf die Klassifikation sein. Daraus ergibt sich, dass seltene Worte eine höhere Bedeutung besitzen. Gleichung (2) beschreibt die IDF formal. [1]

$$IDF(w) = log[\frac{M+1}{k}] \tag{2}$$

Dabei bezeichnet M die Anzahl aller Dokumente in der Collection und k die Anzahl der Dokumente, die das Wort w beinhalten. Die inverse Dokumentfrequenz kann als Faktor zur Termfrequenz verstanden werden.

Grundsätzlich sind Terme mit einem hohen Informationsgehalt und einer eindeutigen Klassenzugehörigkeit am bedeutendsten. Das kann jedoch nur erreicht werden, wenn ein Term mehr als einmal in der Collection auftritt. Kommt der Term dagegen nur einmal vor, ist dieser nicht spezifisch für eine Klasse. Sein Auftreten kann ebenso auf Zufall beruhen.

Es genügt weiterhin nicht, wenn ein Term in nur einem Dokument mehrfach auftritt. Viel mehr sollten klassenspezifische Terme in mehr als einem Dokument vorkommen.

Eine weitere Möglichkeit stellen n-Grams als Feature dar. Zu unterscheiden sind diese in Character n-Grams und Term n-Grams. Im ersten Fall werden nicht Wörter im Ganzen betrachtet, sondern die Abfolge von Buchstaben. Es werden die Aneinanderreihung von n aufeinanderfolgenden Buchstaben betrachtet. Somit können Schlüsse auf die Satzmorphologie gezogen werden.

Bei den Term n-Grams wird die Abfolge ganzer Worte betrachtet. Schon mit der Kombination von zwei aufeinanderfolgenden Wörtern, sogenannte Bigrams, kann die Satzstruktur vollständig wiederhergestellt werden.

Um Feature und Dokumente in Verbindung zu setzen, wird eine Dokument-Term-Matrix (DTM) als Datenstruktur verwendet. Dabei stellen die Dokumente die Zeilen und die Feature die Spalten dar. Die transponierte Version wird Term-Dokument-Matrix genannt. In Tabelle 1 ist eine solche Matrix schematisch dargestellt.

Tabelle 1: Beispiel einer Dokument-Term-Matrix

	Term 1	...	Term n
Dok 1	...	...	...
...	...	...	...
Dok n	...	...	...

Ein sehr weit verbreitetes Problem der Textklassifikation von realen Daten ist die Unausgeglichenheit der Verteilung aller Daten auf die Klassen. Wenige Klassen sind sehr häufig repräsentiert, mehrere Klassen sehr wenig. Dieses Problem wird auch unbalancierter Datensatz genannt. S. Kotsiantis, D. Kanellopoulo und P. Pintelas zeigen in dem Paper „Handling imbalanced datasets: A review“ mehrere Möglichkeiten auf, um trotz Unausgeglichenheit ein schlechtes Klassifikationsergebnis zu vermeiden [3]. Typische Herangehensweisen sind Under- bzw. Oversampling. Im ersten Fall werden die Datensätze balanciert, indem die Gesamtanzahl der Dokumente soweit verringert wird, dass dieselbe Anzahl von positiven und negativen Labels vorhanden ist. Beim Oversampling dagegen werden im einfachsten Fall die positiven Dokumente so oft vervielfacht, dass sie genauso häufig auftreten wie die negativen Dokumente. Die Vervielfältigung kann komplexer gestaltet werden, indem die Terme der gesamten Collection neu zusammengesetzt werden. Somit entstehen neue Dokumente, die strukturell verschieden sind, inhaltlich jedoch identisch. Auch das Einbinden externer Quellen ist möglich. Sind inhaltlich nahe Texte vorhanden, können neue Dokumente aus diesem Inhalt erstellt

werden. Dadurch wird das Wissen über den jeweiligen Kontext erhöht.

2.2 Grundlagen der hierarchischen Textklassifikation

An dieser Stelle werden kurz die Hintergründe der hierarchischen Textklassifikation vorgestellt, da dieses Problem die Grundlage zur Auseinandersetzung mit dem Thema darstellte.

Die hierarchischen Klassifikation ist weniger weit erforscht als klassische Klassifikationsprobleme. Somit muss das Gebiet genauer definiert werden, damit nicht falsche Sachverhalte derselben Art von Problemen zugeschrieben werden. Die Besonderheit der hierarchischen Klassifikation liegt nun darin, dass nicht nur eine einfache Zuordnung von Dokument in Klasse gesucht ist, sondern das Dokument in eine Taxonomie geordnet wird. Somit muss über mehrere Schichten die passende Klasse ermittelt werden. [4]

Eine Hierarchie ist allgemein eine Art Baumstruktur, wie exemplarisch in Abbildung 1 dargestellt.

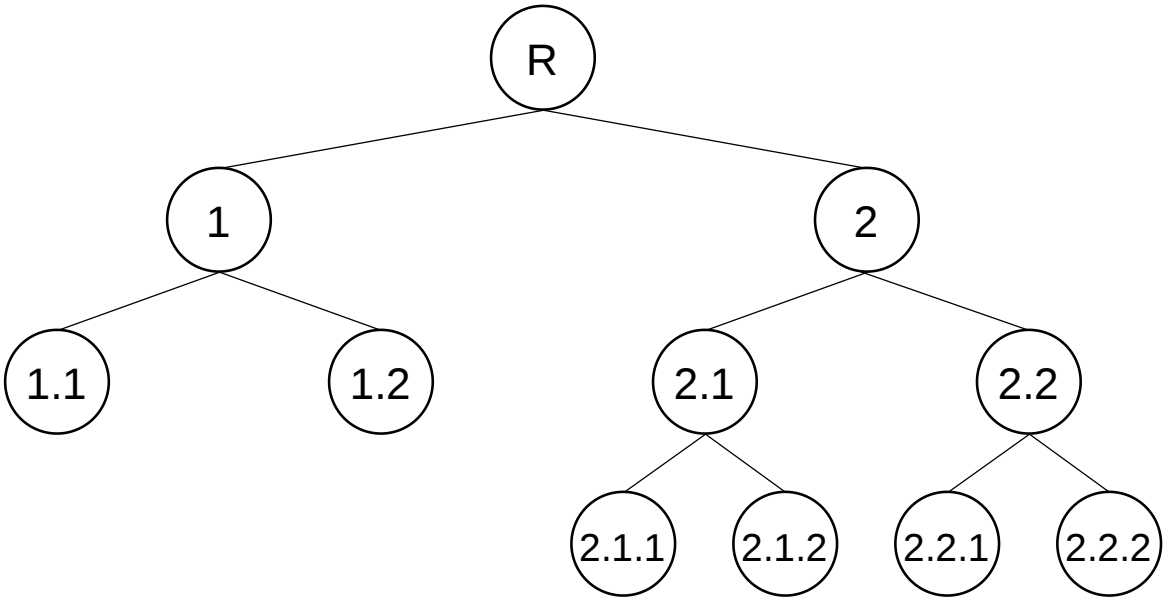

Abbildung 1: Schematische Darstellung einer Taxonomie

Dabei wird eine geordnete Menge $(C, \prec)$ angenommen, wobei C die Menge aller möglichen Klassen darstellt und $\prec$ eine „Ist-Ein"-Beziehung ist (z.B. Asymmetrie: alle Katzen sind Tiere, aber nicht alle Tiere sind Katzen). Die Ordnungsrelation ist asymmetrisch, reflexiv und transitiv und das größte Element ist stets das Wurzelelement R [5]:

1. Asymmetrie: $\forall c_i, c_j \in C : wenn\ c_i \prec c_j, dann\ c_j \not\prec c_i$
2. Reflexivität: $\forall c_i \in C : c_i \not\prec c_i$
3. Transitivität: $\forall c_i, c_j, c_k \in C : c_i \prec c_j \wedge c_j \prec c_k \rightarrow c_i \prec c_k$

2.3 Modelle zur Textklassifikation

In diesem Kapitel werden die theoretischen Aspekte der später angewandten Modelle erläutert. Zu Beginn steht ein Naive Bayes Klassifikator, ein Repräsentant für ein probabilistisches Modell. Danach wird eine Support Vektor Machine betrachtet, wobei die Klassen linear oder nicht-linear im Raum separiert werden und somit ein Klassifikationsmodell erstellt wird. Auch vertreten ist eine Methode, die den Ansatz der k-nächsten Nachbarn betrachtet. Und zum Abschluss folgt ein Random Forest Modell, welches auf Entscheidungsbäumen beruht.

2.3.1 Naive Bayes

Das erste Modell zur Klassifikation der Buchklappentexte ist ein Naive Bayes Klassifikator (NB), ein einfaches überwachtes Lernverfahren. Oftmals wird dieses Modell als Baseline-Performance in der Textklassifikation benutzt [6]. Grundlage stellt das Bayes-Theorem in Gleichung (3) dar [7].

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)} \tag{3}$$

Dabei stellt $P(H)$ die a-priori-Annahme und $P(H|D)$ die a-posteriori-Wahrscheinlichkeit dar. Also ist $P(H)$ die Annahme über den Wahrheitsgehalt der Hypothese H und $P(D|H)$ die Wahrscheinlichkeit der Beobachtung nach dem Eintreten der Hypothese H.

Der Naive Bayes Algorithmus ist ein einfaches statistisches Modell, dass neben dem Bayes-Theorem eine zweite grundlegende Anforderung stellt: Die Unabhängigkeit der Merkmale wird vorausgesetzt. Daraus resultiert der Nachteil, dass mögliche Abhängigkeiten der Merkmale kategorisch ignoriert werden und somit die theoretisch mögliche Genauigkeit eingeschränkt wird.

Allgemein ist der Klassifikator eine Funktion f, die einen n-dimensionalen Vektor auf die Menge aller Klassen C abbildet. Je nach dem, ob ein binäres oder ein Multi-Klassen-Problem betrachtet wird, wird die Menge aller Klassen als $C = \{0, 1\}$ oder als $C = \{0, \ldots, c\}$ dargestellt. Die Daten werden derjenigen Klasse mit der höchsten Wahrscheinlichkeit zugeordnet.

Obwohl in den meisten Anwendungsfällen die Voraussetzung der Unabhängigkeit der Attribute verletzt wird, erzielt der Algorithmus gute und stabile Ergebnisse, seine Implementierung ist in der Regel leicht. Er prozessiert kategorische Daten sehr gut, für numerische Werte wird eine Verteilung vorausgesetzt (z.B. die Gaußsche Normalverteilung). Für die Klassifikation von Texten eignet sich der Naive Bayes Klassifikator ebenfalls gut, weil er zusätzlich die Multi-Class-Prediction erlaubt. [8]

Bei der Zuordnung eines Dokuments $\overrightarrow{d} = (w_1, ..., w_n)$ zu einer bestimmten Klasse $c \in \mathbb{N}_C = \{1, ..., C\}$ hat jedes Feature w_n einen Einfluss. Werden Terme als Feature gewählt, soll jeder Term zur Klassifikation beitragen. Um ein Dokument d_i einer Klasse zuzuordnen, beginnt der Algorithmus bei der Bestimmung der a-priori-Wahrscheinlichkeit $P(c)$ für jede Klasse.

Die a-priori-Wahrscheinlichkeiten sind laut Annahme gleichwahrscheinlich oder abhängig von der Häufigkeit der Klassen im Trainingscorpus. Danach wird der Einfluss jedes einzelnen Features mit hinzugezogen und es entsteht eine Likelihood-Wahrscheinlichkeit für jede Kategorie. Als letzter Schritt folgt eine Arg-Max-Funktion, die sogenannte Maximum-a-posteriori-

Schätzung wie in Gleichung (4). [6]

$$\arg\max_{c\in\mathbb{N}_\mathbb{C}} p(c|\overrightarrow{w}) = \arg\max_{c\in\mathbb{N}_\mathbb{C}} P(c)\, p(\overrightarrow{w}|c) \tag{4}$$

2.3.2 Support Vector Machine

Die Support Vector Machines (SVM's) sind eine weitere Art von überwachten Lernverfahren, die sich zur Textklassifikation und Regression eignen. Dabei werden alle Attribute der Trainingsmenge in einen Vektorraum überführt. Vereinfacht wird an dieser Stelle ein 2D-Problem angenommen. In Abbildung 2 ist dieser Sachverhalt schematisch dargestellt.

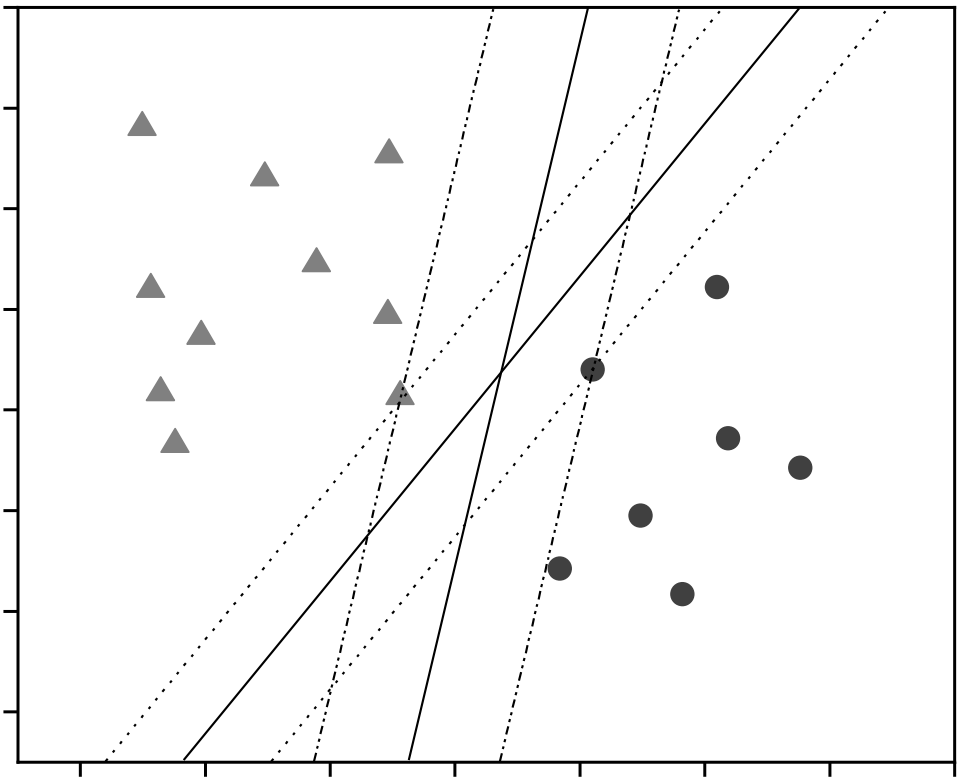

Abbildung 2: Schematische Darstellung der linearen Trennung zweier Klassen

Im einfachsten Fall ist eine lineare Funktion gesucht, die zwei Klassen der Attribute optimal trennt. In der Abbildung 2 sind zwei Trennebenen eingezeichnet, die dem Anspruch genügen, die Klassen zu separieren. Die Aufgabe einer SVM ist das Finden der optimalen Funktion. Dabei muss der Abstand zwischen Trennebene und den nächsten Vektoren, den sogenannten Supportvektoren (Attribute) maximiert werden. Dadurch entsteht eine möglichst breite Bande, durch die die Attribute am besten getrennt werden und die Klassifikation eindeutiger wird. Testdaten werden nun in den Raum eingefügt und anhand der Trennebene zu der jeweiligen Klasse zugeordnet.

Existiert keine Möglichkeit die Daten linear zu trennen, wird eine Datentransformation angewandt. Dabei werden die Daten in einen hochdimensionalen Raum überführt. Der Input-

Space X ist der Ausgangsraum, in ihm liegen die Rohdaten. Mit Hilfe einer nichtlinearen Abbildungsfunktion $\phi : X \rightarrow H$ werden die Datenpunkte in den Feature-Space H transformiert [9].

Abhängig vom Umfang der Ausgangsdaten, kann der Input-Space sehr hochdimensional sein. Durch die Transformation werden nun noch deutlich mehr Dimensionen hinzugefügt und die Rechenanforderungen steigen deutlich. Daraus resultiert ein Optimierungsproblem, in dem die Datenvektoren als Skalarprodukt $x_i \cdot x_j$ auftreten. Dieses muss im Feature-Space nicht explizit berechnet werden, sondern kann mit Hilfe einer Kernel-Funktion zu $k(x_i, x_j)$ substituiert werden [10].

Mögliche Kernel-Funktionen sind polynomiale, RBF (Radial Basis Function) oder sigmoide Funktionen. Die Kernel-Funktion stellt letztlich nur eine Art Ähnlichkeitsfunktion dar, mit der die Datenpunkte verglichen werden können.

Auch kann eine optimale Trennung erfolgen, obwohl nicht alle Trainingsdaten vollständig voneinander isoliert sind. Werden Fehler zugelassen, sogenannte Schlupfvariablen, kann das die Breite der Bande erhöhen. In diesem Fall werden nicht die nächsten Vektoren zur Trennebene als Support Vektoren verwendet, sondern die Vektoren, die die Breite maximieren.

Bei ausreichend gut vorverarbeiteten Daten liefert eine SVM sehr gute Klassifikationsergebnisse, jedoch muss eine gute Kernel-Funktion gewählt werden, damit mit wachsendem Umfang der Trainingsdaten auch der Rechenaufwand umsetzbar ist. Außerdem ist eine Überanpassung des Modells an das Trainingsset zu beachten. Sind beispielsweise die Trainingsdaten von geringem Umgang, werden die Trennebenen nur durch die wenigen Daten berechnet. Dadurch kann der Algorithmus bei den wenigen Trainingsdaten sehr gute Ergebnisse liefern, jedoch verschlechtert sich das Ergebnis schnell bei neuen Daten. Dieses Overfitting ist ein deutlicher Nachteil einer SVM.

2.3.3 k Nearest Neighbor

Der k Nearest Neighbor (kNN) Algorithmus ist ein weiterer populärer Klassifikationsalgorithmus. Der Unterschied zu den bisher genannten Modellen ist, dass der Algorithmus während der Modellerstellung nicht rechenaufwändig ist. Erst nachdem das Modell erstellt wurde und neue Daten gelabelt werden müssen, beginnt der Rechenaufwand.

Der Grundgedanke ist die Klassifikation eines Attributs anhand seiner k-Nächsten Nachbarn im Attributraum. Zu Beginn werden alle Attribute der Trainingsdaten im Attributraum anhand ihrer Werte pro Dimensionen platziert. Damit ist die Modellerstellung bereits abgeschlossen. Mit dem Klassifizieren neuer Daten beginnt die Berechnung, denn mit dem platzieren des neuen Datenpunktes im Raum müssen seine k-nächsten Nachbarn ermittelt werden. In Abbildung 3 ist dieser Prozess grafisch dargestellt.

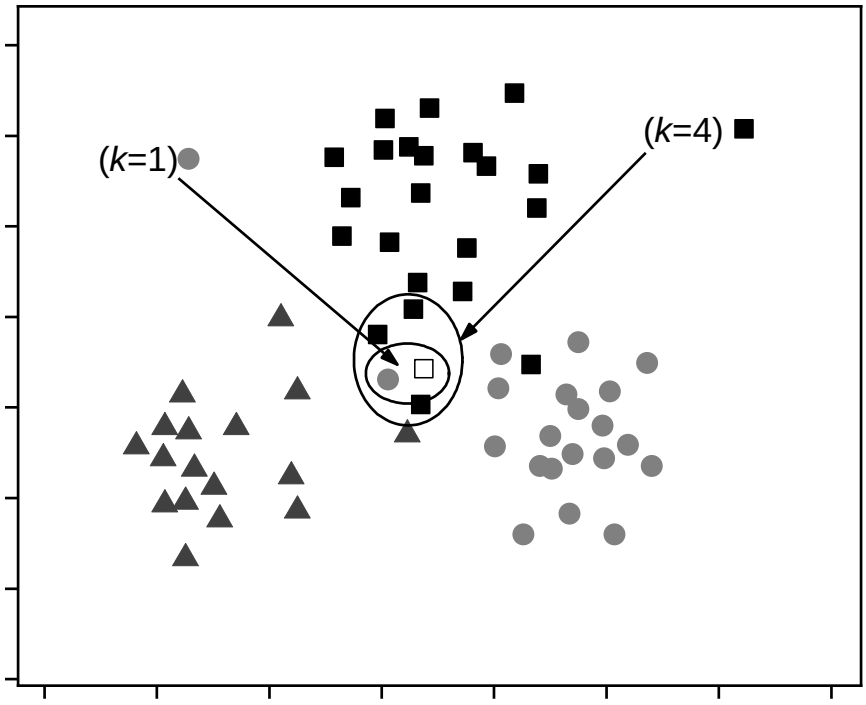

Abbildung 3: Schematische Darstellung einer Klassifikation anhand k-nächster Nachbarn

Im einfachsten Fall wird die Klassenzugehörigkeit anhand der dominierenden Kategorie der k Nachbarn ermittelt. Wird wie in Abbildung 3 $k = 1$ gewählt, spielt nur der nächste Nachbar eine Rolle für die Klassifikation. Bei $k = 4$ wird das zu klassifizierende Attribut zu der dominanten Klasse zugeordnet. [1]

An dieser Stelle muss darauf geachtet werden, dass k kein Vielfaches der Anzahl der Klassen darstellt, da sonst ein Gleichstand der Klassen erreicht werden kann. Weiterhin muss eine Regel gewählt werden, um die nächsten Vektoren im Raum zu finden. Das wird über Distanzmaße realisiert, wobei eine allgemeine Grundlage für verschiedene Maße die Minkowski-Distanz in Gleichung 5 ist. [11]

$$d(i,j) = \sqrt[n]{(|x_{i_1} - x_{j_1}|^n + |x_{i_2} - x_{j_2}|^n + ... + |x_{i_r} - x_{j_r}|^n)} \tag{5}$$

Im Fall $n = 1$ resultiert aus Gleichung 5 die Manhattan-Distanz in Gleichung 6, auch L1-Norm genannt.

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + ... + |x_{i_r} - x_{j_r}| \tag{6}$$

Eine weitere, verbreitete Version aus Gleichung 5 ist die euklidische Distanz in Gleichung 7. Dabei ist $n = 2$.

$$d(i,j) = \sqrt{|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + ... + |x_{i_r} - x_{j_r}|^2} \tag{7}$$

Weitere Maße, unabhängig von der Minkowski-Distanz, sind zum Beispiel die Cosinus-Ähnlichkeit in Gleichung 8 oder der Tanimoto-Koeffizient in Gleichung 9.

$$sim(x, y) = cos(\alpha) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{8}$$

$$sim(x, y) = cos(\alpha) = \frac{x \cdot y}{||x||^2 + ||y||^2 - x \cdot y} \tag{9}$$

Die Maße ermöglichen das Ermitteln der Distanz zweier Objekte im Raum. Natürlich müssen nicht nur numerische Werte verglichen werden. Entweder werden Strings in numerische Daten überführt, oder es wird beispielsweise der Hamming-Abstand oder die Levenshtein-Distanz gewählt. Dadurch kann der Abstand von Zeichenketten bestimmt werden.

Weiterhin kann der Algorithmus verbessert werden, indem nicht nur die dominierende Klasse der k-Nächsten Nachbarn das Klassifikationsergebnis bestimmt, sondern auch der Abstand mit einbezogen wird. So kann die reziproke Entfernung als Gewichtsfaktor das Ergebnis regulieren.

Der kNN-Algorithmus ist für Mehrklassen-Probleme geeignet und liefert gute Ergebnisse. Jedoch ist vor allem bei vielen Dimensionen ein großer Speicherbedarf vorhanden und die Rechnungen werden komplex.

2.3.4 Random Forest

Die Grundlage von Random Forests sind einfache Entscheidungsbäume, die zusammen den Algorithmus aufbauen. Die Entscheidungsbäume sind unkorreliert und entstehen während der Berechnung randomisiert. [12]

Die Klassifikation mit einem Entscheidungsbaum beginnt im Wurzelknoten, indem eine erste Entscheidung über ein Attribut getroffen wird. Danach folgen weitere Knoten unterhalb des Wurzelknotens [13]. Diese Entscheidungen innerhalb der Knoten werden solange getroffen, bis ein Blatt erreicht und somit eine Klasse zugeordnet wird. Wesentlichen Einfluss hat dabei die Art und Weise, wie die Knoten angeordnet werden. Dabei soll im einfachen Entscheidungsbaum ein effizientes Attribut gewählt werden, beispielsweise mit dem ID3-Algorithmus [14]. Mithilfe einer Bewertungsfunktion wird für jedes Attribut der Informationsgehalt bestimmt und dann absteigend des Informationsgehaltes der Baum vom Wurzelknoten aus erstellt. Dadurch wächst ein Baum in die Breite und nicht in die Tiefe. Somit haben Knoten weiter oben im Baum größeren Einfluss auf die Klassifikation.

Abhängig von der Anzahl und Beschaffenheit der Trainingsdaten ist ein Overfitting möglich. Deswegen sollten Methoden wie pre-/ oder postpruning angewendet werden. Im prepruning wird die Baumentwicklung gestoppt, bevor jedes Element perfekt klassifiziert ist. Dagegen wird beim postpruning der Baum komplett entwickelt und im Nachhinein werden Äste abgeschnitten.

Die Knoten, mit denen für jedes Element die Entscheidungen getroffen werden, legen beim univarianten Entscheidungsbaum eine achsenparallele Hyperebene in den Attributraum. Somit wird der Attributraum in die jeweiligen Klassen geteilt. Außerdem existieren multivariante Entscheidungsbäume, deren Trennebenen nicht achsenparallel sind. Eine letzte Klasse sind die nichtlinearen multivarianten Entscheidungsbäume.

Ein Random Forest Algorithmus kombiniert nun eine Vielzahl von Entscheidungsbäumen miteinander um die Resultate zu verbessern. Das überwachte Lernverfahren ist neben der Klassifikation zur Regression geeignet. Während bei einfachen Entscheidungsbäumen absteigend nach effizienten Kriterien gesucht wird, passiert dieser Prozess bei Random Forest Algorithmen zufällig. Beispielsweise wird ein zufälliges Subset von Kategorien genommen und daraus das Effizienteste gewählt. Auch die jeweilige Entscheidung anhand von Schwellwerten im Knoten kann randomisiert werden, indem unterschiedliche Tresholds getestet werden. Somit entsteht eine breite Möglichkeit an verschiedenen Entscheidungsbäumen, deren jeweiliges Resultat in die finale Klassifizierung eingeht.

2.4 Evaluation

Es existieren verschiedene Kennzahlen und Methoden für die Evaluation von Textklassifikationen, die an dieser Stelle vorgestellt werden. Zu Beginn stellt sich jedoch die Frage, anhand welcher Daten die Güte eines Klassifikators ermittelt werden soll. Grundlage ist stets ein ausreichend großer und gelabelter Datensatz, der problemlos in Trainings- und Testdaten geteilt werden kann.

Eine mögliche Aufteilung sind 80% Trainingsdaten und 20% Testdaten. Andere Quellen teilen den Datencorpus in 50% Trainingsdaten, 25% Validationsdaten und 25% Testdaten [15]. Anhand der Trainingsdaten wird ein Klassifikationsmodell erstellt und angelernt. Die Validationsdaten sind nützlich, um verschiedene Modelle und deren Parameter zu justieren. Die Testdaten werden abschließend mit dem Modell gelabelt und die resultierenden Label werden mit den Ausgangsdaten verglichen.

Das starre und einmalige Teilen des Corpus in beispielsweise 80/20 oder 50/25/25 birgt große Fehlerquellen. Da das Modell stets von den Trainingsdaten abhängt, hat die Auswahl der Trainingsdaten einen Einfluss auf die Performanz des Modells. Um ein zufällig gutes oder schlechtes Ergebnis zu vermeiden, sollte diese Auswahl mehrfach getroffen werden. Möglich sind einfache oder multiple Split-Tests, die trotzdem noch eine sehr große Modellvarianz bedeuten. Dabei sind einige Datenpunkte oft im Training vorhanden, andere jedoch nie.

Eine deutlich bessere Lösung ist eine sogenannte k-fold Cross-Validation. Dabei wird der gesamte Datensatz k-mal in Test- und Trainingsdaten geteilt (prozentuale Aufteilung trotzdem vorgeschrieben). Dabei wird jeder Datenpunkt einmal zum Testen verwendet und $k-1$-Mal zum Training. Somit ist kein Datenpunkt ungesehen und die Zufälligkeit des Modells geht verloren. Mögliche Werte für k sind üblicherweise 3, 5 oder 10. Auch eine Repeated-k-fold Cross-Validation ist möglich. Dabei wird das beschriebene Verfahren n-mal wiederholt.

Nachdem das Modell eine Vorhersage der Label geliefert hat, kann diese Vorhersage mit dem realen Label verglichen werden. Dafür existiert eine Vielzahl von Maßen, von denen eine Auswahl vorgestellt wird.

Eine sehr verbreitete Visualisierung von Klassifikationsergebnissen ist die Konfusionsmatrix wie in Tabelle 2 dargestellt.

Tabelle 2: Beispiel einer Konfusionsmatrix

	positiv (P)	negativ (N)
positiv gelabelt	richtig positiv r_p	falsch positiv f_p
negativ gelabelt	falsch negativ f_n	richtig negativ r_n

Die Hauptdiagonale beschreibt die richtig klassifizierten Datenpunkte. Die Felder oben rechts und unten links sind die falsch Klassifizierten. Anhand dieser Aufteilung der Daten kann die Güte eines Modells beschrieben werden.

Die Accuracy stellt ein erstes und einfaches Maß für die Evaluation dar. Sie beantwortet die Frage, wie viele Objekte richtig erkannt wurden. In Gleichung 10 ist dieses Maß formal beschrieben.

$$accuracy = \frac{r_p + r_n}{r_p + f_p + r_n + f_n} \tag{10}$$

Die Accuracy als alleiniges Maß kann den Klassifikator fälschlicherweise als gut beschreiben. Eine wichtige Voraussetzung zur Anwendung ist ein balancierter Datensatz, was bedeutet, dass die selbe Anzahl von positiven und negativen Dokumenten vorhanden ist. Weiterhin müssen die Kosten für eine Fehlklassifikation in jedem Fall gleich hoch sein.

Das zweite Maß ist die Präzision (Precision, P), das sinngemäß angibt, wie viele der positiv gelabelten Objekte tatsächlich positiv sind. Gleichung 11 beschreibt die Berechnung der Precision.

$$precision = \frac{r_p}{r_p + f_p} \tag{11}$$

Ergänzend zur Präzision existiert die Sensitivität (Recall, R) wie in Gleichung 12, die angibt, wie viele der positiven Dokumente als positiv zurückgegeben wurden.

$$recall = \frac{r_p}{r_p + f_n} \tag{12}$$

Ein idealer Klassifikator erreicht in beiden Maßen 100%, er klassifiziert alle Dokumente richtig. Mit realen Textdaten sind solche Maße nicht zu erreichen und das Ziel ist, die Maximierung beider Werte. Um einen Klassifikator anhand dieser Werte einzuschätzen, muss eine Kombination dieser erfolgen. Mithilfe des gewichteten harmonischen Mittels von Precision und Recall, auch F-Maß genannt, kann ein Modell ausreichend genau bewertet werden. In Gleichung 13 ist die allgemeine Formel dargestellt.

$$F_\alpha = (1 + \alpha^2)\frac{precision \cdot recall}{\alpha^2 \cdot precision + recall} \tag{13}$$

Für den Fall $\alpha = 1$ sind Precision und Recall gleich gewichtet, was den häufigsten Anwendungsfall darstellt. Je nach dem, worauf der Fokus gelegt wird, kann das Gewicht zu einer der beiden Größen gelenkt werden ($\alpha > 1$: höhere Gewichtung von Recall; $\alpha < 1$ höhere Gewichtung für Precision).

Eine andere Art der Visualisierung ist eine ROC-Curve (Reciever Operating Characteristics). Dabei geht es um die visuelle Beurteilung eines Klassifikators, was auch einen ersten Vergleich mehrerer Ergebnisse erleichtert. In Abbildung 4 ist eine ROC-Curve dargestellt,

dabei werden die true positive rate (tp rate) aus Gleichung 14 und die false positive rate (fp rate) aus Gleichung 15 gegenübergestellt.

$$tp\ \ rate = \frac{r_p}{P} \tag{14}$$

$$fp\ \ rate = \frac{f_p}{N} \tag{15}$$

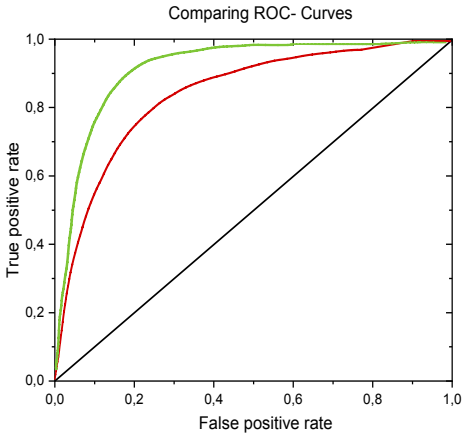

Abbildung 4: Beispiel ROC-Curve, schwarz: schlechte Klassifikation, rot: mäßige Klassifikation, grün: gute Klassifikation

Das Ziel ist den Kurvenverlauf so anzupassen, dass die Kurve möglichst stark ansteigt und die $fp\ \ rate$ erst spät zunimmt. Als numerischer Wert ist die Fläche unter der Kurve zu nehmen, auch AUC - Area Under Curve genannt. Im Idealfall ist diese Fläche gleich eins.

3 Datensatz

In diesem Kapitel werden die zur Verfügung stehenden Daten vorgestellt und grundlegende statistische Auswertungen am Trainings- und Evaluationsdatensatz vorgenommen. Dabei werden die wichtigsten Eigenschaften erklärt und auf Probleme des Datensatzes hingewiesen.

3.1 Datenbeschreibung

Die zur Verfügung stehenden Daten liegen im XML-Format vor. Jedes Buch stellt einen eigenen Knoten mit den dazugehörigen Eigenschaften dar. Dazu zählen Titel, Text, Copyright, Kategorie, Autor, Veröffentlichungsdatum, ISBN und die URL zum Buch.

Der nächste Absatz beschreibt exemplarisch ein Buch, dessen Informationen und Struktur in der XML-Datei.

< bookdate = "2019 − 01 − 04" xml : lang = "de" >
<title> Blueberry Summer</title>
<body>Ein neuer Sommer beginnt für Rory und Isabel - mit einer kleinen Neuerung: Rory ist fest mit Connor Rule zusammen und deshalb als Hausgast in den Hamptons. Und genau das bringt Komplikationen mit sich, denn irgendwie scheint Connor ein Problem damit zu haben, dass Rory nicht mehr für seine Familie arbeitet. Isabel dagegen arbeitet zur Überraschung aller als Kellnerin, um einen süßen Typen zu beeindrucken - irgendwie muss sie ja über ihre Affäre mit Mike hinwegkommen. Das klappt ganz gut, bis Rory auf Isabels Neuen trifft ... Und Isabel wieder auf Mike.</body>
<copyright>(c) Verlagsgruppe Random House GmbH</copyright>
<categories>
<category>
<topic d=„0“>Kinderbuch & Jugendbuch</topic>
<topic d=„1“ label=„True“>Liebe, Beziehung und Freundschaft</topic>
</category>
<category>
<topic d=„0“>Kinderbuch & Jugendbuch</topic>
<topic d=„1“ label=„True“>Echtes Leben, Realistischer Roman</topic>
</category>
</categories>
<authors>Joanna Philbin</authors>
<published>2015-02-09</published>
<isbn>9780451457998<isbn>
<url>https://www.randomhouse.de/Taschenbuch/Blueberry-Summer/
Joanna-Philbin/cbj-Jugendbuecher/e455949.rhd</url>
</book>

Da der zweite Teil der offiziellen Aufgabenstellung die Einordnung in eine Hierarchie darstellt, sind pro Buch die Oberkategorie und die dazugehörigen Unterkategorien beschrieben. Die acht Oberkategorien lauten:

1. Architektur & Garten
2. Ganzheitliches Bewusstsein
3. Glaube & Ethik
4. Kinderbuch & Jugendbuch
5. Künste
6. Literatur & Unterhaltung
7. Ratgeber
8. Sachbuch

Die jeweilige Zuordnung ist im Tag $d = 0$ zu finden, die tiefere Hierarchie in Tag $d = 1$ und $d = 2$.

3.2 Probleme des Datensatzes

Die zur Verfügung gestellten Daten weisen mehrere Fehler auf, die eine einfache Verarbeitung erschweren und zu Beginn beseitigt werden müssen. So ist das Format, in dem die Blurbs vorliegen, ein reines .txt-Format. Die XML-Struktur der Daten ist nicht vollständig vorhanden. Dem Datensatz fehlt der Root-Node (z.B. $< books >< /books >$), ein öffnender und schließender Knoten, der alle vorhandenen Informationen umschließt. Der Knoten wurde händisch eingefügt.

Außerdem wurden in der Beschreibung der Kategorien Zeichen verwendet, die während dem Einlesen der Datei Probleme verursachen. Das „&“-Symbol in den Kategorien „Architektur & Garten“, „Glaube & Ethik“, „Kinderbuch & Jugendbuch“ und „Literatur & Unterhaltung“ kann beim Einlesen der Datei nicht verarbeitet werden. Deshalb wurde das Symbol für jeden Eintrag in ein „+“-Symbol überführt.

Ein weiteres Problem in der Formatierung der Rohdaten, in Kombination mit der Verarbeitung mit R, liegt in der Beschreibung der Kategorie. Der Knoten für die Oberkategorie lautet „<topic d=„0“>. . . </topic>“, wobei öffnender und schließender Tag nicht übereinstimmen. Das gleiche Problem tritt auch in den tiefer liegenden Schichten der Hierarchie auf. Das Problem wurde programmatisch gelöst: im ersten Schritt wurde jede Zeile gelöscht, die mit „<topic d=„1“ “ und „ <topic d=„2“ “ beginnen. Somit bleibt nur noch Level d=0 vorhanden, was die weitere Verarbeitung vereinfacht. Dieses Vorgehen ist nur anwendbar, weil die Klassifikation der tieferen Ebenen ignoriert wird.

Im zweiten Schritt wurde jedes Auftreten von „<topic d=„0“>“ in „<topicd=0>“ geändert (entsprechender schließender Knoten äquivalent). Dadurch sind nur noch die Informationen der Oberkategorie vorhanden, aber die Datei als XML-Format einlesbar.

Theoretisch sollte jedes Buch genau einer Oberkategorie und den treffenden Unterkategorien zugeordnet sein, und somit jeder Text exakt in einen Ast der Taxonomie passen. Doch gibt es mehrere Fälle, in denen dieses Kriterium nicht erfüllt wurde. Zum einen gibt es Blurbs, die mehreren verschiedenen Oberkategorien zugeordnet wurden und zum anderen existieren

Einträge, die zwar die gleiche Oberkategorie besitzen, aber verschiedene hierarchische Abstufungen. In dem einleitenden Beispiel der XML-Datei ist das Problem zu erkennen. Die verschiedenen Unterkategorien spielen in der folgenden Betrachtung keine Rolle, da nur das Level d = 0 betrachtet wurde.

Der erste Fehler, dass verschiedene d = 0 vorhanden sind, wurde korrigiert, in dem jeweils nur das erste Auftreten des Levels pro Text verwendet wurde. Die Auswertung der Daten ergab, dass von allen 14.548 Trainings-Texten 4.510 mehrfach zugeordnet wurden. Davon sind ca. 1.000 Texte zu verschiedenen Oberkategorien zugeordnet.

Ein letzter Fehler, der die inhaltlichen Analysen verfälschen kann, ist das Vorhandensein leerer Knoten. In 188 Fällen der Trainingsdaten war der Tag der Autoren leer, sodass diese Information bei den Büchern komplett fehlt und in 11 Fällen fehlte der Klappentext. Diese 199 Bücher wurden vom Datensatz entfernt und nur mit vollständigen Daten gearbeitet. Die Evaluationsdaten enthalten lediglich 28 Bücher, bei denen die Autoren fehlen.

3.3 Statistik der Trainingsdaten

Der Trainingsdatensatz umfasst 14.548 Einträge, jeweils mit den oben genannten Informationen. In Abbildung 5 und Tabelle 3, Spalte 2 und 3 sind die Verteilungen auf die Oberkategorien dargestellt.

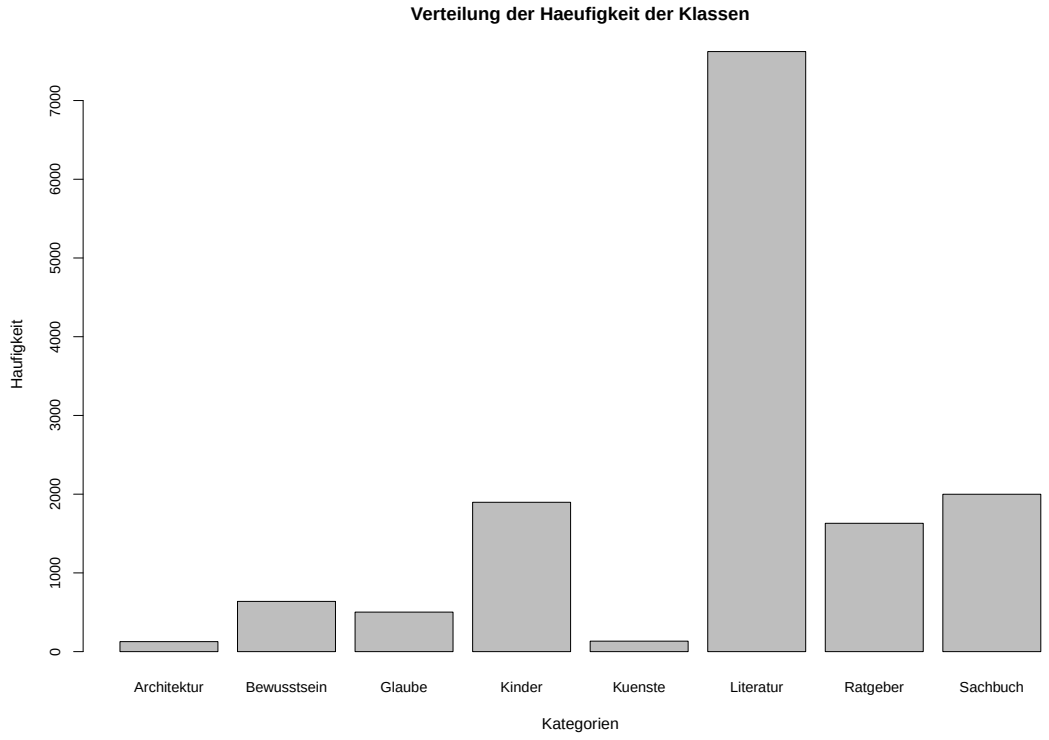

Abbildung 5: Verteilung der Bücher auf die acht Klassen der Trainingsdaten

Tabelle 3: Verteilung der Klassen und Autoren im Trainingsdatensatz

Kategorie	abs. Anzahl Datensätze	rel. Anzahl Datensätze	abs. Anzahl Autoren	rel. Anzahl Autoren
Architektur & Garten	127	0.0087	123	0.0146
Ganzheitliches Bewusstsein	638	0.0439	422	0.0501
Glaube & Ethik	502	0.0345	476	0.0565
Kinderbuch & Jugendbuch	1.897	0.1304	787	0.0934
Künste	133	0.0091	164	0.0195
Literatur & Unterhaltung	7.622	0.5239	3.107	0.3687
Ratgeber	1.630	0.1120	1.440	0.1709
Sachbuch	1.999	0.1374	1.909	0.2265
Gesamt	14.548	1.0000	8.428	1.0000

Der Trainingsdatensatz besitzt mit ca. 15.000 Texten einen ausreichend großen Umfang, um die Modelle zu trainieren. Wie weiter oben schon angesprochen, ist die Verteilung sehr ungleich. Die umfangreichste Klasse Literatur & Unterhaltung besitzt ca. 60 mal mehr Bücher als die kleinste Klasse Architektur & Garten.

Im Durchschnitt besitzen die Klappentexte ca. 94 Wörter pro Buch. Die 10 häufigsten Terme sind in Tabelle 4 zusammengefasst. Dieser Fakt wird auch in Kapitel 4.2 relevant, wenn der Nutzen eines Stoppwortfilters erklärt wird und sich die Wortverteilungen ändern (Vgl. mit Tabelle 9). Eine weitere Eigenschaft, die sich von den Termfrequenzen ableiten lässt, ist die Zipf-Verteilung. Kurz zusammengefasst besagt dieses Gesetz, dass einige wenige Elemente (in diesem Fall Terme) sehr häufig vorkommen. Der Großteil der Elemente hingegen tritt sehr selten auf. In Abbildung 6 ist die Zipf-Verteilung aller Terme des Trainingsdatensatzes dargestellt. Eine andere Verteilung, die diesem Kriterium entspricht, ist beispielsweise die Verteilung der acht Klassen. Literatur tritt im Vergleich zu den anderen Klassen als einzige sehr häufig auf, die anderen jeweils seltener.

Tabelle 4: Die häufigsten Terme des Trainingsdatensatzes

Term	Anzahl im Trainingscorpus
und	49.949
die	42.813
der	36.696
sie	17.026
ist	14.581
den	14.472
das	14.129
mit	13.347
ein	13.293
sich	13.085

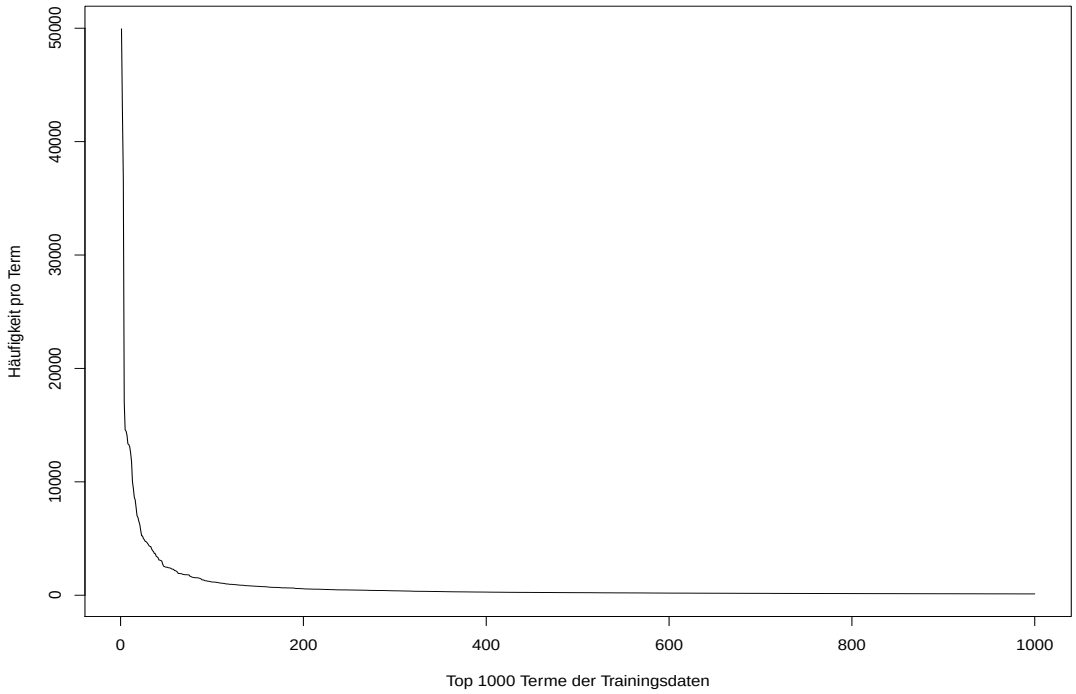

Abbildung 6: Zipfverteilung der Terme der Trainingsdaten

Ein weiterer wichtiger Aspekt sind die Autoren, da sie als mögliches Feature in Frage kommen. Insgesamt gibt es 7.867 verschiedene Autoren im Trainingsdatensatz. In Tabelle 3, Spalte 4 und 5 ist die Verteilung dargestellt, wie viele Autoren pro Klasse bekannt sind. In Tabelle 5 sind die zehn Autoren aufgelistet, die die meisten Bücher verfasst haben.

Am Beispiel der Kategorie Künste ist zu erkennen, dass mehr Autoren als Bücher existieren, was daran liegt, dass ein Buch von mehreren Autoren geschrieben wurde. Ein Gegenteil dazu stellt Literatur dar, wo es deutlich mehr Bücher als Autoren gibt. Die Summe über die Autoren aus Tabelle 3 ergibt 8.428, was ebenso über der Anzahl von 7.867 verschiedenen Autoren liegt. Der Grund dafür liegt in der Möglichkeit, dass die Autoren in verschiedenen Klassen/Genres parallel publizieren.

Tabelle 5: Hauptautoren des Trainingsdatensatzes

Autor	Anzahl Bücher
Nora Roberts	102
Ingo Siegner	75
Enid Blyton	63
Clive Cussler	56
Terry Pratchett	55
Ruediger Dahlke	54
Stephen King	50
Robert Ludlum	42
Anne Perry	39
James Patterson	38

3.4 Statistik der Evaluationsdaten

Der Evaluationsdatensatz enthält 2.079 Bücher (mit Abzug der Bücher mit leeren Feldern 2.051). In Abbildung 7 und Tabelle 6, Spalte 2 ist die Verteilung dargestellt.

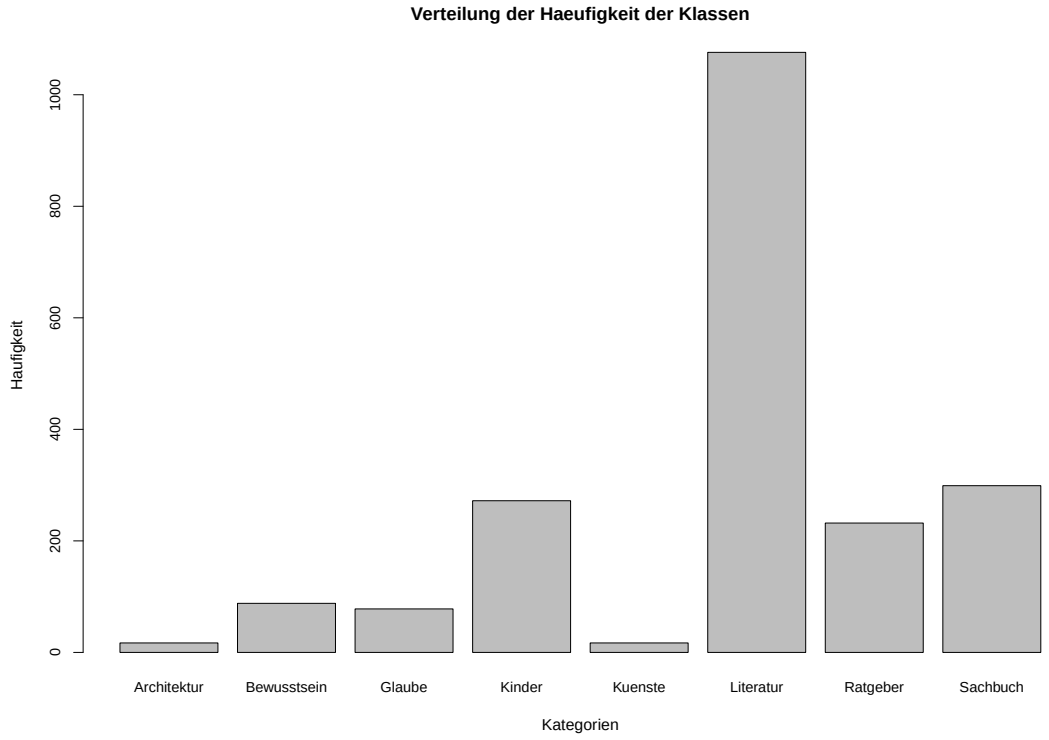

Abbildung 7: Verteilung der Bücher auf die acht Klassen der Evaluationsdaten

Tabelle 6: Verteilung der Klassen und Autoren im Evaluationsdatensatz

Kategorie	abs. Anzahl Datensätze	rel. Anzahl Datensätze	abs. Anzahl Autoren	rel. Anzahl Autoren
Architektur & Garten	17	0.0082	21	0.0112
Ganzheitliches Bewusstsein	88	0.0423	95	0.0505
Glaube & Ethik	78	0.0375	116	0.0617
Kinderbuch & Jugendbuch	272	0.1308	209	0.1112
Künste	17	0.0082	24	0.0128
Literatur & Unterhaltung	1.076	0.5176	816	0.4340
Ratgeber	232	0.1116	260	0.1383
Sachbuch	299	0.1438	339	0.1803
Gesamt	2.079	1.0000	1.880	1.0000

Die Verteilung der Klassen im Evaluationsdatensatz ist der Verteilung der Trainingsdaten sehr ähnlich. Die größte Klasse ist auch hier Literatur & Unterhaltung und die kleinste ist Architektur & Garten (neben Künste). Die durchschnittliche Wortanzahl beträgt 96 Wörter pro Klappentext. Die zehn häufigsten Terme der Evaluationsdaten sind in Tabelle 7 dargestellt.

Tabelle 7: Die häufigsten Terme des Evaluationsdatensatzes

Term	Anzahl im Evaluationscorpus
und	7208
die	6195
der	5247
sie	2461
ist	2068
den	2049
das	2047
sich	1954
ein	1950
von	1896

Auf eine graphische Darstellung der Termverteilung (Zipf-Verteilung) der Evaluationsdaten wird verzichtet, da diese exakt denselben Kurvenverlauf wie bei den Trainingsdaten aufweist.

Der große Vorteil liegt darin, dass die relativen Zahlen der Klassen in beiden Datensätzen identisch sind. Die Modelle zur Klassifikation beachten bei der Modellerstellung genau dieses Verhältnis. Somit kann mit den Trainingsdaten trainiert werden und das Modell ohne eine Anpassung, beispielsweise durch eine Gewichtung, auf den zweiten Datensatz angewandt werden.

Für den zweiten Datensatz existieren 1.835 verschiedene Autoren, welche in Tabelle 6, Spalte 4 auf die Klassen verteilt dargestellt sind. In Tabelle 8 sind die zehn häufigsten Autoren dargestellt.

Tabelle 8: Hauptautoren des Evaluationsdatensatzes

Autor	Anzahl Bücher
Nora Roberts	19
Ingo Siegner	18
Enid Blyton	12
Agatha Christie	7
Hanns-Josef Ortheil	7
Jeffery Deaver	7
Mary Higgins Clark	7
Stephen King	7
Wolfgang Jeschke	7

4 Datenvorverarbeitung

In diesem Kapitel wird die Vorverarbeitung der Daten erläutert. Es wird der gesamte Weg von den Rohdaten, bis zum finalen Format für die Modellierung dargelegt. Darüber hinaus wird die Methodik für das Balancieren der Datensätze vorgestellt.

4.1 Standardschritte der Vorverarbeitung

Der Großteil der Vorverarbeitung wurde mit der Scriptsprache R umgesetzt, zum Teil kam Java ergänzend zum Einsatz. Die angewandten Methoden und Schritte werden nun genauer beschrieben und mit Hilfe von Flowcharts visualisiert.

Die ersten Schritte widmeten sich der Lesbarkeit der XML-Dateien. Wie In Kapitel 3.2 beschrieben, sind strukturelle Fehler in den Rohdaten vorhanden. Sowohl die Trainingsdaten, als auch die Evaluationsdaten mussten bearbeitet werden. In Abbildung 8 sind noch einmal alle nötigen Schritte zusammengefasst.

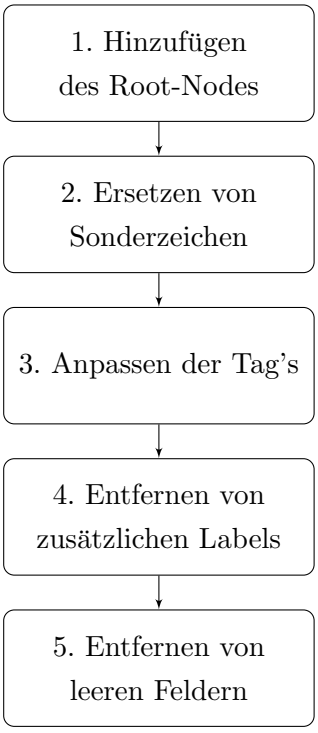

Abbildung 8: Schritte zur Beseitigung der Probleme

Da die Fehler in Kapitel 3.2 ausführlicher beschrieben wurden, werden nun nur kurz die Schritte benannt. Die Root-Nodes wurden händisch am Anfang und Ende der Datei eingefügt. Die Sonderzeichen wurden per einfacher Suchen & Ersetzen-Methodik entfernt. Die Datei mit den Fehlerhaften und nicht benötigten Tag's (Level > d=0) wurde zeilenweise eingelesen und jede Zeile, die einen nicht benötigten Tag enthält, wurde entfernt. Somit resultierte eine Datei, die nur Level d=0 enthält. Die Anpassung der Darstellung dieses Tags wurde wieder mit Suchen & Erstzen erreicht. Der letzte Schritt stellte das Entfernen von leeren Feldern dar. Zu diesem Zeitpunkt konnte die Datei wie erforderlich in R als XML-Datei geparst werden.

Nachdem die XML-Datei in ein Dataframe überführt wurde, konnten die Zeilen ermittelt werden, die leere Felder beinhalteten. Diese wurden in der weiteren Verarbeitung ignoriert.

Der nächste Schritt war das balancieren der Datensätze. Das Vorgehen ist im nächsten Kapitel beschrieben.

Ausgangspunkt für die finale Bearbeitung im Script zur Klassifikation ist ein Data Frame, der einen balancierten Datensatz enthält. Pro Buch sind noch alle Informationen enthalten, die im originalen Datensatz vorhanden waren. Zusätzlich ist eine Spalte mit den eindeutigen Labels vorhanden. Diese Datenstruktur kann nun zur inhaltlichen Vorverarbeitung genutzt werden. In Abbildung 9 sind alle Schritte schematisch dargestellt.

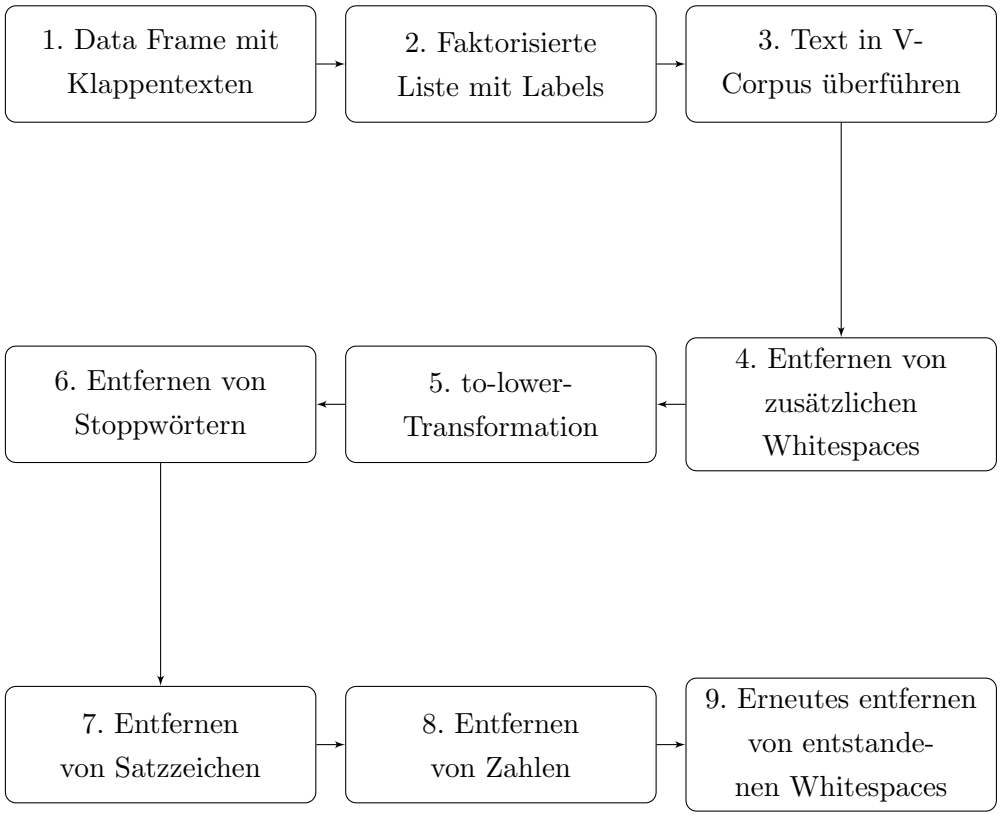

Abbildung 9: Schritte der inhaltlichen Datenvorverarbeitung

Der Data Frame wurde als .RDS eingelesen (zuvor als solche exportiert). Für die Verarbeitung waren lediglich die Klappentexte und die zugehörige Kategorie relevant. Deshalb wurden zwei neue Dataframes erstellt, die jeweils die Texte und die Labels beinhalten. Die Labels wurden faktorisiert, was eine Kategorisierung vereinfacht und im Allgemeinen bei nur zwei Wertausprägungen naheliegt. Die Labels sind so für die Modellierung bereit. Für die weitere Verarbeitung der Texte wurden diese in ein V-Corpus (Volatile - vollständig im Hauptspeicher gehalten) überführt. [2]

Das Entfernen von Whitespaces dient lediglich einer besseren Lesbarkeit und Formatierung des Textes, inhaltliche Relevanz besitzt dieser Schritt nicht. Die to-lower-Transformation dagegen trägt zum Informationsgewinn bei, da Terme nicht zwischen Groß- und Kleinschreibung

[2] Corpus-Methode aus tm-package entnommen, siehe: `https://cran.r-project.org/web/packages/tm/tm.pdf`, letzter Zugriff am 25.07.2019

differenziert werden. Somit werden beispielsweise keine Termfrequenzen verfälscht, wenn ein Term in beiden Varianten auftritt.

Da nun alle Terme eindeutig im Text vorhanden sind, kann ein Stoppwortfilter eingesetzt werden. Dafür wurde eine Kombination aus der deutschsprachigen tm-Liste für Stoppwörter und einer externen Quelle eingesetzt. [3] Durch den Einsatz werden alle Worte, die keinen Informationsgehalt besitzen, vom Corpus entfernt. Somit wird ein Großteil der Terme aus den Daten entfernt. Das Vorgehen ist weiterhin relevant, damit Terme mit hoher Frequenz (Stoppwörter) keinen Einfluss auf das Klassifikationsergebnis haben.

In Tabelle 9 und 10 sind die 15 häufigsten Terme mit der dazugehörigen Frequenz des Trainings- und Evaluationsdatensatzes nach dem Stoppwortfilter aufgelistet.

Tabelle 9: Terme des Trainingsdatensatzes nach dem Anwenden eines Stoppwortfilters

Term	Anzahl im Trainingscorpus
leben	4749
welt	2566
menschen	2286
buch	2164
macht	1812
liebe	1801
zeit	1670
ganz	1598
gibt	1567
frau	1507
jahre	1355
mann	1343
geschichte	1293
jahren	1229
familie	1205

Tabelle 10: Terme des Evaluationsdatensatzes nach dem Anwenden eines Stoppwortfilters

Term	Anzahl im Evaluationscorpus
leben	708
welt	406
menschen	335
buch	291
macht	270
liebe	252
zeit	242
ganz	241
frau	223
gibt	219
mann	211
geschichte	187
jahren	179
junge	179
jahre	178

Da die Morphologie des Textes für die angewandten Klassifikationsmodelle irrelevant sind, wurden auch Satzzeichen und mögliche Sonderzeichen entfernt. Die Modelle basieren lediglich auf den Termen und deren Frequenzen. Auch Zahlen sind weniger von Bedeutung, weshalb sie ebenfalls entfernt wurden. Durch das häufige Entfernen von Objekten im Text können erneut Whitespaces entstehen, die zum Abschluss noch einmal herausgefiltert werden.

Mit diesen Schritten ist die inhaltliche Verarbeitung abgeschlossen. In Abbildung 10 sind die nun folgenden Schritte bis zur Modellierung dargestellt.

[3] Eingesetzte Liste entommen von: `https://github.com/solariz/german\_stopwords/blob/master/german\_stopwords\_plain.txt`, Letzter Zugriff am 25.07.2019

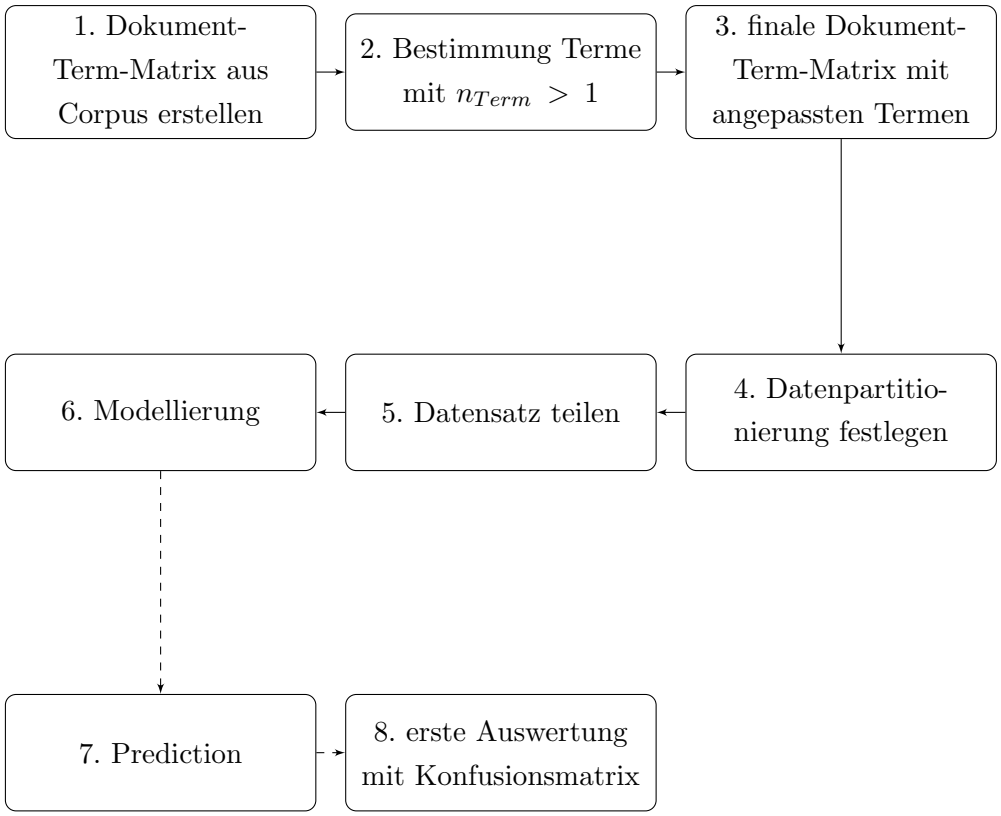

Abbildung 10: Schritte der Datenverarbeitung bis zur Klassifikation

Nach dem alle Schritte der Vorverarbeitung im Corpus ausgeführt wurden, kann eine Dokument-Term-Matrix aus diesem erstellt werden. Je nach gewähltem Feature wird die Matrix mit Bitvektoren, Termfrequenzen oder TF-IDF gefüllt. Für die erste Matrix wurden Termfrequenzen gewählt, damit Terme gefunden werden, die nur einmal im Corpus auftreten. Im Grunde werden bei der Klassifikation Terme gesucht, die spezifisch für eine bestimmte Klasse sind. Sind alle Terme bestimmt, die die Mindestanzahl besitzen, können diese als zusätzlicher Listenparameter genutzt werden und eine neue, finale Dokument-Term-Matrix wird erstellt (mit dem gewünschten Feature). [4] Diese Matrix kann sehr speicherintensiv werden, abhängig von der Größe der Ausgangsdaten.

Im Falle, dass nur ein Datensatz zur Verfügung steht, was bei den ersten Berechnungen mit den Trainingsdaten der Fall war, müssen die Dokumente in Trainings- und Testdaten aufgeteilt werden. Dazu wird entweder eine einfache Datenpartitionierung [5] oder eine Crossvalidation gewählt. Sind die Daten entsprechend dem Partitionierungsschema geteilt, wird die Modellberechnung mit den Trainingsdaten gestartet. Nach der Berechnung wird eine Vorhersage der Labels für die Testdaten generiert. Falls die Testdaten schon ein Label besitzen, kann die Vorhersage mit den tatsächlichen Labels in einer Konfusionsmatrix verglichen werden.

An dieser Stelle muss noch eine weitere mögliche Vorverarbeitung vorgestellt werden. Die später angewandten Methoden und Modelle beinhalten n-Grams und die Kombination von n-

[4] Mithilfe der findFreqTerms-Funktion: `https://www.rdocumentation.org/packages/tm/versions/0.7-6/topics/findFreqTerms`, letzter Zugriff: 28.07.2019

[5] beispielsweise mit: `https://www.rdocumentation.org/packages/caret/versions/6.0-84/topics/createDataPartition`, letzter Zugriff: 28.07.2019

Grams und Termfrequenzen. Für die Generierung von n-Grams muss ein Tokenizer verwendet werden. [6] Dieser kann als Parameter die Anzahl von Termen (n_{term}) entgegennehmen und wird als Funktion in der Erstellung der Dokument-Term-Matrix aufgerufen. Danach werden wieder nur n-Grams verwendet, deren Auftreten größer als Zwei ist. Nach dieser Selektion wird die Matrix der n-Grams für die Klassifikation verwendet.

Sind Terme und n-Grams als Feature zu benutzen, werden die beschrieben Schritte zur Erstellung von Dokument-Term-Matrizen mit Termen und n-Grams jeweils separat ausgeführt und anschließend ein Merge zwischen diesen ausgeführt. Dabei stellen die Zeilen immer noch die Dokumente dar, die erste Hälfte der Spalten das erste Feature (z.B. die Terme) und die zweite Hälfte die n-Grams. In Tabelle 11 ist dieser Sachverhalt schematisch dargestellt.

Tabelle 11: Beispiel einer erweiterten Dokument-Term-Matrix

	Term 1	...	Term n	n-Gram 1	...	n-Gram n
Dok 1	...	...	...	...	...	...
...	...	...	...	...	...	...
Dok n	...	...	...	...	...	...

Neben den hier aufgezählten Schritten, existieren zwei weitere mögliche Vorverarbeitungen. Das Stemming kann dazu eingesetzt werden, um verschiedene Wortformen eines Terms auf dieselbe Basisform abzubilden. Dadurch sollen Precision und Recall verbessert werden. Da das Stemming mehrere Terme auf eine Stem-Form abbildet, kann auch die Datensatzgröße reduziert werden. Bei der Stem-Form handelt es sich um eine Formatierung, bei der nur die variable Endung eines Terms abgeschnitten wird. Dieser Schritt ist noch vor dem Stoppwortfilter anzusiedeln, damit keine Stoppwörter ignoriert werden, nur weil sie in einer nicht aufgelisteten Form vorliegen. [16]

Eine andere Form ist das Lemmatisieren. Dabei werden Wörter in die Wörterbuchform überführt. Beispielsweise bei Substantiven wird die Singular-Form substituiert oder bei Verben die finite Form. Der Nutzen ist derselbe wie beim Stemming, nur das hierfür eine Datenbank (Dictionary etc.) benötigt wird, in der alle Formen vorhanden sind. [17]

Beide Varianten haben Vorteile, die in vielen Fällen eine Einbindung rechtfertigen. Dennoch ist mit diesen Methoden immer ein Informationsverlust verbunden, da die Varianz in den Termen verloren geht. Für die Berechnung der Modelle wird weitgehend auf die Nutzung eines Stemmers oder der Lemmatisierung verzichtet. Ein Beispiel der Auswirkung wird in Kapitel 5.2.6 gezeigt.

4.2 Balancierte Datensätze

Wie im vorangegangen Kapitel schon kurz erwähnt, wurde nicht der gesamte Datensatz pro Berechnung verwendet. Pro Kategorie wurde eine Binarisierung und Balancierung des Datensatzes angewandt.

[6] `https://cran.r-project.org/web/packages/RWeka/RWeka.pdf`, letzter Zugriff: 28.07.2019

Binarisierung

Bei der Einordnung der Bücher in die acht Kategorien handelt es sich um ein Multi-Klassen-Problem, da bei jeder Entscheidung eine der acht Klassen ausgewählt werden muss. Es existieren Modelle, wie zum Beispiel die eingesetzte Naive Bayes Implementierung, die Multi-Class-Prediciton erlauben. Dabei wird über die Wahrscheinlichkeiten, zu welcher Klasse ein Buch gehört, die dominierende ausgewählt. Ein solches Vorgehen ist beispielsweise OVA - die One-Versus-All Methode. Dabei wird ein Multiklassen-Problem vereinfacht, in dem nur noch binäre Entscheidungen betrachtet werden. Es wird also jede Möglichkeit gegen jede andere betrachtet. Die finale Entscheidung der Klasse ist eine Maximierungsfunktion, die höchste Wahrscheinlichkeit wird genutzt. [18]

Um einen genaueren Eindruck von den Klassifikationsergebnissen zu bekommen und beurteilen zu können, warum welches Buch zu welcher Klasse zugeordnet wird, ist eine komplett binäre Klassifikation möglich. In den späteren Berechnungen wurde ein Modell jeweils auf acht Klassen angewandt und entschied pro Klasse lediglich binär, ob die aktuelle Klasse vorliegt oder nicht. Dieses Vorgehen ist eine verbreitete Methode um Multi-Class-Prediction zu vereinfachen [19]. Ist eine automatisierte Klassifikation aller Bücher über eine binäre Herangehensweise gefordert, so muss nach allen Klassifikationen das Problem der Mehrfachzuordnung gelöst werden.

Dazu waren neue Klassen-Listen nötig. Bei der Klassifikation der Architektur-Bücher bekamen alle Bücher, die zur Architektur-Klasse zählen eine 1 und alle anderen Bücher, egal zu welcher Klasse sie gehören, eine 0. Dieses Vorgehen auf alle acht Klassen angewandt ergibt acht verschiedene Listen, die das Label jeweils eindeutig zuordnen.

Balancierung

Der zweite Schritt war das Balancieren des Datensatzes. Neben dem Ausgleichen der Klassenungleichgewichte standen zu Beginn der Modellierung die Hardwaregrenzen im Vordergrund. Wie bei der Einführung der Dokument-Term-Matrizen erwähnt, können solche Strukturen sehr speicherintensiv werden. Die ersten Berechnungen wurden auf einem lokalen PC ausgeführt. Bereits nach den ersten Versuchen führten die zu hohen Speicher- und Zeitbeanspruchungen zu Problemen. Deshalb wurde ein Rechenserver eingesetzt, der sehr viel größere Ressourcen bereitstellte.

Vermutlich ist die Speicherverwaltung und die Struktur der Daten in R weniger optimiert, als es beispielsweise in Python der Fall ist. Manche Berechnungen benötigen übermäßig viel Zeit und Speicher. Eine Rechendauer von fünf oder mehr Tagen für die Literaturklasse mit einer Hauptspeicherbelegung von ca. 60 GB waren keine Ausnahme.

Um neben der Hardwareverbesserung die Performance zu beeinflussen, wurden balancierte Datensätze verwendet. Die Erstellung wird am Beispiel der Architektur-Klasse kurz erläutert, da das Vorgehen maßgeblichen Einfluss auf die Klassifikationsergebnisse besitzt.

Die Architektur-Klasse besitzt 127 Bücher, welche komplett in den neuen, balancierten Datensatz übernommen wurden. Um eine Balance zu erreichen, wurden zu den positiven Einträgen dieselbe Anzahl an Texten (wieder 127) von den anderen Klassen hinzugefügt. Die Auswahl erfolge zufällig aus dem gesamten Datensatz aller Bücher (einzige Voraussetzung war eine andere Klasse als Architektur). Somit entstand ein Datensatz mit $2 \cdot 127 = 254$ Büchern.

Im Vergleich dazu würde der Datensatz ohne die Balancierung eine Größe von knapp 15.000 Büchern aufweisen. [20]

Neben der Balancierung für eine Performanceverbesserung, ist dieses Vorgehen inhaltlich relevant. Der Hintergrund wurde im Kapitel zur Theorie der Textklassifikation erläutert.

5 Klassifikationsergebnisse

In diesem Kapitel werden die Ergebnisse der Klassifikation vorgestellt. Dabei teilt sich die Auswertung in Trainingsdaten und Evaluationsdaten. Es werden jeweils die Feature mit deren Einfluss auf das Ergebnis präsentiert. Einleitend folgt die Vorstellung der Implementierungen.

5.1 Eingesetzte Modelle

Wie der Theorie dieser Arbeit zu entnehmen, wurden Naive Bayes, Support Vector Machine, k-Nearest Neighbor und Random Forest als Modelle gewählt. Die Modellberechnung erfolgte in der Scriptsprache R.

Die ersten Versuche der Modellierung wurden mit dem Caret-Packet durchgeführt. [7] Für die Naive Bayes Implementierung wurde die nb-Methode verwendet, die SVM wurde mit der svmLinear2-Implementierung umgesetzt und für den kNN wurde die knn-Implementierung verwendet (jeweils im Caret-Paket vorhanden).

Die Resultate der ersten Versuche waren nicht zufriedenstellend, da die jeweiligen F_1-Maße zu schlecht waren. Zusätzlich waren die benötigten Zeiten für die Modellerstellung sehr lang.

Deshalb wurden andere Implementierungen gewählt, die nach einem Test deutlich verbesserte Ergebnisse lieferten. Alle Modelle wurden ohne die Caret-Umgebung berechnet. So wurde das fast Naive Bayes Package [8], eine andere SVM-Implementierung aus dem e1071-Package [9] und eine Random Forest Methode [10] gewählt.

5.2 Trainingsdaten

In den folgenden Kapiteln wird jedes einzelne Feature tabellarisch und graphisch ausgewertet und kurz erläutert. Bei den zu klassifizierenden Daten handelt es sich ausschließlich um Trainingsdaten, welche in Trainings- und Testdaten geteilt wurden. In jeder Tabelle finden sich die jeweiligen Werte für Precision (P), Recall (R) und das F_1-Maß pro Klasse und Modell. Das dazugehörige Diagramm visualisiert den F_1-Verlauf als Balkendiagramm. Am Ende des Kapitels zu den Trainingsdaten werden die Feature und Modelle untereinander verglichen.

5.2.1 Bitvektor

Das erste Feature ist der Bitvektor, bei dem nur das alleinige Auftreten eines Wortes für die Klassifikation relevant ist. Die Übersicht über die Klassifikationsergebnisse sind in Tabelle 12 und Abbildung 11 zu finden.

[7] Webreferenz: `https://cran.r-project.org/web/packages/caret/caret.pdf`, letzter Zugriff: 30.07.2019
[8] `https://cran.r-project.org/web/packages/fastNaiveBayes/index.html`, letzter Zugriff: 30.07.2019
[9] `https://cran.r-project.org/web/packages/e1071/e1071.pdf`, letzter Zugriff: 30.07.2019
[10] `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`, letzter Zugriff: 30.07.2019

Tabelle 12: Klassifikationsergebnis der Trainingsdaten - Bitvektor

	fast Naive Bayes			SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.8571	0.9600	**0.9057**	0.8696	0.8000	0.8333	0.8750	0.8400	0.8571
Ganzheitliches Bewusstsein	0.8633	0.9524	0.9057	0.9328	0.8810	**0.9061**	0.8906	0.9048	0.8976
Glaube & Ethik	0.8738	0.9474	**0.9091**	0.8750	0.8105	0.8415	0.9101	0.8526	0.8804
Kinder- & Jugendbuch	0.8560	0.8774	**0.8666**	0.8159	0.8273	0.8216	0.8093	0.8273	0.8181
Künste	0.9565	0.9167	**0.9362**	0.9565	0.9167	**0.9362**	0.9524	0.8333	0.8889
Literatur & Unterhaltung	0.8431	0.9092	**0.8749**	0.8222	0.9256	0.8709	0.8144	0.8466	0.8302
Ratgeber	0.8787	0.9252	**0.9014**	0.8896	0.8785	0.8840	0.8580	0.8660	0.8620
Sachbuch	0.8153	0.8542	**0.8344**	0.8113	0.7563	0.7828	0.7991	0.8492	0.8234
Mittelwert	0.8680	0.9178	**0.8918**	0.8716	0.8495	0.8596	0.8636	0.8525	0.8572

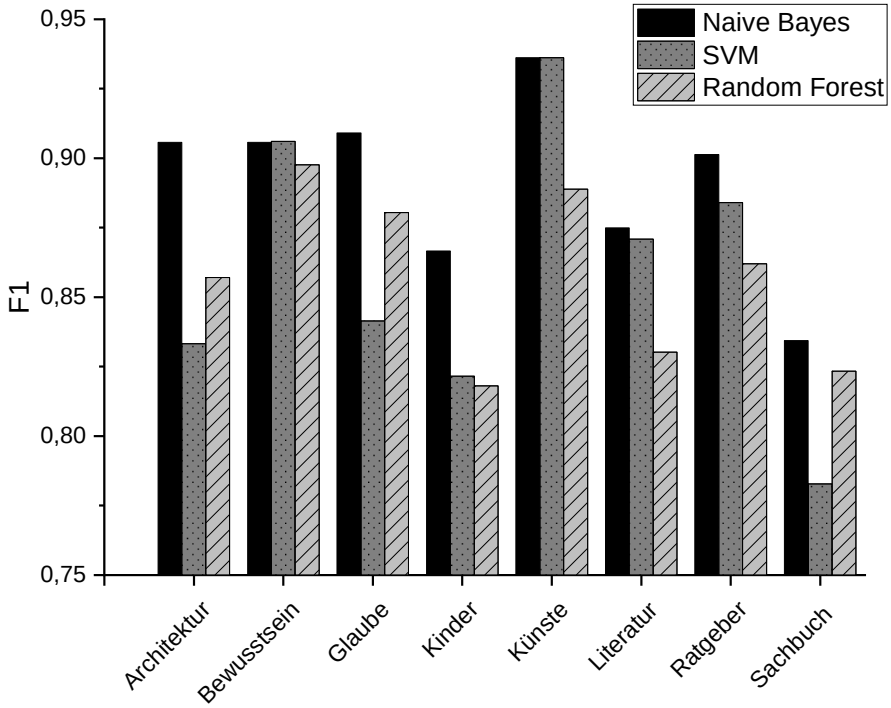

Abbildung 11: Klassifikationsergebnis der Trainingsdaten - Bitvektor

Allgemein sind die Resultate sehr gut, vor allem im Hinblick auf das gewählte Feature, da es eines der einfachsten ist. Das beste Modell ist Naive Bayes mit einem F_1-Maß von 89,18 %. Im Mittel liegen die F_1-Maße der drei Modelle weniger als 5% voneinander entfernt. Dabei liegen die Mittelwerte der Modelle SVM und Random Forest besonders nahe beieinander. Precision und Recall zeigen keine größeren Differenzen oder einen Trend zu einer von beiden Maßzahlen. Das geringste F_1-Maß ist bei Naive Bayes und der SVM die Klasse Sachbuch, bei Random Forest ist es Kinder- und Jugendbuch.

Das besonders hohe F_1-Maß in Künste ist bemerkenswert, da diese Klasse mit nur 133 Büchern sehr klein ist. Es besteht dadurch die Gefahr der Überanpassung des Modells, vor allem bei der SVM.

Alle drei Modelle liefern nahezu identische Klassifikationsergebnisse in der Kategorie Bewusstsein.

Des Weiteren ist auffällig, dass durch die SVM die Kategorie Architektur am schlechtesten, die Kategorie Künste jedoch am besten berechnet wird. Trotz des geringen Umfangs, gilt die Annahme, dass die Terme für Kunst spezifischer als die Terme für Architektur sind, sodass sie im Raum besser zu separieren sind.

5.2.2 Termfrequenz

Das nächste Feature ist die Termfrequenz, bei der die Anzahl der Terme mit berücksichtigt wird. In Tabelle 13 und Abbildung 12 sind die Ergebnisse zusammengefasst.

Tabelle 13: Klassifikationsergebnis der Trainingsdaten - TF

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.8333	1.000	0.9091	0.8696	0.8000	0.8333	1.000	0.8400	**0.9130**
Ganzheitliches Bewusstsein	0.8696	0.9524	0.9091	0.9350	0.9127	**0.9237**	0.8943	0.8730	0.8835
Glaube & Ethik	0.8750	0.9579	**0.9146**	0.8723	0.8632	0.8677	0.9367	0.7789	0.8506
Kinder- & Jugendbuch	0.8726	0.8774	**0.8750**	0.8333	0.8217	0.8275	0.8197	0.8367	0.8276
Künste	1.000	0.8750	0.9333	1.000	0.9167	**0.9565**	1.000	0.8333	0.9091
Literatur & Unterhaltung	0.8482	0.8973	**0.8720**	0.8189	0.9171	0.8652	0.8254	0.8466	0.8359
Ratgeber	0.8713	0.9283	**0.8989**	0.8949	0.8754	0.8850	0.8871	0.8816	0.8844
Sachbuch	0.8105	0.8166	0.8135	0.8090	0.7663	0.7871	0.8060	0.8769	**0.8400**
Mittelwert	0.8726	0.9131	**0.8907**	0.8791	0.8591	0.8682	0.8962	0.8459	0.8680

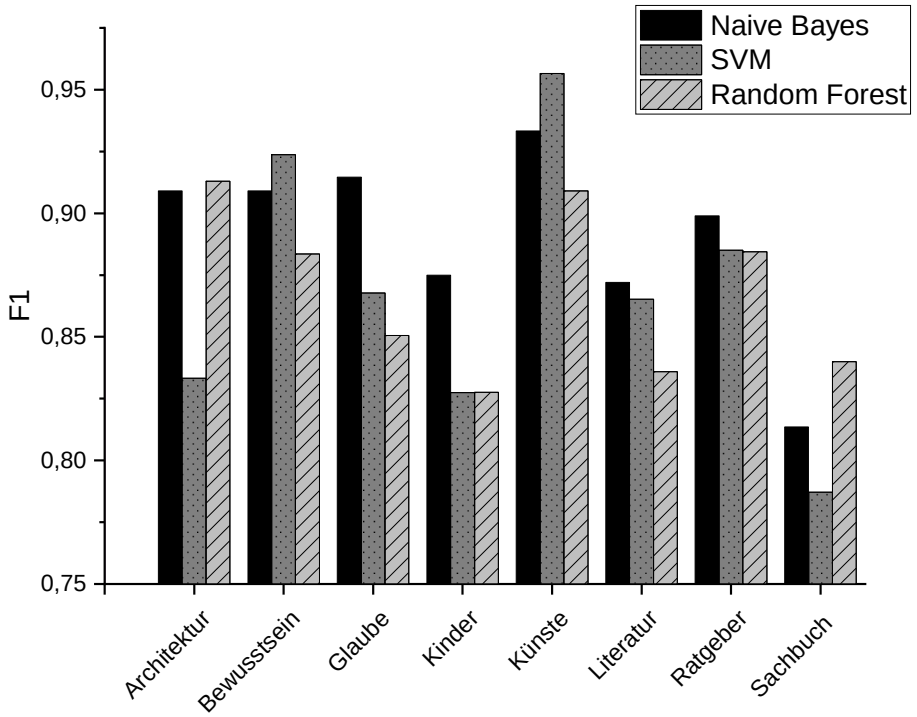

Abbildung 12: Klassifikationsergebnis der Trainingsdaten - TF

Das beste Modell ist Naive Bayes mit einem gemittelten F_1- Maß von 89,07%. Alle Modelle liefern sehr gute Ergebnisse und liegen nahe beieinander, da sie sich im mittleren F_1- Maß nur um ca. 3% voneinander unterscheiden, wobei SVM und Random Forest annähernd denselben Mittelwert besitzen.

Auffällig ist die Klassifikation der Kategorie Künste, da alle drei Modelle eine Precision von 1 liefern und somit ein sehr hohes F_1 Maß resultiert. Erneut liegen Precision und Recall bei hohen Werten, sodass kein Trend erkennbar ist.

Weniger gut wurden die Kategorien Kinder & Jugendbuch, Literatur und Sachbuch klassifiziert, wobei es sich um die Klassen handelt, die am schwierigsten zu spezifizieren sind.

Die Beziehung zwischen SVM und den Kategorien Künste und Architektur ist erneut erkennbar.

5.2.3 Termfrequenz - Inverse Dokumentenfrequenz

Nun folgt eine weitere Verbesserung im Feature, da neben der Termfrequenz auch das Vorkommen in der gesamten Collection berücksichtig wird. In Tabelle 14 und Abbildung 13 sind die Ergebnisse zu sehen.

Tabelle 14: Klassifikationsergebnis der Trainingsdaten - TF-IDF

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.8276	0.9600	**0.8889**	0.8276	0.9600	**0.8889**	1.000	0.8000	**0.8889**
Ganzheitliches Bewusstsein	0.8521	0.9603	0.9030	0.9268	0.9048	**0.9157**	0.8852	0.8571	0.8707
Glaube & Ethik	0.8585	0.9579	**0.9055**	0.8381	0.9261	0.8800	0.9136	0.7789	0.8409
Kinder- & Jugendbuch	0.8575	0.9053	**0.8808**	0.9689	0.4345	0.6000	0.7937	0.8468	0.8194
Künste	0.9200	0.9583	**0.9388**	0.8000	1.000	0.8889	1.000	0.8333	0.9091
Literatur & Unterhaltung	0.8407	0.9276	**0.8820**	0.8764	0.7979	0.8353	0.8355	0.8550	0.8451
Ratgeber	0.8584	0.9439	**0.8991**	0.9469	0.7227	0.8198	0.8851	0.8879	0.8865
Sachbuch	0.8141	0.9020	**0.8558**	0.8648	0.5302	0.6573	0.7947	0.8668	0.8293
Mittelwert	0.8536	0.9394	**0.8942**	0.8949	0.7845	0.8107	0.8885	0.8407	0.8612

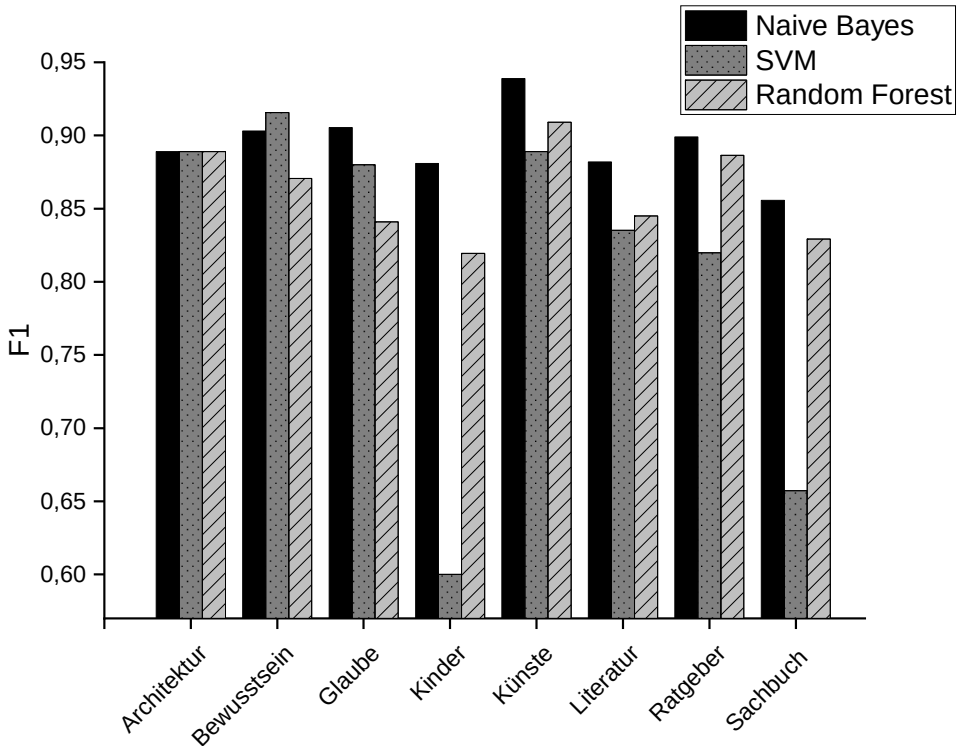

Abbildung 13: Klassifikationsergebnis der Trainingsdaten - TF-IDF

Die Kombination aus Termfrequenz und inverser Dokumentenfrequenz liefert ebenfalls hohe Klassifikationsergebnisse, wobei die Modelle eindeutiger voneinander getrennt liegen (Intervall von 10%). Wieder besitzt Naive Bayes das beste mittlere F_1-Maß mit 89,42%, welches geringfügig besser ist als die alleinige Termfrequenz.

Die SVM gibt schlechte Ergebnisse in den Klassen Kinder & Jugendbuch und Sachbuch, da der Recall wesentlich niedriger als die Precision ist. Die Kategorie Literatur wird vermutlich nur wegen der hohen Anzahl an Dokumenten besser klassifiziert.

Im Vergleich zur alleinigen Termfrequenz ist die Klassifizierung von Architektur und Künste besser.

5.2.4 Bigrams

Das Feature der Bigrams beinhaltet die Kombination aus zwei Termen, wobei das Bigram mit den Frequenzen in die Berechnung eingeht (Abwandlung der Termfrequenz). In Tabelle 15 und Abbildung 14 sind die Ergebnisse dargestellt.

Tabelle 15: Klassifikationsergebnis der Trainingsdaten - Bigrams

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.9167	0.4400	**0.5946**	0.9091	0.4000	0.5556	1.0000	0.2800	0.4375
Ganzheitliches Bewusstsein	0.8598	0.7302	**0.7897**	0.8977	0.6270	0.7383	0.8852	0.4286	0.5775
Glaube & Ethik	0.8400	0.6632	**0.7412**	0.8462	0.5789	0.6875	0.9444	0.3579	0.5191
Kinder- & Jugendbuch	0.8409	0.6184	**0.7127**	0.7554	0.5850	0.6593	0.8431	0.4791	0.6110
Künste	0.9091	0.4167	0.5714	0.8750	0.5833	0.7000	1.0000	0.5833	**0.7368**
Literatur & Unterhaltung	0.8273	0.8953	**0.8599**	0.8199	0.7762	0.7974	0.6550	0.9263	0.7674
Ratgeber	0.8433	0.7882	**0.8148**	0.8836	0.6386	0.7414	0.8882	0.4704	0.6151
Sachbuch	0.7781	0.6960	**0.7347**	0.7938	0.5804	0.6753	0.8100	0.4070	0.5418
Mittelwert	0.8519	0.5787	**0.7274**	0.8476	0.5962	0.6943	0.8782	0.4916	0.6008

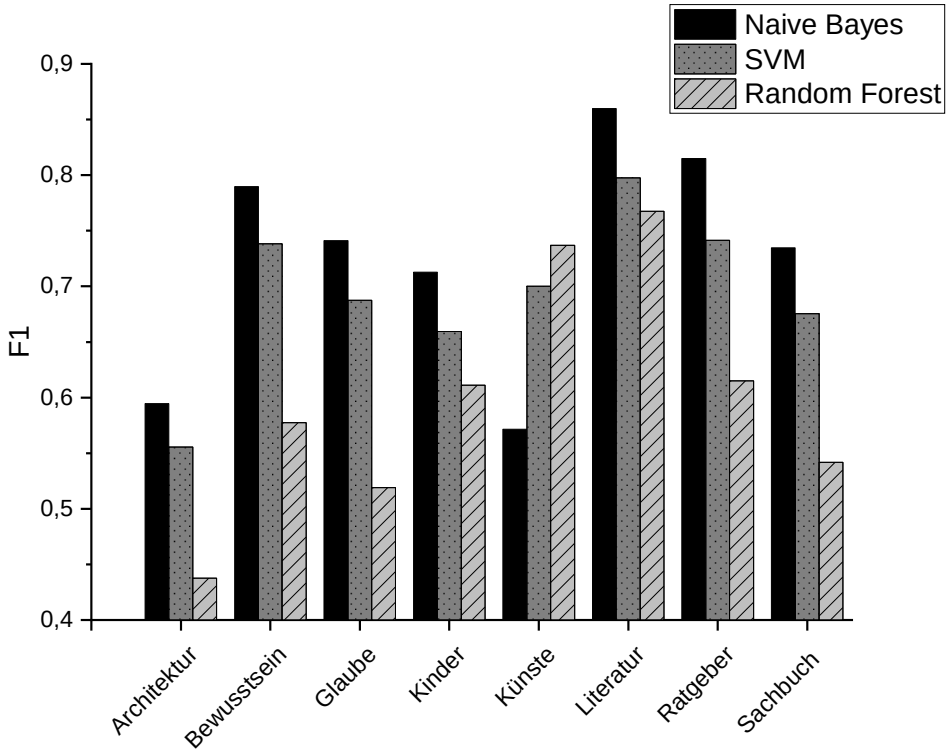

Abbildung 14: Klassifikationsergebnis der Trainingsdaten - Bigrams

Diese Ergebnisse unterscheiden sich deutlich von den vorangegangenen Klassifikationen. Das beste Ergebnis liefert Naive Bayes, gefolgt von der SVM. Das mittlere F_1-Maß von Random Forest liegt deutlich unter den anderen beiden Modellen.

Bei allen drei Modellen liegt der Recall mit ca. 55 % unter den bisherigen Ergebnissen, wodurch das F_1-Maß sinkt.

Ein weiterer wichtiger Fakt ist die andere Verteilung der am besten klassifizierten Kategorien. Die bisher besten Klassen wie Architektur & Garten oder Künste liefern nun sehr schlechte Ergebnisse. Ein möglicher Grund dafür kann im Datensatzumfang liegen. Die beiden Klassen besitzen einen sehr geringen Datenumfang, weshalb auch die Anzahl der Bigrams, die wiederholt auftreten (da auch hier nur Bigrams mit $n > 1$ gilt), gering ist. Im Gegensatz dazu liefert Literatur & Unterhaltung mit das beste Ergebnis, da der Datensatzumfang sehr groß ist und häufiger dieselben Bigrams auftreten.

Der Argumentation folgen ebenso die Werte von Precision und Recall. Bei allen Modellen ist die Precision sehr viel höher, so wurden die positiven Elemente auch als positiv erkannt. Aber die Anzahl der generell erkannten positiven Elemente ist zu gering (niedriger Recall).

5.2.5 Erweiterung der Dokument-Term-Matrix: TF + Bigrams

Eine etwas komplexere Herangehensweise ist die Kombination von Termen und Bigrams. In diesem Fall werden sie über die Frequenzen in der Dokument-Term-Matrix verknüpft, wie in Kapitel 4.2 erläutert. In Tabelle 16 und Abbildung 15 sind die Klassifikationsergebnisse zusammengefasst.

Tabelle 16: Klassifikationsergebnis der Trainingsdaten - Bigrams + TF

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.8571	0.9600	0.9057	0.8626	0.7600	0.8085	1.0000	0.8400	**0.9130**
Ganzheitliches Bewusstsein	0.8769	0.9048	0.8906	0.9426	0.9127	**0.9274**	0.8983	0.8412	0.8688
Glaube & Ethik	0.8866	0.9053	**0.8958**	0.8830	0.8737	0.8783	0.9375	0.7895	0.8571
Kinder- & Jugendbuch	0.8916	0.8022	**0.8446**	0.8494	0.8327	0.8410	0.8270	0.8524	0.8395
Künste	1.0000	0.8750	0.9333	1.0000	0.9167	**0.9565**	1.0000	0.8333	0.9091
Literatur & Unterhaltung	0.8564	0.8677	0.8620	0.8470	0.9184	**0.8812**	0.8222	0.8677	0.8443
Ratgeber	0.8716	0.8879	0.8796	0.8978	0.8754	0.8864	0.8879	0.8879	**0.8879**
Sachbuch	0.8049	0.7462	0.7744	0.8155	0.7663	0.7902	0.8112	0.8744	**0.8420**
Mittelwert	0.8867	0.8686	**0.8733**	0.8872	0.8570	0.8712	0.8980	0.8483	0.8702

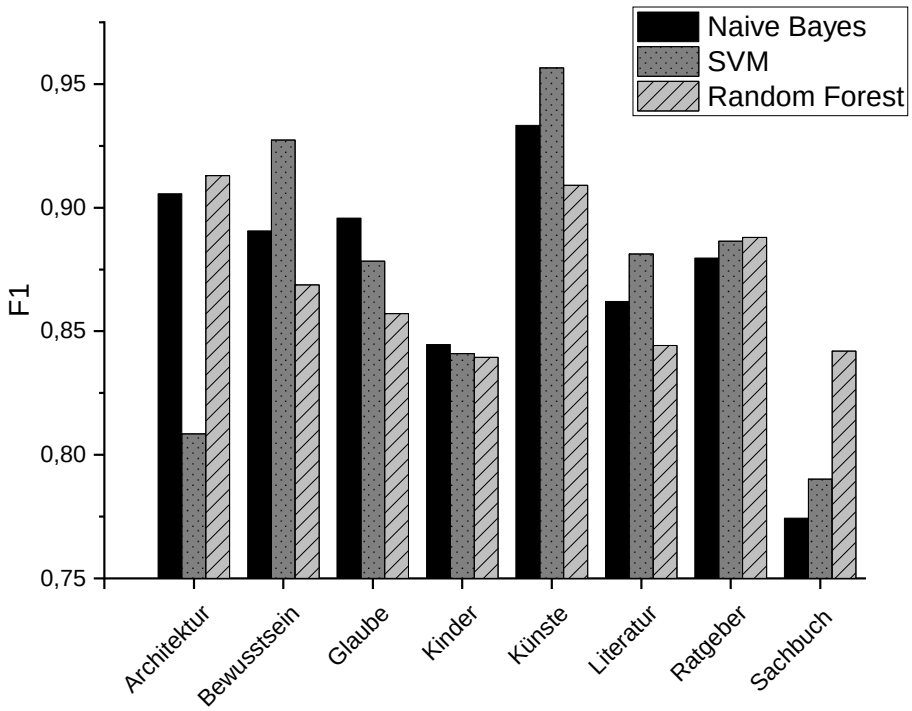

Abbildung 15: Klassifikationsergebnis der Trainingsdaten - Bigrams + TF

Auffallend bei der Kombination beider Feature ist, dass die Methoden kaum unterschiedliche Ergebnisse zeigen. Naive Bayes liefert erneut das beste Ergebnis, obwohl hier die Unabhängigkeit der Daten nicht gegeben ist. Precision und Recall zeigen keine großen Differenzen, nur in der Klasse Künste tritt erneut eine Precision von 1 auf.

Es zeigt sich nochmals die schlechte Bestimmung von Architektur im Gegensatz zu der Kategorie Künste durch die SVM. Sachbuch wurde erneut am unpräzisesten klassifiziert.

5.2.6 Erweiterung der Dokument-Term-Matrix: TF-IDF + Bigrams

Mit dieser Featurekombination wird das vorangegangene Feature noch weiter ausgebaut, in dem TF-IDF mit den Bigrams in einer Dokument-Term-Matrix betrachtet werden. In Tabelle 17 und Abbildung 16 sind die Ergebnisse zusammengefasst.

Tabelle 17: Klassifikationsergebnis der Trainingsdaten - Bigrams + TF-IDF

	fast Naive Bayes			SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.8571	0.7200	0.7826	0.9286	0.5200	0.6667	1.0000	0.8000	**0.8889**
Ganzheitliches Bewusstsein	0.7755	0.9048	0.8352	0.7134	0.9286	0.8069	0.9052	0.8333	**0.8678**
Glaube & Ethik	0.7905	0.8737	0.8300	0.8000	0.9263	**0.8585**	0.9342	0.7474	0.8304
Kinder- & Jugendbuch	0.7914	0.8245	0.8076	0.6992	0.8162	0.7532	0.8207	0.8412	**0.8308**
Künste	0.9167	0.9167	**0.9167**	0.8947	0.7083	0.7907	1.0000	0.8333	0.9091
Literatur & Unterhaltung	0.8547	0.9026	**0.8780**	0.8531	0.8716	0.8623	0.8433	0.8651	0.8541
Ratgeber	0.8213	0.8879	0.8533	0.8023	0.8723	0.8358	0.8952	0.8785	**0.8868**
Sachbuch	0.7736	0.8241	0.7981	0.7272	0.8040	0.7637	0.8028	0.8794	**0.8393**
Mittelwert	0.8226	0.8568	0.8377	0.8023	0.8059	0.7922	0.9002	0.8348	**0.8634**

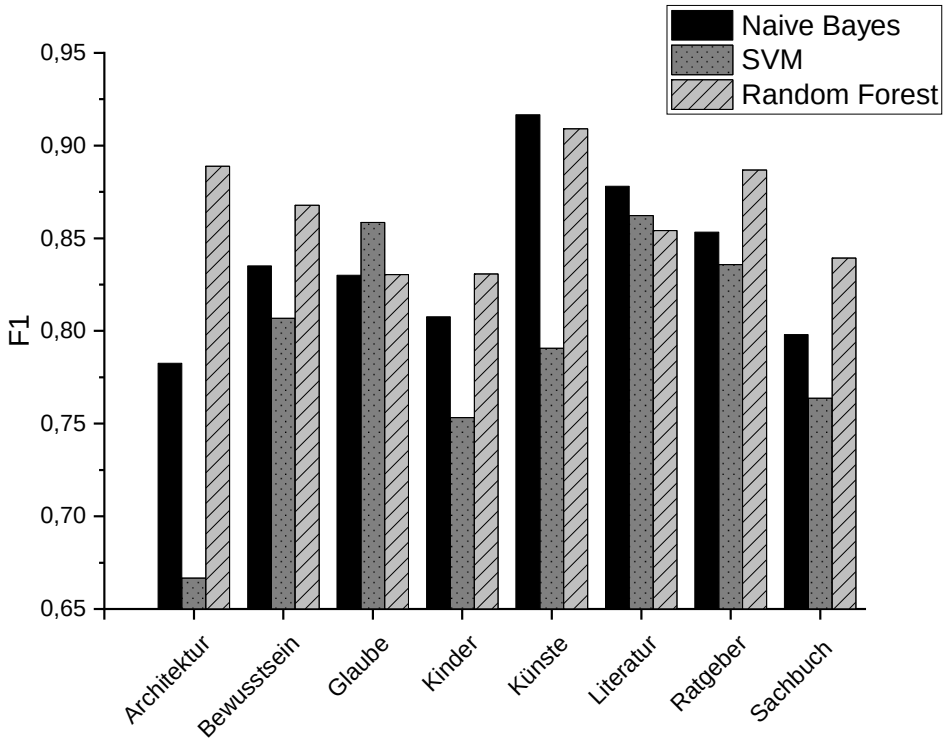

Abbildung 16: Klassifikationsergebnis der Trainingsdaten - Bigrams + TF

Im Allgemeinen schwanken die Mittelwerte der F_1- Maße stärker als bei der vorherigen Kombination der Feature.

Die Ergebnisse des Random Forest verhalten sich weitestgehend stabil.

Die SVM liefert im gemittelten F_1-Maß die schlechtesten Ergebnisse. Sie klassifiziert jede Kategorie schlechter als die beiden anderen Modelle (mit Ausnahme von Glaube).

Der Naive Bayes Klassifikator liefert trotz wiederholter Verletzung der Unabhängigkeitsvorraussetzung ebenso stabile Ergebnisse.

Die Kategorie Literatur wird von allen Modellen annähernd gleich gut bestimmt, was wieder am hohen Datensatzumfang liegen kann. Dagegen liefern die kleinen Klassen wie Architektur und Künste sehr verschiedene Ergebnisse: die SVM labelt die Klassen sehr unkorrekt, Random Forest und Naive Bayes dagegen erzielen gute Resultate.

5.2.7 Stemming

Dieser Abschnitt zeigt zum Abschluss der Klassifikation der Trainingsdaten den Effekt eines Stemmers. Dabei wird als Feature die Termfrequenz gewählt und in die Modellierung das Stemming als Teil der Vorverarbeitung eingebaut. In Tabelle 18 und Abbildung 17 sind die Ergebnisse zusammengefasst.

Tabelle 18: Klassifikationsergebnis der Trainingsdaten - TF + Stemming

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.8519	0.9200	0.8846	0.9048	0.7600	0.8261	1.0000	0.8400	**0.9130**
Ganzheitliches Bewusstsein	0.8561	0.9444	0.8981	0.9492	0.8889	**0.9180**	0.9040	0.8968	0.9004
Glaube & Ethik	0.8679	0.9684	**0.9154**	0.8913	0.8632	0.8770	0.9250	0.7789	0.8457
Kinder- & Jugendbuch	0.8521	0.8830	**0.8673**	0.8438	0.8273	0.8354	0.8113	0.8384	0.8247
Künste	0.9167	0.9167	**0.9167**	0.9545	0.8750	0.9130	1.0000	0.7917	0.8837
Literatur & Unterhaltung	0.8427	0.8920	**0.8666**	0.8142	0.9118	0.8602	0.8389	0.8511	0.8450
Ratgeber	0.8746	0.9128	0.8933	0.8972	0.8972	**0.8972**	0.8697	0.8941	0.8817
Sachbuch	0.8086	0.8065	0.8075	0.8194	0.7638	0.7906	0.8037	0.8744	**0.8375**
Mittelwert	0.8588	0.9055	**0.8812**	0.8843	0.8484	0.8647	0.8941	0.8457	0.8665

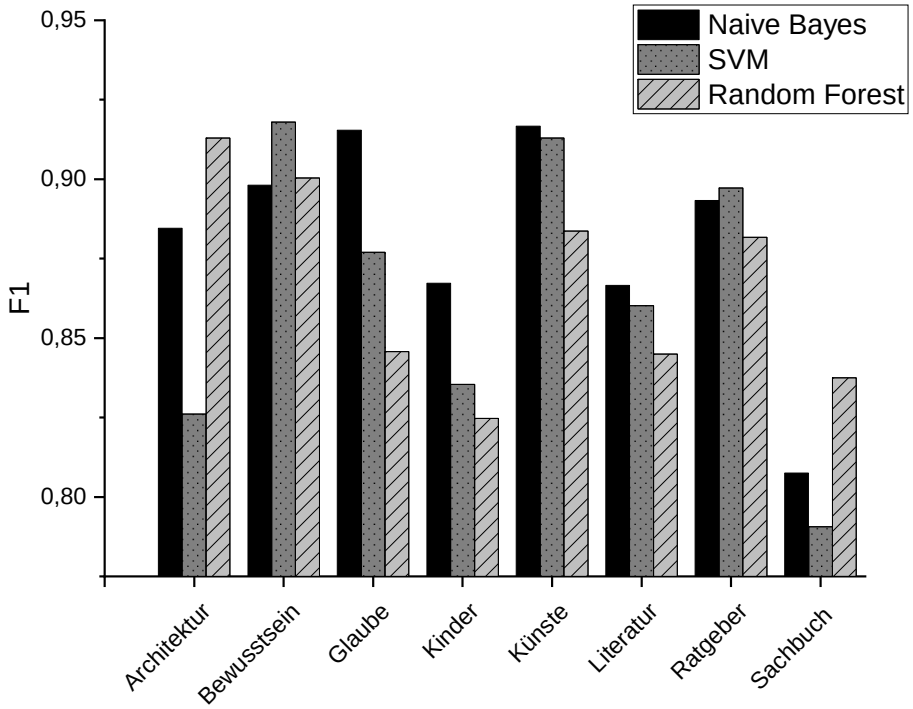

Abbildung 17: Klassifikationsergebnis der Trainingsdaten - TF + Stemming

Im Vergleich zur reinen Termfrequenz als Feature, wird in diesem Fall die Klassifikation unter Verwendung eines Stemmers nicht verbessert. Die Werte unterscheiden sich zur Modellierung ohne Stemmer nur sehr geringfügig um bis zu ca. 1%.

Eine mögliche Erklärung ist in der Stem-Form zu finden. Da beim Stemming oftmals nur die Endung eines Wortes abgeschnitten wird, entsteht nicht automatisch ein Informationsgewinn. Durch die fehlende Endung kann ebenso ein Rauschen entstehen, welche eine eindeutige Klassifikation erschweren. Eine Verbesserung kann an dieser Stelle eine Lemmatisierung liefern, bei der Wörter auf ihr Wörterbuchform abgebildet werden.

5.2.8 Zusammenfassung

In Abbildung 18 sind alle mittleren F_1-Maße pro Feature und Modell graphisch zusammengefasst.

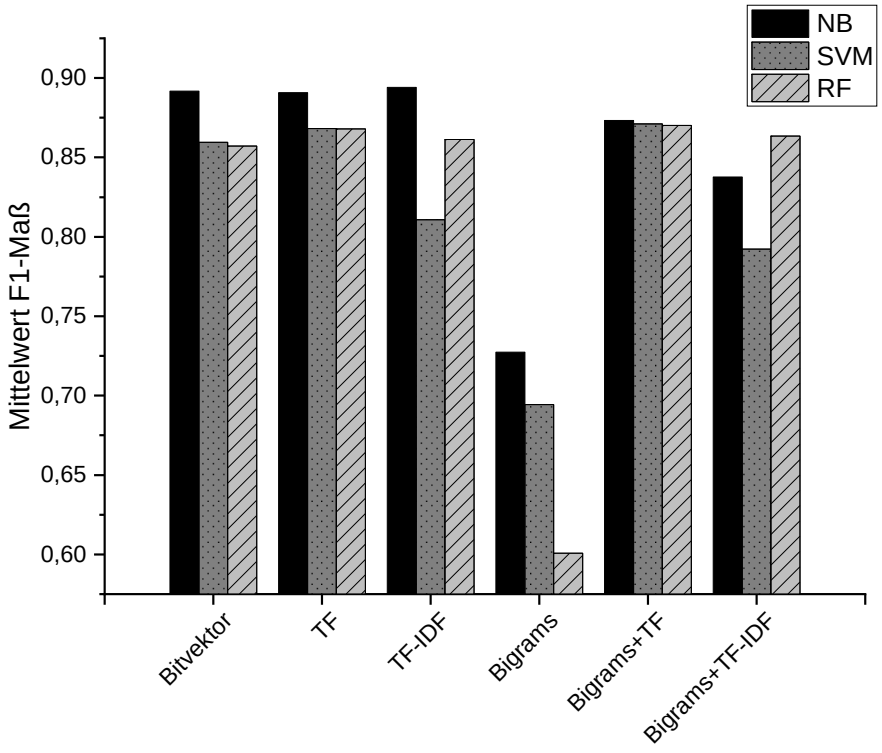

Abbildung 18: Zusammenfassung der mittleren F_1-Maße pro Feature und Modell

Die Klassifikationsergebnisse der Trainingsdaten sind insgesamt sehr gut, lediglich die Klassifikation mit den Bigrams stellt eine Ausnahme dar. Durch diese als alleiniges Feature lieferten alle Modelle das schlechteste Ergebnis. Eine mögliche Erklärung lässt sich in der Textstruktur finden. Schlüsselterme, die jeweils pro Klasse spezifisch sind, treten sehr viel häufiger alleine als in immer wiederkehrender Kombination mit anderen Termen auf. Durch das alleinige Verwenden von Bigrams, geht die Spezifität der Schlüsselworte verloren und es existieren deutlich weniger charakteristische Terme bzw. Bigrams. [21]

Werden jedoch bestehende Feature mit Bigrams erweitert, wird der Nutzen beider Feature

in die Klassifikation eingehen. Schlüsselterme werden zur Spezifizierung der Klassen beitragen und wiederkehrende Bigrams festigen das Klassifikationsergebnis.

Trotz der steigender Komplexität der Feature, folgen die Klassifikationsergebnisse keinem Trend, was in Abbildung 18 sichtbar ist.

Das im Mittel beste Ergebnis liefert der Naive Bayes Klassifikator. An dieser Stelle sei nochmal die verletzte Voraussetzung der unabhängigen Merkmale erwähnt. Vor allem bei den kombinierten Feature sind die dennoch hohen Ergebnisse bemerkenswert.

Sie SVM liefert bei der Kombination von Bigrams und Termfrequenz das beste Ergebnis, auch wenn das Ergebnis nur um wenige Prozentpunkte überwiegt. Der Random Forest Algorithmus liefert die stabilsten Ergebnisse, was ebenso bedeutet, dass die komplexer werdenden Feature kaum Einfluss auf das Ergebnis haben.

Die Klassen Kinder- & Jugendbuch, Literatur und Sachbücher sind am schwierigsten zu klassifizieren, da deren Terme sehr ähnlich und wenig charakteristisch sind. Dennoch sind die Ergebnisse vor allem bei Literatur durchgängig hoch, was an dem sehr großen Anteil der Bücher liegen kann.

Im Gegensatz dazu stehen die Klassen Architektur und Künste, die nur einen Bruchteil des Umfangs besitzen. Trotzdem werden auch diese gut klassifiziert, was aber an den Termen selbst liegt. Diese sind weitaus charakteristischer und erlauben so eine bessere Klassifikation.

5.3 Evaluationsdaten

In diesem Kapitel werden die Klassifikationsergebnisse der Evaluationsdaten erläutert. Dabei teilt sich jedes Kapitel in drei Auswertungen: voller Trainingsdatenumfang, Undersampling und Oversampling der Trainingsdaten. Mit den jeweils vorverarbeiteten Trainingsdaten wird das Modell trainiert und damit werden im Anschluss die kompletten Evaluationsdaten klassifiziert.

5.3.1 Bitvektor

Komplette Trainingsdaten

Die erste Klassifikation ist die Berechnung des Modells mit dem kompletten Trainingsdatenumfang ohne Balancierung und dem Bitvektor als Feature. In Tabelle 19 und Abbildung 19 sind die Klassifikationsergebnisse zusammengefasst.

Tabelle 19: Klassifikationsergebnis der Evaluationsdaten - Bitvektor - Full

	fast Naive Bayes			SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.5000	0.2353	0.3200	1.0000	0.5294	**0.6923**	1.0000	0.0588	0.1111
Ganzheitliches Bewusstsein	0.5700	0.6552	**0.6096**	0.6184	0.5402	0.5767	0.6154	0.1839	0.2832
Glaube & Ethik	0.7736	0.5694	**0.6560**	0.2073	0.7083	0.3208	0.8696	0.2778	0.4211
Kinder- & Jugendbuch	0.7407	0.6038	**0.6653**	0.2190	0.7132	0.3351	0.8316	0.2981	0.4389
Künste	0.6667	0.2667	0.3810	1.0000	0.4000	**0.5714**	1.0000	0.1333	0.2353
Literatur & Unterhaltung	0.8462	0.9132	**0.8784**	0.8470	0.8573	0.8521	0.8445	0.7312	0.7838
Ratgeber	0.5875	0.7911	**0.6742**	0.2153	0.7111	0.3306	0.7083	0.3022	0.4237
Sachbuch	0.5608	0.6342	**0.5953**	0.2076	0.6409	0.3136	0.5700	0.1913	0.2864
Mittelwert	0.6557	0.5836	**0.5975**	0.5393	0.6376	0.4991	0.8049	0.2721	0.3729

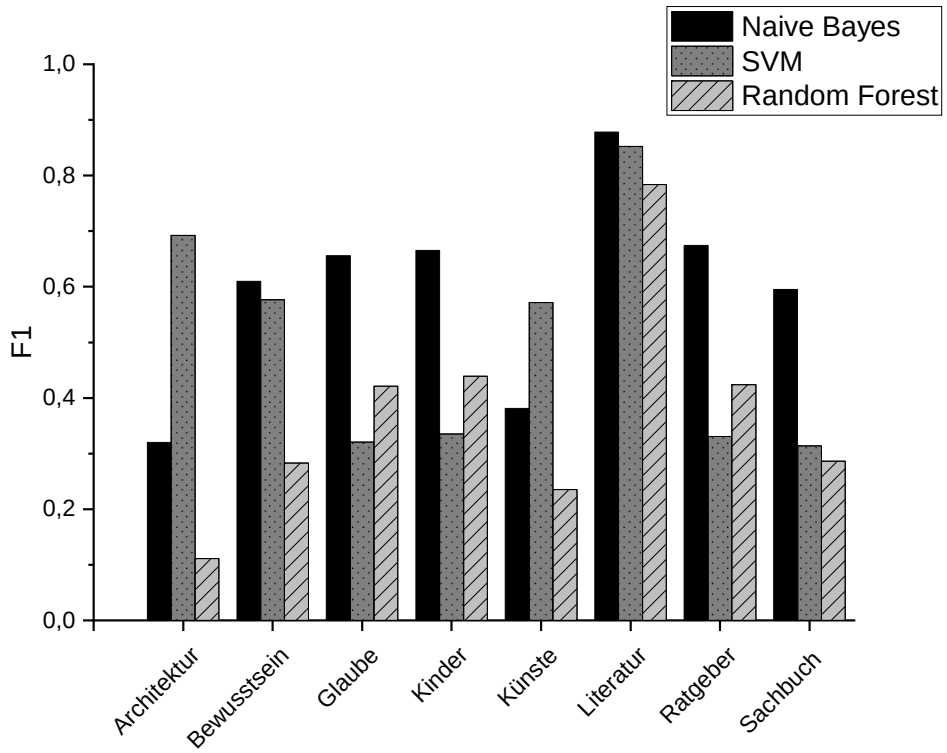

Abbildung 19: Klassifikationsergebnis der Evaluationsdaten - Bitvektor - Full

Die gemittelten F_1-Maße sind sehr niedrig, vor allem im Vergleich zu den gesamten Klassifikationen des vorangegangenen Kapitels.

Der Naive Bayes Algorithmus erzielt das beste Ergebnis, Random Forest dagegen liegt im gemittelten F_1-Maß ca. 20 % darunter.

Die Werte für Precision und Recall sind bei Naive Bayes und der SVM ähnlich, bei Random Forest dagegen ist der Trend zu einer deutlich höheren Precision erkennbar.

Auffällig ist das hohe F_1-Maß bei Literatur, was wieder am hohen Datensatzumfang liegt. Dabei liegt das Ergebnis auf demselben Niveau wie die bisherigen.

Dagegen werden die kleinen Klassen (Architektur und Künste) sehr schlecht klassifiziert. Die SVM liefert hier trotzdem die höchsten Ergebnisse. Die Precision ist bei den Modellen SVM und Random Forest bei eins.

Ein Klassifikationsergebnis von ca. 50 % oder schlechter hat für die Praxis keine Relevanz, da alle Ergebnisse ebenso auf Zufall beruhen können. Deshalb sind die Modelle von SVM und Random Forest nicht aussagekräftig. Das gilt ebenso für alle weiteren Klassifikationen mit F_1-Maßen unter 50 %.

Undersampling der Trainingsdaten

In diesem Abschnitt wird die Auswirkung des Undersamplings betrachtet. Dabei wird der Trainingsdatensatz wie in Kapitel 4.2 angepasst und weiterhin der Bitvektor als Feature verwendet. In Tabelle 20 und Abbildung 20 sind die Klassifikationsergebnisse dargestellt.

Tabelle 20: Klassifikationsergebnis der Evaluationsdaten - Bitvektor - Undersampling

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.0519	0.9412	0.0985	0.1702	0.9412	0.2883	0.3333	0.9412	**0.4923**
Ganzheitliches Bewusstsein	0.2240	0.9425	0.3620	0.3112	0.8621	**0.4573**	0.2630	0.8736	0.4043
Glaube & Ethik	0.1789	0.9444	0.3009	0.2800	0.8750	**0.4242**	0.2735	0.8889	0.4183
Kinder- & Jugendbuch	0.3979	0.8604	0.5442	0.4483	0.8340	**0.5831**	0.4214	0.8604	0.5658
Künste	0.0389	0.9333	0.0747	0.2308	0.8000	0.3582	0.3667	0.7333	**0.4889**
Literatur & Unterhaltung	0.8463	0.9142	0.8789	0.8436	0.9310	**0.8851**	0.8137	0.8874	0.8490
Ratgeber	0.4471	0.9022	0.5979	0.4525	0.8889	**0.5997**	0.4217	0.8622	0.5664
Sachbuch	0.4194	0.8289	0.5569	0.4479	0.8221	**0.5799**	0.3391	0.7181	0.4607
Mittelwert	0.3256	0.9084	0.4268	0.3981	0.8693	0.5220	0.4041	0.8456	**0.5307**

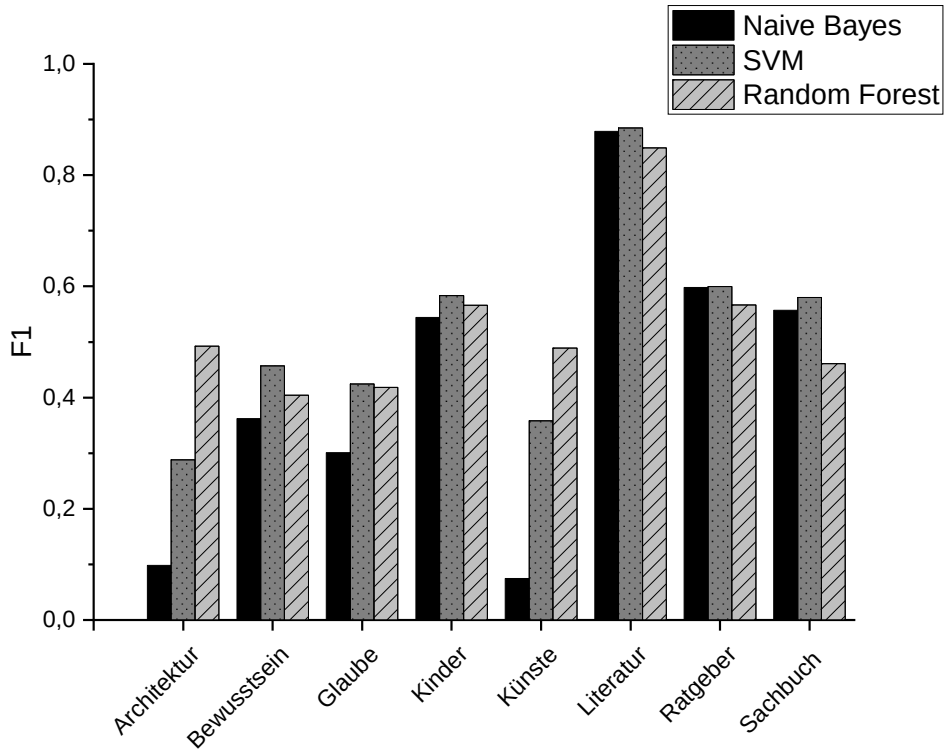

Abbildung 20: Klassifikationsergebnis der Evaluationsdaten - Bitvektor - Undersampling

Auch hier sind die Ergebnisse nicht signifikant besser als beim Bitvektor. Die F_1-Maße sind zu niedrig und somit nicht aussagekräftig. Dabei fällt auf, dass Random Forest und die SVM annähernd das gleiche Resultat erzielen, der Naive Bayes Algorithmus mit ca. 42 % das schlechteste Modell darstellt. Die Schwankungsbreite der F_1-Maße hat sich halbiert und sie schwanken nun noch im Bereich von ca. 10 %.

Die mit Abstand beste Klasse ist Literatur, die schlechtesten Architektur und Künste, die vor allem von Naive Bayes schlecht gelabelt wurden.

Bei dieser Klassifikation fällt auf, dass bei allen Modellen der recall deutlich die Precision überwiegt. Einzige Ausnahme stellt die Klasse Literatur, bei der sich beide Maßzahlen ähneln.

Oversampling der Trainingsdaten

Die letzte Klassifikation mit dem Bitvektor nutzt das Oversampling als Vorverarbeitungsschritt. In Tabelle 21 und Abbildung 21 sind die Ergebnisse des Oversamplings zusammengetragen.

Tabelle 21: Klassifikationsergebnis der Evaluationsdaten - Bitvektor - Oversampling

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.7500	0.5294	0.6207	0.5909	0.7647	0.6667	0.7368	0.8236	**0.7778**
Ganzheitliches Bewusstsein	0.4823	0.7816	**0.5965**	0.4012	0.7931	0.5328	0.3077	0.8276	0.4486
Glaube & Ethik	0.7429	0.7222	**0.7324**	0.5600	0.7778	0.6512	0.3588	0.8472	0.5041
Kinder- & Jugendbuch	0.7602	0.7057	**0.7319**	0.6103	0.7623	0.6779	0.4356	0.7396	0.5483
Künste	0.6667	0.2667	0.3810	0.6154	0.5333	**0.5714**	0.4615	0.4000	0.4286
Literatur & Unterhaltung	0.8462	0.9132	0.8784	0.8438	0.9319	**0.8856**	0.7226	0.9282	0.8126
Ratgeber	0.6560	0.8222	**0.7298**	0.5150	0.8400	0.6385	0.4485	0.8133	0.5782
Sachbuch	0.5993	0.5772	0.5889	0.5400	0.7248	**0.6189**	0.4155	0.6107	0.4946
Mittelwert	0.6880	0.6648	**0.6575**	0.5846	0.7660	0.6554	0.4859	0.7488	0.5741

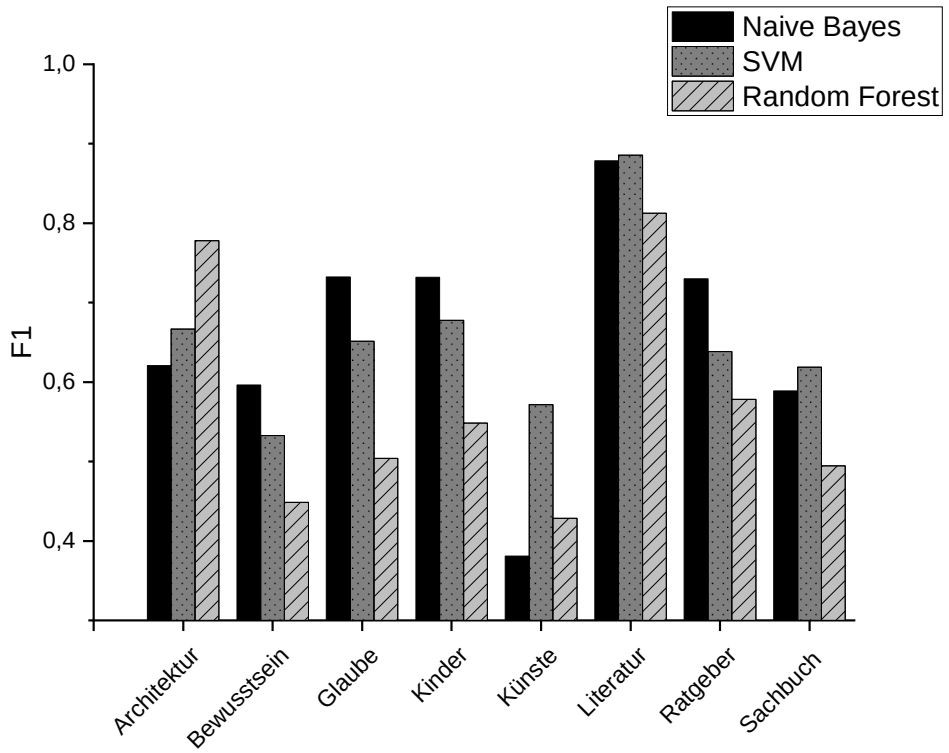

Abbildung 21: Klassifikationsergebnis der Evaluationsdaten - Bitvektor - Oversampling

Das Oversampling erzielt beim Bitvektor das beste Ergebnis. Naive Bayes und die SVM erzielen beide F_1-Maße von ca. 65 %. Random Forest liegt mit ca. 57 % weniger als zehn Prozent davon entfernt.

Hier liegen die Werte für Precision und Recall näher beieinander, Random Forest besitzt den größten Unterschied von ca. 20 %.

Die Klasse Literatur liefert erneut ein stabiles Ergebnis von über 80 % im F_1-Maß. Auffällig ist jedoch die gute Klassifikation von Architektur durch Random Forest. Die Methode des Oversamplings scheint die große Unausgeglichenheit des Datenumfangs am besten auszugleichen.

Bei der Naive Bayes Implementierung liegen Precision und Recall näher zusammen, bei SVM und Random Forest ist ein größerer Trend zum höheren Recall zu erkennen.

5.3.2 Termfrequenz

Komplette Trainingsdaten

Diese Klassifikation wurde mit dem gesamten Trainingsdaten durchgeführt, als Feature wurde die Termfrequenz gewählt. In Tabelle 22 und Abbildung 22 sind die Klassifikationsdaten zusammengefasst.

Tabelle 22: Klassifikationsergebnis der Evaluationsdaten - TF - Full

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.5000	0.2941	0.3704	1.0000	0.5882	**0.7407**	0.2500	0.1765	0.2069
Ganzheitliches Bewusstsein	0.5289	0.7356	**0.6154**	0.1413	0.5977	0.2286	0.2250	0.1034	0.1417
Glaube & Ethik	0.6866	0.6389	**0.6619**	0.1873	0.7361	0.2986	0.5000	0.2222	0.3077
Kinder- & Jugendbuch	0.7016	0.6566	**0.6784**	0.2062	0.7019	0.3188	0.4855	0.2528	0.3325
Künste	0.5000	0.2667	**0.3478**	1.0000	0.2000	0.3333	0.1111	0.0667	0.0833
Literatur & Unterhaltung	0.8508	0.8993	**0.8744**	0.7289	0.5718	0.6409	0.8133	0.7223	0.7651
Ratgeber	0.5865	0.8133	**0.6815**	0.1854	0.7556	0.2977	0.4965	0.3156	0.3859
Sachbuch	0.5385	0.6107	**0.5723**	0.2602	0.6007	0.3631	0.3667	0.1846	0.2455
Mittelwert	0.6116	0.6144	**0.6003**	0.4637	0.5940	0.4027	0.4060	0.2555	0.3086

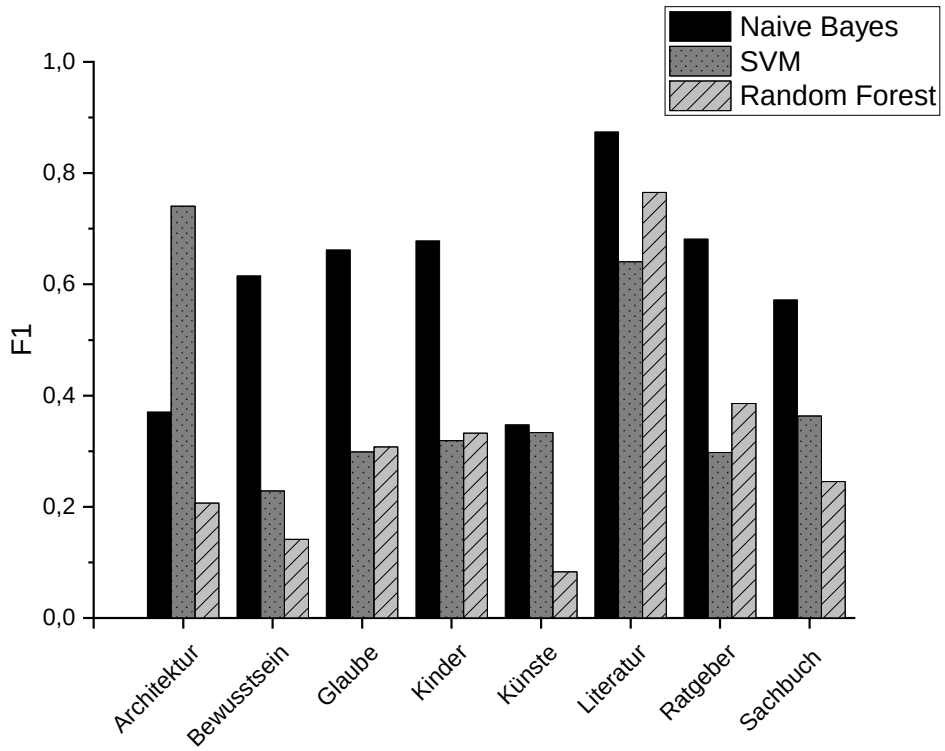

Abbildung 22: Klassifikationsergebnis der Evaluationsdaten - TF - Full

Das mit Abstand beste Modell ist Naive Bayes mit ca. 60 %. SVM und Random Forest erzielen sehr niedrig Werte für Precision und Recall, was einen ebenso niedriges F_1-Maß bedeutet. Die Resultate sind erneut zu gering für eine aussagekräftige Beurteilung.

Random Forest ist ca. 30 % schlechter als Naive Bayes mit einem F_1-Maß von 31 %.

Die SVM liefert ein sehr gutes Ergebnis für die Klasse Architektur, die Precision liegt bei eins. Die beste Klasse ist erneut Literatur.

Undersampling der Trainingsdaten

Nun folgt die Kombination aus Undersampling der Trainingsdaten und Termfrequenz als Feature. In Tabelle 23 und Abbildung 23 sind die Klassifikationsergebnisse zusammengefasst.

Tabelle 23: Klassifikationsergebnis der Evaluationsdaten - TF - Undersampling

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.0554	0.9412	0.1046	0.2143	0.8824	0.3448	0.3636	0.9412	**0.5246**
Ganzheitliches Bewusstsein	0.2198	0.9425	0.3565	0.2946	0.8161	**0.4329**	0.2577	0.8621	0.3968
Glaube & Ethik	0.1918	0.9722	0.3204	0.3140	0.9028	**0.4859**	0.2625	0.8750	0.4038
Kinder- & Jugendbuch	0.4179	0.8453	0.5593	0.4978	0.8604	**0.6307**	0.4153	0.8415	0.5561
Künste	0.0426	0.9333	0.0814	0.2564	0.6667	**0.3704**	0.2571	0.6000	0.3600
Literatur & Unterhaltung	0.8508	0.8993	0.8744	0.8420	0.9347	**0.8859**	0.7991	0.8555	0.8263
Ratgeber	0.4665	0.8978	**0.6140**	0.4554	0.8844	0.6012	0.4283	0.8756	0.5752
Sachbuch	0.4148	0.7919	0.5444	0.4585	0.8154	**0.5870**	0.3746	0.8322	0.5167
Mittelwert	0.3325	0.9029	0.4319	0.4166	0.8454	**0.5424**	0.3948	0.8354	0.5199

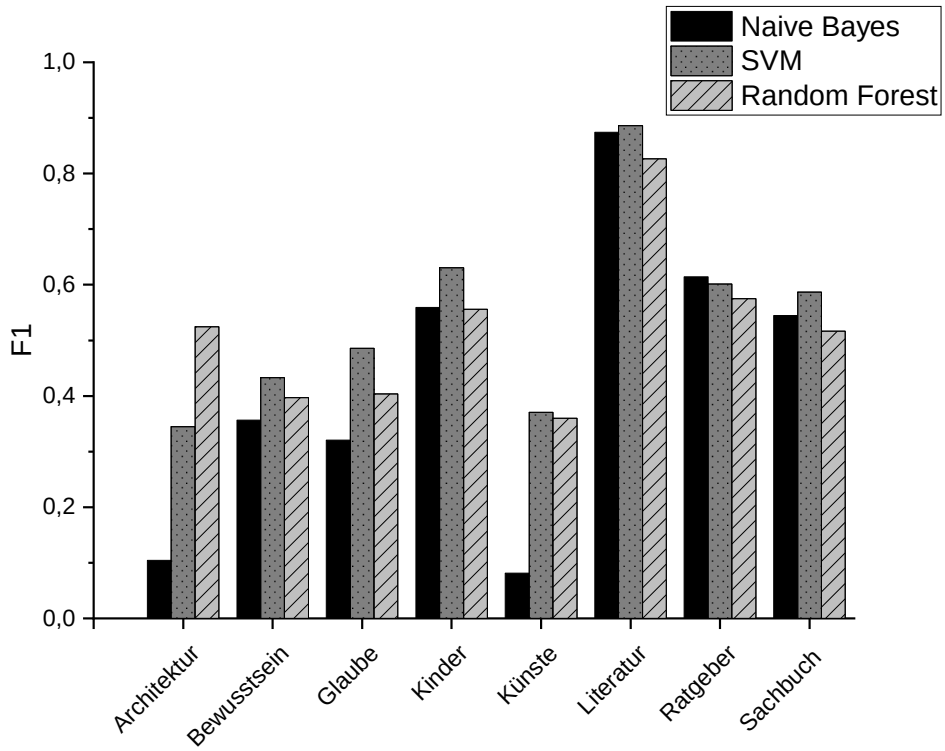

Abbildung 23: Klassifikationsergebnis der Evaluationsdaten - TF - Undersampling

Das gemittelte F_1-Maß ist erneut bei allen Modellen niedrig, erneut ist beim Undersampling Naive Bayes das schlechteste Modell. Random Forest und die SVM liegen zwischen 50 % und 55 %.

Bei allen Modellen ist der Recall deutlich höher als die Precision. Auffällig sind die sehr niedrigen Werte der Precision von Architektur und Künste bei Naive Bayes. Die beste Klasse ist Literatur.

Oversampling der Trainingsdaten

Zum Abschluss der Termfrequenz folgt nun die Klassifikation mit dem Oversampling der Trainingsdaten. In Tabelle 24 und Abbildung 24 sind die Ergebnisse zusammengefasst.

Tabelle 24: Klassifikationsergebnis der Evaluationsdaten - TF - Oversampling

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.7273	0.4706	0.5714	0.5238	0.6471	0.5789	0.7500	0.8824	**0.8108**
Ganzheitliches Bewusstsein	0.4929	0.7931	**0.6079**	0.4012	0.7701	0.5276	0.3153	0.8046	0.4531
Glaube & Ethik	0.7397	0.7500	**0.7448**	0.5816	0.7917	0.6706	0.3974	0.8333	0.5381
Kinder- & Jugendbuch	0.7750	0.7019	**0.7366**	0.6390	0.7547	0.6920	0.4493	0.7358	0.5579
Künste	0.6667	0.2667	0.3810	0.3750	0.4000	0.3871	0.5385	0.4667	**0.5000**
Literatur & Unterhaltung	0.8516	0.8993	0.8748	0.8416	0.9366	**0.8865**	0.7305	0.9282	0.8176
Ratgeber	0.6618	0.8089	**0.7280**	0.5132	0.8622	0.6434	0.4637	0.7956	0.5859
Sachbuch	0.5813	0.5638	0.5724	0.5319	0.7282	**0.6147**	0.4167	0.6208	0.4987
Mittelwert	0.6870	0.6568	**0.6521**	0.5509	0.7363	0.6251	0.5077	0.7584	0.5953

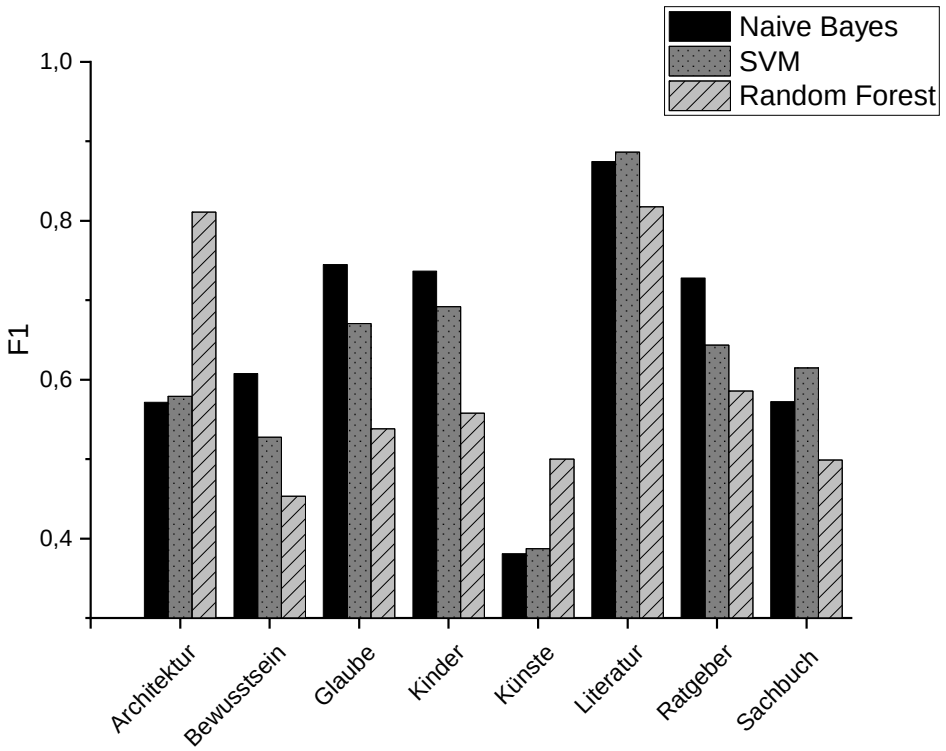

Abbildung 24: Klassifikationsergebnis der Evaluationsdaten - TF - Oversampling

Beim Oversampling treten erneut alle drei Modelle näher zusammen auf, die F_1-Werte liegen zwischen 60 und 65 %. Damit übertrifft das Oversampling erneut die bisherigen Balancierungen. Das beste Modell ist Naive Bayes.

Außer bei Naive Bayes ist ein Trend zu höheren Recall-Werten erkennbar.

Random Forest liefert bei den Klassen Architektur und Künste erneut das mit Abstand beste Ergebnis, über alle Klassen ist es dennoch das schlechteste Modell.

Die beste Klasse ist weiterhin Literatur.

5.3.3 Termfrequenz - Inverse Dokumentenfrequenz

Komplette Trainingsdaten

Zum Abschluss der Klassifikation der Evaluationsdaten folgt TF-IDF als Feature. Die erste Berechnung wurde mit dem gesamten Traingsdatenumfang ausgeführt. In Tabelle 25 und Abbildung 25 sind die Ergebnisse zusammengefasst.

Tabelle 25: Klassifikationsergebnis der Evaluationsdaten - TF-IDF - Full

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.0000	0.0000	0.0000	0.2462	0.9412	**0.3902**	0.0000	0.0000	0.0000
Ganzheitliches Bewusstsein	1.0000	0.0460	0.0879	0.7105	0.3103	**0.4320**	0.5714	0.1379	0.2222
Glaube & Ethik	1.0000	0.1944	0.3256	0.0514	1.0000	0.0978	0.7241	0.2917	**0.4158**
Kinder- & Jugendbuch	0.9831	0.2189	0.3580	0.5000	0.4830	**0.4914**	0.8272	0.2528	0.3873
Künste	0.0000	0.0000	0.0000	0.0828	0.8667	**0.1512**	0.0000	0.0000	0.0000
Literatur & Unterhaltung	0.8402	0.9366	**0.8858**	0.8181	0.6502	0.7245	0.7929	0.8498	0.8204
Ratgeber	0.8725	0.3956	0.5443	0.4985	0.7156	**0.5876**	0.6814	0.3422	0.4556
Sachbuch	0.8438	0.1812	**0.2983**	0.1688	0.9765	0.2878	0.6282	0.1644	0.2606
Mittelwert	0.6925	0.2466	0.3125	0.3845	0.7429	**0.3953**	0.5282	0.2549	0.3202

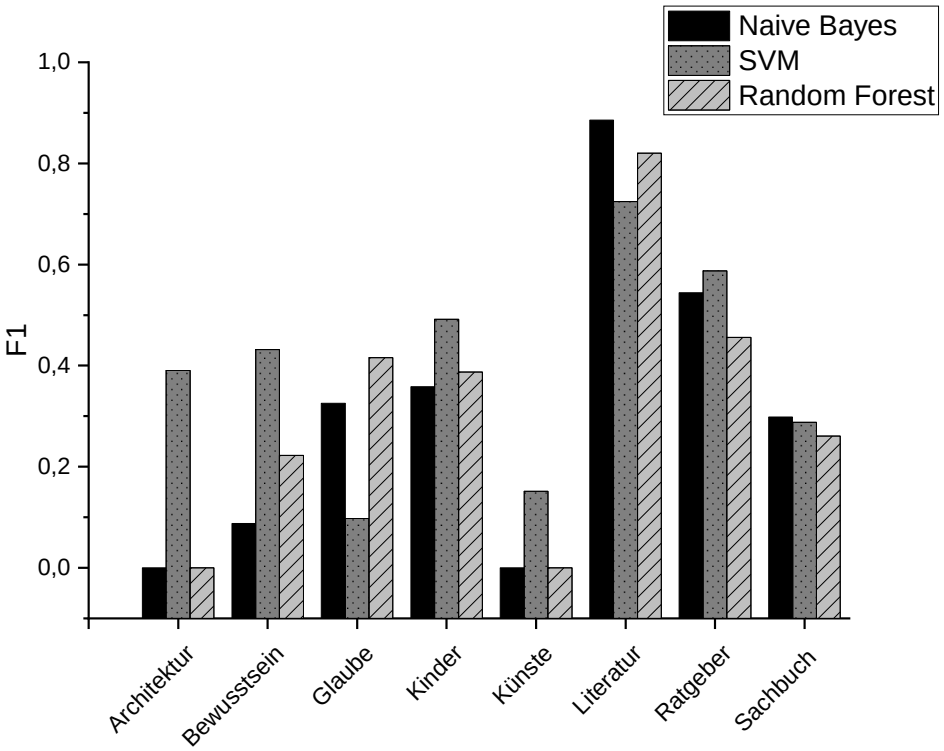

Abbildung 25: Klassifikationsergebnis der Evaluationsdaten - TF-IDF - Full

Mit weniger als 40 % im F_1-Maß stellt diese Klassifikation die bislang schlechteste dar, das höchste Maß erreicht die SVM. Die Ergebnisse weichen nur geringfügig voneinander ab.

Besonders auffällig sind die Null-Werte für Architektur und Künste bei Naive Bayes und Random Forest. Werden die Null-Werte ignoriert, dann erzielt der Naive Bayes Algorithmus eine sehr hohe Precision von über 90 %.

Bei der SVM liegt der Trend hin zu einem höheren Recall, Random Forest dagegen liefert eine höhere Precision.

Erneut wurde die Klasse Literatur mit Abstand am besten klassifiziert.

Undersampling der Trainingsdaten

Der nächste Schritt ist das Undersampling der Trainingsdaten in Kombination mit TF-IDF als Feature. In Tabelle 26 und Abbildung 26 sind die Ergebnisse zusammengetragen.

Tabelle 26: Klassifikationsergebnis der Evaluationsdaten - TF-IDF - Undersampling

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.0431	1.0000	0.0827	0.2500	0.9412	**0.3951**	0.1379	0.7059	0.2308
Ganzheitliches Bewusstsein	0.1923	0.9770	0.3213	0.2470	0.9540	0.3924	0.2664	0.8391	**0.4044**
Glaube & Ethik	0.1592	0.9861	0.2741	0.3801	0.9028	**0.5350**	0.2877	0.8472	0.4296
Kinder- & Jugendbuch	0.4051	0.9019	0.5591	0.5706	0.7774	**0.6581**	0.3262	0.8075	0.4647
Künste	0.0369	1.0000	0.0711	0.3889	0.4667	**0.4242**	0.1538	0.6667	0.2500
Literatur & Unterhaltung	0.8402	0.9366	**0.8858**	0.7080	0.9907	0.8258	0.7886	0.8144	0.8013
Ratgeber	0.4315	0.9378	0.5910	0.6709	0.6978	**0.6841**	0.4066	0.8222	0.5441
Sachbuch	0.4286	0.9060	**0.5819**	0.2539	0.9900	0.4041	0.3504	0.7819	0.4839
Mittelwert	0.3171	0.9557	0.4209	0.4337	0.8525	**0.5399**	0.2899	0.7621	0.3871

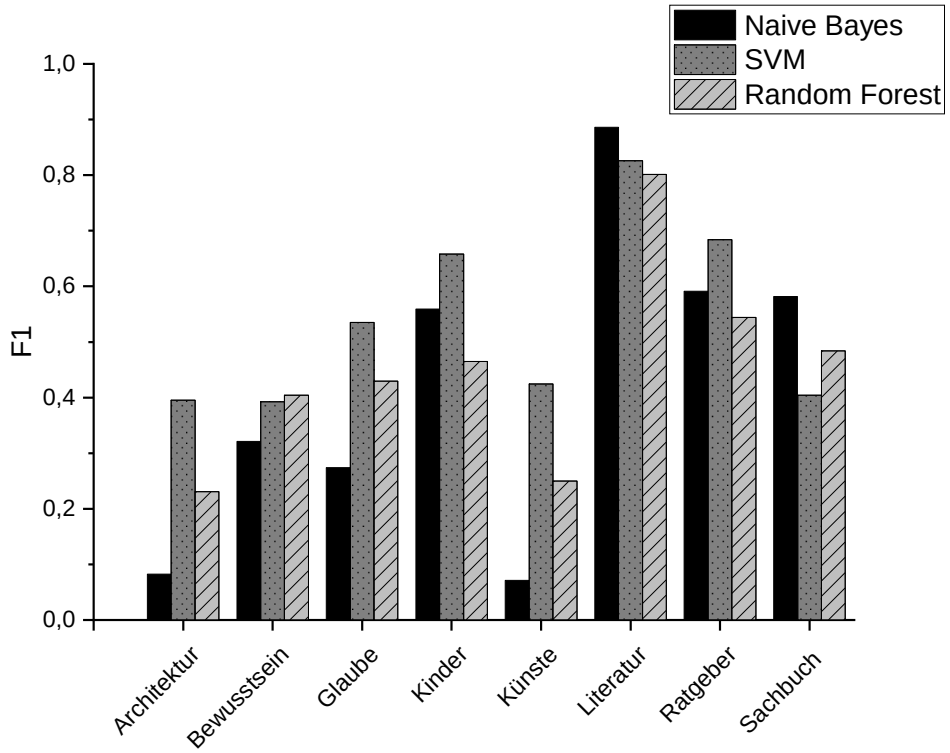

Abbildung 26: Klassifikationsergebnis der Evaluationsdaten - TF-IDF - Undersampling

Anhand diesen Ergebnissen fällt auf, dass das Undersampling die Klassifikation mit dem TF-IDF-Feature verbessert. Die SVM als besten Modell erzielt ein F_1-Maß von ca. 54 %. Random Forest und Naive Bayes liegen sehr nahe beieinander (ca. 40 % im F_1-Maß).

Bei allen drei Modellen ist der Recall deutlich höher als die Precision. Die beste Klasse ist weiterhin Literatur.

Oversampling der Trainingsdaten

Die letzte Klassifikation kombiniert das Oversampling mit dem TF-IDF Feature. In Tabelle 27 und Abbildung 27 sind die Ergebnisse zusammengefasst.

Tabelle 27: Klassifikationsergebnis der Evaluationsdaten - TF-IDF - Oversampling

	fast	Naive	Bayes	SVM			RF		
	P	R	F_1	P	R	F_1	P	R	F_1
Architektur & Garten	0.6667	0.7059	0.6857	0.6522	0.8824	**0.7500**	0.6667	0.8235	0.7368
Ganzheitliches Bewusstsein	0.4247	0.9080	**0.5788**	0.4239	0.8966	0.5756	0.3077	0.8276	0.4486
Glaube & Ethik	0.5463	0.8194	**0.6556**	0.4224	0.9444	0.5837	0.4161	0.7917	0.5454
Kinder- & Jugendbuch	0.7156	0.7584	**0.7417**	0.5459	0.8755	0.6725	0.4403	0.7509	0.5551
Künste	0.5714	0.5333	**0.5517**	0.3889	0.4667	0.4242	0.4375	0.4667	0.4516
Literatur & Unterhaltung	0.8409	0.9366	**0.8861**	0.6953	0.9897	0.8168	0.7680	0.8862	0.8229
Ratgeber	0.6069	0.8578	**0.7109**	0.5216	0.9111	0.6634	0.4836	0.7867	0.5990
Sachbuch	0.6066	0.7349	**0.6646**	0.4115	0.8892	0.5626	0.3606	0.6812	0.4715
Mittelwert	0.6224	0.7818	**0.6844**	0.5077	0.8570	0.6311	0.4851	0.7518	0.5789

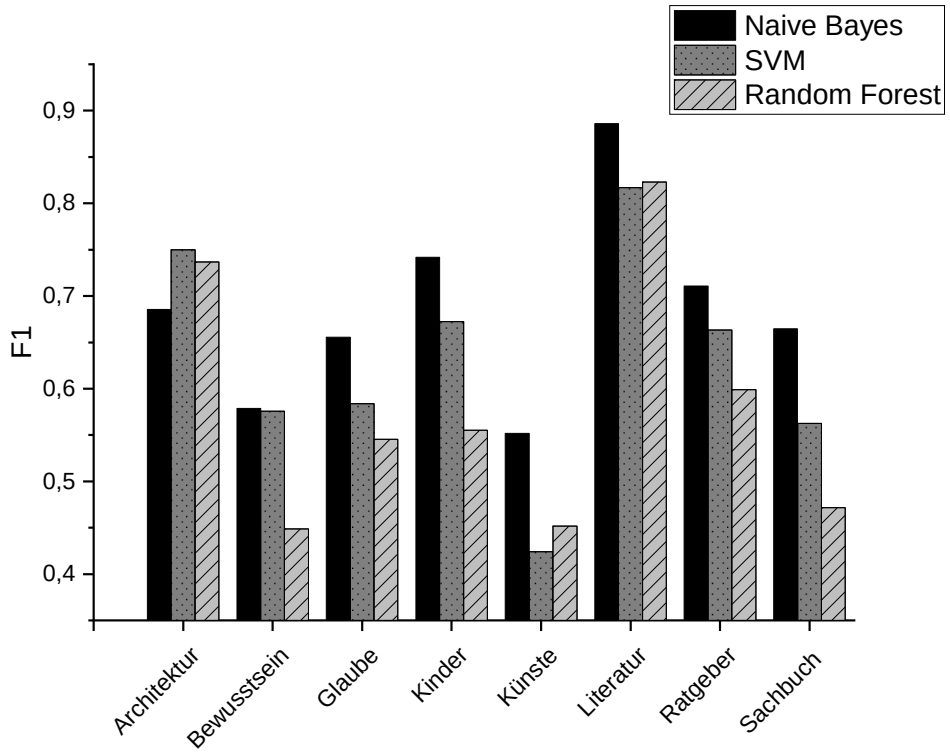

Abbildung 27: Klassifikationsergebnis der Evaluationsdaten - TF-IDF - Oversampling

Die gemittelten F_1-Maße liegen in einem Intervall von ca. 10 %, wobei Naive Bayes mit knapp 70 % das beste Modell ist. Damit erreicht diese Klassifikation ein höheres Ergebnis als die vorangegangenen mit TF-IDF als Feature.

Auffällig sind die sehr guten Berechnungen der Klasse Architektur mit ca. 70 % im F_1-Maß. Künste wird weiterhin schlechter klassifiziert. Die beste Klasse ist auch bei dieser Modellierung Literatur.

Bei allen Modellen ist der Recall höher als die Precision.

5.3.4 Zusammenfassung

In Abbildung 28 sind die gemittelten F_1-Maße pro Feature und Modell graphisch zusammengefasst. Dabei wurde eine andere Anordnung als bei den Kapiteln gewählt: der markanteste Trend liegt nicht im Verlauf der Feature, sondern in der Wahl der Vorverarbeitung.

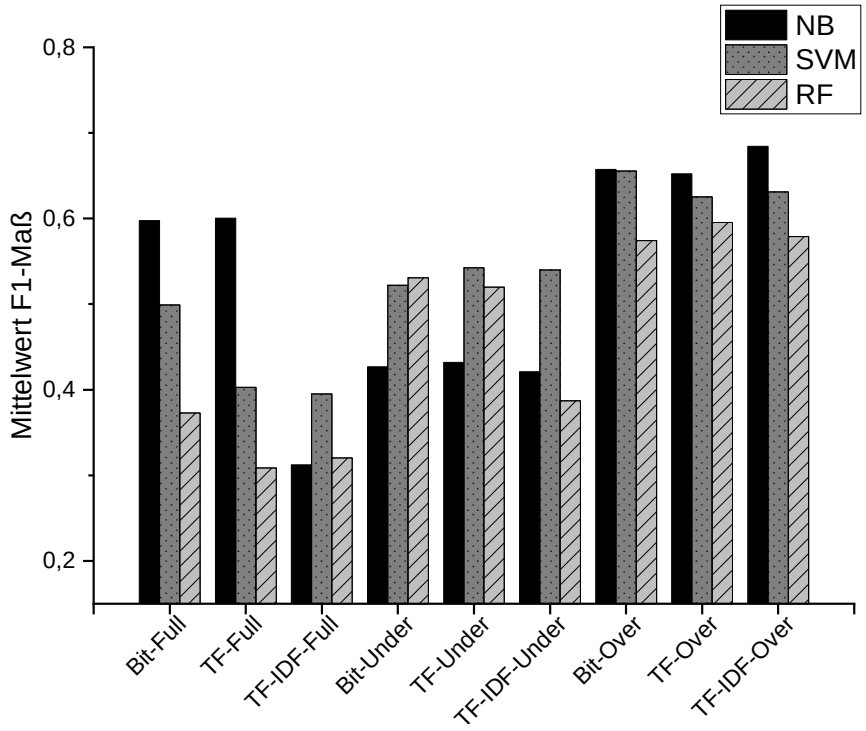

Abbildung 28: Zusammenfassung der mittleren F_1-Maße pro Feature und Modell

Auch wenn Naive Bayes eine Ausnahme in den ersten beiden Klassifikationen darstellt (kompletter Trainingsdatensatz - Bit-Full & TF-Full), liefert ein unbalancierter Datensatz mit allen drei Features das schlechteste Ergebnis. Außer bei Naive Bayes verbessert das Undersampling die Klassifikation merklich, vor allem die SVM liefert konstant bessere Resultate.

Das mit Abstand beste Ergebnis erzielt das Oversampling. Dabei erreichen alle Modelle ihr Maximum im F_1-Maß.

Die Klasse Literatur lieferte stets das beste Klassifikationsergebnis, was an dem großen Datensatzumfang liegt. Die Sample-Methoden haben kaum Einfluss auf diese Klasse, da sie die Hälfte des gesamten Datensatzes ausmacht und somit in binärer Sicht als einzige Klasse

balanciert ist.

Naive Bayes erzielt bei den Klassifikationen ohne Sample-Methoden die mit Abstand besten Ergebnisse. Als probabilistischer Klassifikator ist er vermutlich resistenter bei unausgeglichenen Datensätzen. Beim Verwenden des TF-IDF-Feature halbiert sich das Ergebnis.

Die SVM liefert jeweils pro Sample-Methode annähernd konstante Ergebnisse. Das bedeutet, dass der größere Einfluss auch hier auf dem Sampling statt auf dem Feature liegt. Das gleiche Verhalten ist bei Random Forest zu beobachten.

6 Zusammenfassung

6.1 Vergleich der Daten

Beide Datensätze erzielen im Vergleich sehr verschiedene Resultate, die Trainingsdaten wurden besser klassifiziert als die Evaluationsdaten.

Da bei den Trainingsdaten der Fokus nur auf die Auswahl der Feature gelegt wurde, wurde deren Einfluss auf die Güte der Klassifikation untersucht. Das beste Ergebnis erzielt Naive Bayes in Verbindung mit den einfachsten Feature wie Bitvektor oder Termfrequenz. Die SVM lieferte das beste F_1-Maß in der Kombination von Bigrams und Termfrequenz, auch wenn dieses nur sehr geringfügig besser ist. Random Forest erzielte sehr stabile Resultate, die annähernd unabhängig von den gewählten Feature sind. Die Bigrams eignen sich weniger gut beim alleinigen Einsatz in der Textklassifikation. [21]

Zusammenfassend sind die grundlegenden Feature die besten, da ebenso der Ressourcenaufwand mit berücksichtigt werden muss. Durch die Kombination mehrerer Feature in einer Dokument-Term-Matrix steigt die Speicheranforderung enorm und die Laufzeit verlängert sich. Somit sind die Trainingsdaten am effektivsten mit einem einfachen Feature und der Naive Bayes Implementierung klassifiziert.

Bei der Klassifikation der Evaluationsdaten fällt das F_1-Maß auf ein sehr niedriges Level. Hier zeigt sich, dass das Optimieren der Feature nicht ausreicht, und der Fokus auf die Verbesserung des Sampling-Methoden gelegt werden muss.

Sollte die hier entwickelte Implementierung Anwendung finden, ist das Oversampling die beste Methode. Der Naive Bayes liefert mit dem TF-IDF-Feature das beste Ergebnis, die SVM in Verbindung mit dem Bitvektor und Random Forest mit der Termfrequenz.

Der große Unterschied in den F_1-Maßen spricht für eine Überanpassung der Modelle. Wenn nur die Trainingsdaten klassifiziert werden, liefern die Modelle sehr gute Ergebnisse. Die Evaluationsdaten, also ungesehene Daten, verursachen ein schlechtes Ergebnis. Um dieses Problem zu vermeiden, sind im nächsten Kapitel einige Ansätze zusammengetragen, die eine mögliche Lösung bieten.

Abschließend lässt sich noch sagen, dass alle hier angewandten Modelle, Feature und Vorverarbeitungsschritte sehr abstrakter und genereller Natur sind. Eine Anpassung und Annäherung an das Problem erlaubt somit eine Fokussierung auf dasselbe. Mögliche Feature, die aus der Struktur und dem Inhalt der Daten abgeleitet werden können, sind ebenso im nächsten Kapitel vorgestellt.

Die hier entstandenen Implementierungen sind möglicherweise für andere Klassifikationsprobleme anwendbar. Wie in der Einleitung erwähnt, ist ein breites Spektrum der Anwendung von Textklassifikation gegeben und die Ausrichtung auf ein Problem ist beispielsweise durch problemspezifische Feature möglich. Bezug nehmend auf das anfängliche, forensische Beispiel, handelt es sich um bestimmte Informationen aus dem Mailverkehr einer Firma, wie Namen, Daten oder Geldbeträge, die direkt in die Klassifikation einfließen können.

6.2 Ausblick

Diese Arbeit hat den Fokus auf die Kombination verschiedener Feature und Vorverarbeitungsschritte gelegt. Dabei wurden Standardmethoden angewandt und deren Auswirkungen

miteinander verglichen.

Um die Resultate weiter zu verbessern, können an vielen Stellen andere Herangehensweisen erprobt werden. Eine erste Möglichkeit sind andere, spezifischere Feature zu nutzen. So können Autoreninformationen in die Klassifikation einfließen. Diese können einerseits zur Erweiterung der Dokument-Term-Matrix eingesetzt werden, andererseits zur Gewichtung derselben benutzt werden. Dabei wird auf das bisherige Wissen über die Autoren zurückgegriffen, indem bekannte Autoren ihren Klassen zugeordnet werden und somit eine Vermutung über die Klasse der zu klassifizierenden Bücher angestellt wird.

Eine anderer Problemlösungsansatz ist die Wissenserweiterung über externe Quellen. Der hier vorliegende Datensatz aus Trainings- und Evaluationsdaten hat seine finale Größe erreicht und somit kommen keine weiteren Bücher hinzu, die das Trainieren verbessern können. Dennoch kann das Wissen über die jeweiligen Klassen und Terme vergrößert werden, indem der Fokus auf andere Wissensquellen ausgerichtet wird. So können Wikipediaeinträge, passend zu den Oberkategorien, als eine Möglichkeit genutzt werden. Werden daraus Listen mit den häufigsten Termen erstellt, kann das die Klassifikation verbessern, da somit spezifischere Terme gefunden werden. Dadurch entstehen eine Vielzahl von neuen Feature, die jeweils eine binäre Entscheidung treffen, abhängig von auftretenden Termen: religiöse Begriffe für die Kategorie Glaube & Ethik, fiktive Namen oder Fabelwesen für Literatur oder Kinderbücher oder das Detektieren von lateinischen Prä- und Suffixen für Sachbücher.

Das Wissen über die verwendeten Terme kann mit Hilfe von Synsets verbessert werden. Dabei werden nicht nur die explizit in den Büchern verwendeten Wörter zur Klassifikation eingesetzt, sondern auch deren Synonyme. Eine mögliche Bibliothek kann OpenThesaurus darstellen.

Neben der Anpassung der Feature, kann ebenso die Vorverarbeitung optimiert werden. Bei der Klassifikation der Trainingsdaten wird ausschließlich ein Undersampling zur Balancierung angewandt, wobei auch das Oversampling aufschlussreich wäre. Weiterhin können komplett andere Sample-Methoden erprobt werden. Es existieren durchaus komplexere Methoden, um unausgeglichene Datensätze trotzdem korrekt zu klassifizieren.

Damit die vorgestellten Klassifikationen komplett automatisiert ein Ergebnis der Klasse pro Buch liefern, muss die Binarisierung erweitert werden. Beim Einsatz einer Multi-Class-Prediction liefert der Algorithmus direkt die beste Klasse pro Buch. In der hier vorliegenden Implementierung wird jedes Buch pro Kategorie binär klassifiziert. Dieses Vorgehen lässt die Möglichkeit offen, dass ein Buch zu mehreren Klassen zugeordnet wird. Die finale Klasse wird anhand der größten Wahrscheinlichkeit bestimmt.

Der letzte Punkt bezieht sich auf die initiale Aufgabenstellung der Universität Hamburg. Die gesamte Arbeit beleuchtet nur die erste Ebene der Hierarchie, eine flache Klassifikation der Bücher. Um den zweiten Abschnitt der Aufgabe zu bearbeiten, muss die hierarchische Multi-Label-Klassifiktion mit betrachtet werden. Somit muss die Klassifikation je Ebene optimiert werden, um die Komplexität der Daten zu erfassen und sie somit besser hierarchisch klassifizieren zu können.

Literatur

[1] C. Zhai and S. Massung. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. Morgan & Claypool Publishers, 2016.

[2] Charu C. Aggarwal and ChengXiang Zhai, editors. *Mining Text Data*, chapter 6. Springer, New York, 2012.

[3] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30:25–36, 11 2005.

[4] Aixin Sun, Ee-Peng Lim, and Wee-Keong Ng. Performance measurement framework for hierarchical text classification. *Journal of the American Society for Information Science and Technology*, 54(11):1014–1028, 2003.

[5] Carlos N. Silla and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1), Jan 2011.

[6] Shou Xu. Bayesian naïve bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59, 2018.

[7] Jan-Nikolas Sulzmann. Pairwise naive bayes classifier. 2011.

[8] Harry Zhang and Jiang Su. Naive bayes for optimal ranking. *Journal of Experimental & Theoretical Artificial Intelligence*, 20(2):79–93, 2008.

[9] Jun Ting Chen, Jian Zhong, Yi Cai Xie, and Cai Yun Cai. Text classification using svm with exponential kernel. *Applied Mechanics and Materials*, 519-520:807–810, 02 2014.

[10] Thorsten Joachims. Estimating the generalization performance of an svm efficiently. 01 2000.

[11] Jürgen Cleve and Uwe Lämmel. *Data mining*. de Gruyter, Oldenbourg, 2014.

[12] Sameen Maruf, Kashif Javed, and Haroon A. Babri. Improving text classification performance with random forests-based feature selection. *Arabian Journal for Science and Engineering*, 41(951–964), 03 2016.

[13] Andreas Wierse and Till. Riedel. *Smart Data Analytics : Mit Hilfe von Big Data Zusammenhänge erkennen und Potentiale nutzen*. De Gruyter Oldenbourg, 2017.

[14] Xiao Juan Chen, Zhi Gang Zhang, and Yue Tong. An improved id3 decision tree algorithm. *Advanced Materials Research*, 2014.

[15] Charu C. Aggarwal. *Machine Learning for Text*. Springer International Publishing, 2018.

[16] Jasmeet Singh and Vishal Gupta. Text stemming : Approaches, applications, and challenges approaches, applications, and challenges. 2016-09-16.

[17] Haibin Liu, Tom Christiansen, William A Baumgartner, and Karin Verspoor. Biolemmatizer: a lemmatization tool for morphological processing of biomedical text. 01 2012.

[18] Maksim Lapin, Matthias Hein, and Bernt Schiele. Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification. 2018-07-01.

[19] Ayon Sen, Md Monirul Islam, Kazuyuki Murase, and Xin Yao. Binarization with boosting and oversampling for multiclass classification. 2016.

[20] Marcelo Beckmann, Nelson F. F. Ebecken, and Beatriz S. L. Pires de Lima. A knn undersampling approach for data balancing. 2015.

[21] Sida Wang and Christopher D. Manning. Baselines ans bigrams: Simple, good sentiment ans topic classification. *Proceeding of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 90–94, 07 2012.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.
Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, den 13. August 2019