Automatisierte Klassifizierung von Textinhalten

Muhammet Altindal

2014-01-20

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Kapitel 1

Einleitung

1.1 Problematisierung der Thematik

Die Text Klassifizierung bezeichnet die Aufgabe der Zuordnung von Textinhalten in vordefinierte Kategorien. Das Thema führt bis zu Maron (1961) zurück. Maron (1961) verwendet einen probabilistischen Ansatz und versucht neue Dokumente aufgrund bestimmter vorkommender Wörter zu klassifizieren. Die automatisierte Text Klassifizierung kann in verschiedensten Problemstellungen Anwendung finden. Zudem kommt die enorme Menge an Daten hinzu, sodass ein manuelles Zuordnen von Dokumenten zu Kategorien zeitintensiv sein kann. Die enorme Datenmenge macht die Anwendung maschineller Lernalgorithmen attraktiv. Durch die Anwendung maschineller Lernalgorithmen können Klassifikatoren automatisiert generiert werden.

Die folgende Auflistung einer Auswahl von Themengebieten zeigt mögliche Anwendungsfälle einer automatisierten Text Klassifizierung.[1]

- Identifikation von Autoren (Forsyth, 1999)
- Identifikation einer Sprache (Cavnar/Trenkle, 1994)
- Indexierung von Dokumenten (Fuhr/Knorz, 1984; Tzeras/Hartmann, 1993)

[1] Vgl. Sebastiani, Fabrizio: Machine Learning in Automated Text Categorization. ACM Comput. Surv. 34 März 2002, Nr. 1 ⟨URL: http://dx.doi.org/10.1145/505282.505283⟩, ISSN 0360–0300, S. 5.

- Organisierung von Dokumenten (Kessler/Numberg/Schütze, 1997)
- Filtern von Inhalten (Sriram et al., 2010)

1.2 Zielsetzung

Die Zielsetzung der Arbeit ist die Darstellung der Möglichkeiten und Grenzen von Methoden für die automatisierte Klassifizierung von Textinhalten. Der Fokus liegt dabei auf Methoden des maschinellen Lernens. Statt der vollständigen Darstellung aller Lernalgorithmen soll eine Untermenge der Lernalgorithmen beschrieben werden. Der Schwerpunkt der Darstellung einiger Lernalgorithmen liegt in der Intuition sowie der Funktionsweise der Lernalgorithmen. Die Beschreibung der Methoden bezieht sich auf die Anwendung auf Textinhalte im technischen Sinne, d.h. es wird angenommen, dass die Daten bereits in Form von Textdateien zur Verfügung stehen. Die Extraktion von Texten aus Audio- oder Bilddateien ist nicht Gegenstand dieser Arbeit. Zudem werden überwachte Lernszenarien betrachtet, d.h. es wird angenommen, dass Text Exemplare zusammen mit der Zuordnung zu den Klassen vorhanden sind.

Sebastiani (2002) unterscheidet zwischen der eindeutigen Zuweisung einer Klasse und der Zuweisung von mehreren Klassen zu einem Textdokument. Einige Lernalgorithmen, wie zum Beispiel probabilistische Verfahren, liefern Wahrscheinlichkeiten für die Zugehörigkeit der Klassen zu einem Textdokument. Dies würde die Zuweisung von mehreren wahrscheinlichen Klassen zu einem Dokument erlauben. Jedoch bieten nicht alle Lernalgorithmen diese Möglichkeit an. Dieses Details wird im Rahmen dieser Arbeit nicht weiter berücksichtigt.

Die Zielgruppe dieser Arbeit stellen Software Ingenieure dar. Für Software Ingenieure soll diese Arbeit eine Übersicht über die Methoden entlang des idealtypischen Aufbaus einer automatisierten Klassifizierung geben. Die Vorbereitung, Selektion sowie Gewichtung von Merkmalen bzw. Wörtern gehören zu den essentiellen Methoden der Text Klassifizierung. Die Vorverarbeitung der Merkmale kann einen unmittelbaren Einfluss auf die Performanz der Klassifikatoren haben. Das Ziel ist daher die strukturierte und systematische Aufbereitung anwendbarer und praktischer Ideen.

1.3 Aufbau der Arbeit

Im Grundlagen Kapitel werden Terminologien und Formalitäten hinsichtlich der Themengebiete des maschinellen Lernens und der Computerlinguistik (Natural Language Processing) definiert. Die Grundlagen zum maschinellen Lernen umfassen Hinweise zur Multi-Klassen Klassifizierung, zur Evaluation der Performanz und zum Vektorraummodell. Das Vektorraummodell bildet für einige Lernverfahren die grundlegende Struktur, in der Textinhalte erwartet werden. Die Tokenisierung, Normalisierung und automatisierte Spracherkennung enthalten relevante Methoden in der Regel zur Vorverarbeitung von Textinhalten und werden im Grundlagen Kapitel Natural Language Processing beschrieben.

Das Hauptkapitel Text Klassifizierung konzentriert sich auf drei wesentliche Schritte. Diese sind zum einen die Verarbeitung von Merkmalen, die Lernalgorithmen zur automatisierten Generierung von Modellen bzw. Klassifikatoren und die Evaluation der Klassifikatoren. Die Verarbeitung der Merkmale wird in die drei Themengebiete Selektion, Gewichtung und Transformation strukturiert. Das Kapitel Algorithmen zur Klassifizierung enthält Erläuterungen zu den Lernalgorithmen Entscheidungsbaum, Nächste-Nachbarn-Klassifikator, Naive Bayes und Support Vector Machines. Anschließend wird die Performanz einer Auswahl an Lernalgorithmen anhand des *Twenty Newsgroups* Korpus evaluiert. Abschließend werden die dargestellten Methoden mit Bezug auf die Merkmalsverarbeitung und den Lernalgorithmen beurteilt.

Kapitel 2

Grundlagen

2.1 Maschinelles Lernen

2.1.1 Begriffe

Die folgende Definition des maschinellen Lernens lehnt sich an Mitchell (1997) an. Demnach soll ein Computer Programm aus einer Erfahrung (experience) E unter Berücksichtigung einer Aufgabe (task) T und Performanzmaß (performance) P lernen, wenn es die Performanz P hinsichtlich der Aufgabe T mit der Erfahrung E verbessern kann.[1]

Die Text Klassifizierung kann als ein maschinelles Lernproblem betrachtet werden. In diesem Fall entspricht die *Aufabe* T der Klassifizierung eines Textinhalts, das *Performanzmaß* P der korrekt klassifizierten Textinhalte und die *Erfahrung* E der Trainingsmenge. Eine Trainingsmenge enthält die Textdokumente mit den zugeordneten Klassen.[2]

Im folgenden werden weitere Begriffe des maschinellen Lernens anhand der Text Klassifizierung in Anlehnung an Mohri/Rostamizadeh/Talwalkar (2012) erläutert. Mit der Text Klassifizierung wird das Lernproblem, neue Textinhalte zu einer Klasse zuzuordnen, bezeichnet.[3]

[1] Vgl. Mitchell, Tom M.: Machine Learning. 1. Auflage. McGraw-Hill Science/Engineering/Math, 3 1997, 432 Seiten, ISBN 9780070428072, S. 2.

[2] Vgl. a. a. O., S. 3.

[3] Vgl. Mohri, Mehryar/Rostamizadeh, Afshin/Talwalkar, Ameet: Foundations of Machine Learning (Adaptive Computation and Machine Learning series). The MIT Press, 8 2012, 432 Seiten, ISBN 9780262018258, S. 4 f.

Exemplar: Ein Exemplar bezeichnet dabei ein Textdokument, welches entweder für das Lernen eines Modells oder die Evaluierung eines Modells verwendet werden kann. Der Exemplarraum X beinhaltet alle Exemplare.

Merkmal: Aus einem Exemplar können Merkmale extrahiert werden. Im Rahmen der Text Klassifizierung repräsentieren Wörter oder Worthäufigkeiten idealtypische Merkmale. Die Merkmale eines Exmeplars werden in der Regel in Form eines Vektors dargestellt.

Klasse: Die Klassen $C = \{c_1, ..., c_{|C|}\}$ entsprechen den vordefinierten Kategorien, denen Textdokumente zugeordnet werden.

Trainingsmenge: Die Trainingsmenge D enthält die bereits klassifizierten Exemplare bzw. Textdokumente $\langle d, c\rangle \in X \times C$, welche für das Lernen eines Modells verwendet wird. Im Kontext der Klassifizierung von natürlichsprachlichen Textinhalten kann die Trainingsmenge auch als Korpus bezeichnet werden.

Validierungsmenge: Die Validierungsmenge wird für die Entwicklung der Modelle verwendet. Durch die Validierung können möglicherweise die Parameter von Lernalgorithmen angepasst werden, sodass für die gegebene Trainingsmenge ein besseres Modell trainiert werden kann.

Testmenge: Die Testmenge beinhaltet die Exemplare zur Evaluierung der Performanz eines erlernten Modells.

Hypothesenmenge: Die Hypothesenmenge definiert die Menge der Funktionen zur Abbildung von Exemplaren bzw. Merkmalsvektoren auf Klassen.

2.1.2 Multi-Klassen Klassifizierung

Einige Lernalgorithmen unterstützen ausschließlich eine binäre Klassifizierung. Bei einer binären Klassifizierung sind zwei Klassen $C = \{-1, 1\}$ vorhanden. Dabei wird die Zugehörigkeit eines Testdokuments zu eines dieser beiden Klassen überprüft. Es kann sich dabei beispielsweise um die Klassifizierung eines E-Mails als *Spam* oder *Nicht Spam* handeln. Im Rahmen der Text Klassifizierung kann es vorkommen, dass Textdokumente unter Berücksichtigung von mehr als zwei Klassen klassifiziert werden sollen.[4]

[4] Vgl. Sebastiani: ACM Comput. Surv. 34 [2002], S. 4.

Mohri/Rostamizadeh/Talwalkar (2012) strukturiert die Lernalgorithmen für die Klassifizierung mit $|C| > 2$ in zwei Punkten. Die *unkombinierten Algorithmen* sind Algorithmen, welche speziell für den Umgang mit mehr als zwei Klassen konzipiert wurden. Ein Entscheidungsbaum wird beispielsweise zu den unkombinierten Algorithmen gezählt. Die *aggregierten Algorithmen* dagegen reduzieren die Aufgabe der Klassifizierung mit mehreren Klassen in eine Aufgabe mit mehreren binäre Klassifizierungen. Dazu wird für jede Klasse $c \in C$ ein binärer Klassifikator trainiert. Der Multi-Klassen Klassifikator wird dann als eine Kombination aus mehreren binärer Klassifikatoren definiert. Die Beschreibungen der Ansätze *one-versus-all*, *one-versus-all* und *error-correction output coding* lehen sich an Mohri/Rostamizadeh/Talwalkar (2012) an.[5]

One-versus-all

Beim *one-versus-all* Verfahren werden $|C|$ binäre Klassifikatoren $h_c : X \mapsto \{-1, 1\}, c \in C$ trainiert. Die Aufgabe jedes Klassifikators ist es, die Klasse c von allen anderen Klassen zu unterscheiden. Dazu werden die Klassen der Testdokumente vor dem Trainieren angepasst. Dabei werden den Testdokumenten, welche zur Klasse c gehören die Klasse 1 zugewiesen. Die Testdokumente, welche nicht zur Klasse c gehören werden zur Klasse -1 zugeordnet. Somit kann das *one-versus-all* Verfahren entsprechend der Gleichung 2.1 ausgedrückt werden.[6]

$$\forall x \in X, h(x) = \arg\max_{c \in C} h_c(x) \tag{2.1}$$

One-versus-one

Beim *one-versus-one* Verfahren wird für jedes Paar unterschiedlicher Klassen $(c, c') \in C^2, c \neq c'$ ein binärer Klassifikator $h_{cc'} : X \mapsto \{-1, 1\}$ trainiert, welcher die Dokumente hinsichtlich der Klassen c und c' unterscheiden soll. Ein Dokument x wird mit den binären Klassifikatoren $h_{cc'}$ klassifiziert. Für eine binäre Klassifizierung mit den Klassen c und c' bedeutet 1 als Resultat, dass x der Klasse c' zugeordnet wird. Nachdem der Durchführung aller binären Klassifizierungen wird dem Dokument x die häufigste vorausgesagte

[5] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 183.

[6] Vgl. a. a. O., S. 198 f.

Klasse zugeordnet. Die one-versus-one Technik kann gemäß der Gleichung 2.2 definiert werden.[7]

$$\forall x \in X, h(x) = \arg\max_{c' \in C} |\{c : h_{cc'}(x) = 1\}| \tag{2.2}$$

Error-correction output coding

Die Idee der *error-correction output coding* (ECOC) Technik basiert auf die Verwendung von error-correction codes. Dazu wird jeder Klasse ein Vektor mit n Dimensionen zugewiesen. Die Koeffizienten können die Werte 0 und 1 annehmen. Ein Beispiel für error-correction codes für die Klassifizierung mit mehreren Klassen wird in Abbildung 2.1 dargestellt. Für jede Spalte wird ein binär Klassifikator trainiert. Ein Klassifikator soll hierbei zwischen zwei Gruppen bestehend von Textdokumenten möglicherweise bestehend aus mehreren Klassen unterscheiden. Für die ECOC in der Abbildung 2.1 werden demnach sechs binäre Klassifikatoren trainiert. Für die Klassifizerung eines Textdokumentes x werden alle binären Klassifikatoren angewendet. Das Ergebnis kann als ein Zeilenvektor interpretiert werden. Mit einem Vergleich des Zeilenvektors mit den Zeilenvektoren der Klassen kann die Zielklasse ermittelt werden. Die Klasse mit einem Zeilenvektor, das einen minimalen Hamming-Abstand aufweist kann dann zum Textdokument x zugeordnet werden.[8,9]

2.1.3 Evaluation einer Hypothese

Metriken

Die Grundlage für die Performanzmaße ist die Kontingenztabelle 2.1. Das Ziel der Performanzanalyse ist es, eine Hypothese hinsichtlich der Nützlichkeit zu untersuchen. Zur Evaluation von Multi-Klassen Klassifikatoren weist Lewis (1991) auf die Verwendung einer Konfusionsmatrix hin. Für die

[7] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 199.

[8] Vgl. a. a. O., S. 202.

[9] Vgl. Dietterich, Thomas G./Bakiri, Ghulum: Solving Multiclass Learning Problems via Error-correcting Output Codes. J. Artif. Int. Res. 2 Januar 1995, Nr. 1 ⟨URL: http://dl.acm.org/citation.cfm?id=1622826.1622834⟩, ISSN 1076–9757, S. 264 ff.

Codes

Klassen	1	2	3	4	5	6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
0	1	1	0	1	1

Neues Testdokument x

In Anlehnung an: Mohri/Rostamizadeh/Talwalkar (2012), S. 202

Abbildung 2.1: Aufbau und Funktionsweise von ECOC für die Multi-Klassen Klassifizierung

Tabelle 2.1: Kontingenztabelle zur Analyse der Performanz

	Zuweisung ist korrekt	Zuweisung ist nicht korrekt
zugewiesen	true positive (tp)	false positive (fp)
nicht zugewiesen	false negative (fn)	true negative (tn)

In Anlehnung an: Lewis (1991), S. 313

folgenden Beschreibungen wird von einer binären Klassifizierung ausgegangen. Die Kontingenztabelle enthält vier Informationen mit denen sich Performanzmetriken berechnen lassen. Die Spalten der Kontingenztabelle geben die tatsächlich korrekte Klasse an. Mit den Spalten wird demnach die Wahrheit aus Sicht des Nutzers dargestellt. In den Zeilen wird jeweils entschieden, ob ein Dokument zu einer Klasse zugeordnet wurde oder nicht. Die Zeilen entsprechen der Sicht des Systems. Die Diagonale von links oben nach rechts unten definiert demnach die Häufigkeit der korrekt klassifizierten Dokumente. Mit diesen beiden Sichten können vier mögliche Situationen betrachtet werden.[10,11]

[10] Vgl. Lewis, David D.: Evaluating Text Categorization. In Proceedings of the Workshop on Speech and Natural Language. Stroudsburg, PA, USA: Association for Computational Linguistics, 1991, HLT '91 ⟨URL: http://dx.doi.org/10.3115/112405.112471⟩, S. 313.

[11] Vgl. Yang, Yiming: An Evaluation of Statistical Approaches to Text Categorization. Inf. Retr. 1 Mai 1999, Nr. 1-2 ⟨URL: http://dx.doi.org/10.1023/A:1009982220290⟩, ISSN 1386–4564, S. 75.

Tabelle 2.2: Metriken zur Evaluation der Performanz

Metrik	Definition
accuracy	$acc = \frac{tp+tn}{tp+fp+tn+fn}$
precision	$p = \frac{tp}{tp+fp}$
recall	$r = \frac{tp}{tp+fn}$
f-measure	$F(r,p) = \frac{1}{\alpha\frac{1}{p}+(1-\alpha)\cdot\frac{1}{r}} = \frac{(\beta^2+1)\cdot p\cdot r}{\beta^2\cdot p+r}$

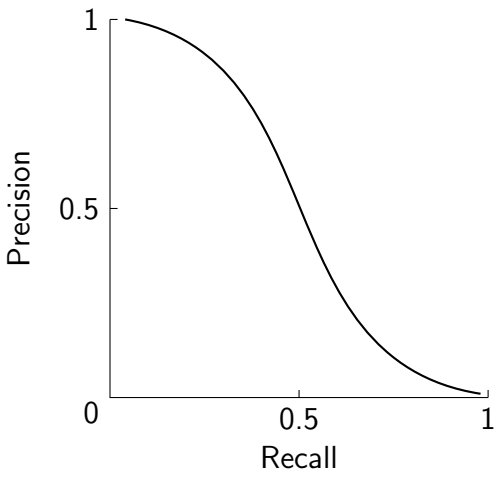
In Anlehnung an Yang (1999), S. 5

Die Tabelle 2.2 gibt eine Übersicht über die relevanten Metriken für die Text Klassifizierung. Die *Richtigkeit* (engl. accuracy) kann unter Verwendung der Kennzahlen aus der Kontingenztabelle entspricht dem Anteil der korrekt klassifizierten Exemplare bezogen auf alle klassifizierten Exemplare.[12]

Die *accuracy* bringt jedoch ein Problem mit sich. Unter der Annahme, dass beispielsweise von 100 E-Mails 99 korrekterweise als *Nicht Spam* klassifiziert werden und nur eine E-Mail irrtümlich als *Nicht Spam* klassifiziert wird entspricht dies einer *accuracy* von $\frac{99}{100} = 99,00$ %. Demnach könnte eine Funktion, welche immer alle E-Mails unabhängig vom Textinhalt als *Nicht Spam* klassifiziert eine Richtigkeit von 99.00 % erzielen. Dadurch würde die *accuracy* keine Hinweise auf die Nützlichkeit eines Algorithmus geben. Um diesem Problem entgegen zu wirken werden die Metriken *precision* und *recall* ermittelt. Die *precision* berücksichtigt die zur Klasse zugewiesenen Exemplare und gibt den Anteil der korrekterweise zugewiesenen Exemplare. Mit *recall* werden $tp + fn$ Exemplare betrachtet, dessen Zuordnung zur Klasse korrekt wären. Von dieser Anzahl an Exemplaren wird der Anteil der zur Klasse zugewiesenen Exemplare ermittelt.[13,14]

Nach Lewis (1991) kann ein Klassifikator ein hohes recall erreichen, indem es die Dokumente selten nicht zur Klasse zuordnet, oder eine hohe precision erreichen, indem es selten Dokumente zur Klasse zuordnet. Zur Evaluation der Effektivität einer Hypothese seien demnach beide Metriken - recall und precision - notwendig. In diesem Zusammenhang wird in Gordon/Kochen (1989) auf ein Zielkonflikt zwischen den Metriken precision und recall hin-

[12] Vgl. Yang: Inf. Retr. 1 [1999], S. 75.
[13] Vgl. a. a. O., S. 74.
[14] Vgl. Lewis: Evaluating Text Categorization, S. 313.



In Anlehnung an: Gordon/Kochen (1989), S. 146

Abbildung 2.2: Idealtypischer Trade-off zwischen precision und recall

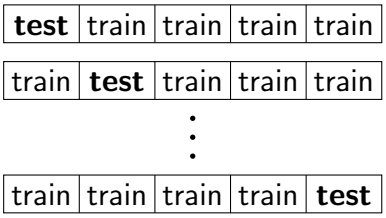
gewiesen. Die Abbildung 2.2 illustriert einen idealtypischen recall-precision Trade-off bzw. Zielkonflikt. Für eine hohe Performanz sollen Precision und recall sollen möglichst hohe prozentuale Werte erzielen. Die kombinierende Metrik F-measure (Van Rijsbergen, 1979) liefert für gleichmäßig hohes precision und recall hohe Werte. Wenn entweder precision oder recall niedrig ausfällt, nimmt F-measure einen Wert nahe null an.[15,16]

Kreuzvalidierung

Die Kreuzvalidierung ist ein Verfahren, dass eingesetzt werden kann, wenn der Exemplarraum X klein ist, sodass das Trennen von X in eine Trainings- und Testmenge zu einer unzureichendenden Trainingsmenge führt. Die Abbildung 2.3 illustriert die prinzipielle Vorgehensweise bei der Kreuzvalidierung. Der Exemplarraum X bestehend aus m Exemplaren wird in n zufälligen Untermengen partitioniert. Die i-te Partition besteht damit aus m_i klassifizierten Exemplaren $((x_{i1}, c_{i1}), ..., (x_{im_i}, c_{im_i}))$. Für alle $i \in [1, n]$ wird ein Modell bzw. eine Hypothese h_i auf Basis aller Partitionen bis auf die i-te Partition trainiert. Mit der i-ten Partition wird die erlernte Hypothese h_i getestet. Die

[15] Vgl. Lewis: Evaluating Text Categorization, S. 313.

[16] Vgl. Gordon, Michael/Kochen, Manfred: Recall-precision Trade-off: A Derivation. J. Am. Soc. Inf. Sci. 40 Mai 1989, Nr. 3 ⟨URL: http://dx.doi.org/10.1002/(SICI)1097-4571(198905)40:3<145::AID-ASI1>3.0.CO;2-I⟩, ISSN 0002–8231, S. 145 f.



In Anlehnung an: Mohri/Rostamizadeh/Talwalkar (2012), S. 6

Abbildung 2.3: Kreuzvalidierung mit 5 Partitionen

Kreuzvalidierung mit $n = m$ wird *Leave-one-out* Kreuzvalidierung genannt, da für jede Iteration genau ein Exemplar von der Trainingsmenge separiert wird und als Testexemplar dient.[17]

2.1.4 Vektorraummodell

Das Vektorraummodell wurde initial von Salton/Wong/Yang (1975) eingeführt. Die Hypothese der Multimenge von Wörtern (engl. bag of words) bildet dabei die Grundlage des Vektorraummodells. Beispielsweise kann eine Multimenge $\{a, b, b, c, c, c\}$ mit dem Vektor $\{1, 2, 3\}$ ausgedrückt werden. Die Voraussetzung ist, dass über die Reihenfolge der Angabe von Häufigkeiten im Vektor eine Vereinbarung getroffen wird. Somit kann eine Häufigkeit einem linguistischen Element zugeordnet werden. Das Vektorraummodell definiert die Grundlage für eine mögliche Repräsentation der Merkmale. Einige Methoden der Merkmalsvorbereitung oder Lernverfahren nutzen die Struktur des Vektorraummodells.[18]

Die Kernidee des Vektorraummodells ist es die Dokumente als Vektoren in einem Vektorraum darzustellen. In dieser Arbeit werden die Begriffe *Wort* und *Merkmal* als Synonym betrachtet. Ein Wort wird dabei als eine Sequenz bestehend aus Zeichen aus einem Alphabet verstanden. Ein Vokabular V repräsentiert dabei die Menge aller in der Trainingsmenge vorkommenden Wörter. Ein Dokument kann innerhalb des Vektorraummodells als ein Vektor $\vec{d} = (tf(t_1, d), ..., tf(t_{|V|}, d))$ repräsentiert werden. Die Dimensionen des

[17] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 5 f.

[18] Vgl. Turney, Peter D./Pantel, Patrick: From Frequency to Meaning: Vector Space Models of Semantics. J. Artif. Int. Res. 37 Januar 2010, Nr. 1 ⟨URL: http://dx.doi.org/10.1109/MSP.2007.914237⟩, ISSN 1076–9757, S. 147.

Dokumentvektors können dabei mit der Häufigkeit $tf(t_i, d)$ des Vorkommens eines Wortes t_i innerhalb des Dokumentes d assoziiert werden.[19,20]

In einer Wort-Dokument Matrix beziehen sich die Zeilen auf die Worte und die Spalten auf die Dokumente. Eine Wort-Dokument Matrix D kann entsprechend der Gleichung 2.3 aussehen. Die Worthäufigkeit tf_{ij} bezieht auf das Wort w_i im Dokument d_j.[21]

$$D = \begin{pmatrix} tf_{1,1} & \cdots & tf_{1,j} \\ \vdots & \ddots & \vdots \\ tf_{i,1} & \cdots & tf_{i,j} \end{pmatrix} \tag{2.3}$$

2.2 Natural Language Processing

2.2.1 Tokenisierung

Die innerhalb dieser Arbeit beschriebenen Lernalgorithmen basieren in der Regel auf das Vorhandensein von Wörtern oder auf Worthäufigkeiten innerhalb von Textdokumenten. Die Segmentierung von Textinhalten in die Worteinheiten ist die Voraussetzung der Ermittlung von Worthäufigkeiten. Dieser Vorgang wird im Kontext der Computerlinguistik auch Tokenisierung genannt.[22]

Ein intuitiver Ansatz könnte die Segmentierung des Textinhalts anhand der vorkommenden Leerzeichen sein. Die Problematik dabei besteht in der Mehrdeutigkeit von Satzzeichen. Ein Punkt kann beispielsweise neben der Verwendung für das Beenden eines Aussagesatzes auch unter anderem zur Trennung bei Datumsangaben und Tausenderstellen in Zahlen verwendet werden. Die Lösung der Mehrdeutigkeiten liegt in der Disambiguierung. In Grefenstette/Tapanainen (1994) werden verschiedene Lösungen, zur Erkennung von Abkürzungen und Trennzeichen in Zahlen, unter Verwendung von regulären

[19] Vgl. Shawe-Taylor, John/Cristianini, Nello: Kernel Methods for Pattern Analysis. Cambridge University Press, 6 2004, 474 Seiten, ISBN 9780521813976, S. 328 f.

[20] Vgl. Turney/Pantel: J. Artif. Int. Res. 37 [2010], S. 147.

[21] Vgl. Shawe-Taylor/Cristianini: Kernel Methods for Pattern Analysis, S. 329.

[22] Vgl. Jackson, P./Moulinier, I.: Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization. John Benjamins Pub., 2002, Natural language processing, ISBN 9789027249890, S. 10.

Ausdrücken präsentiert. Eine Übersicht über die verschiedenen Verfahren zur Tokenisierung wird in Carstensen et al. (2009) gegeben.[23,24]

2.2.2 Normalisierung

Die Normalisierung eines Textinhalts kann das Ziel verfolgen, morphologisch unterschiedliche Wörter auf einen gemeinsamen Nenner zu normalisieren. Dies kann möglicherweise das Vokabular reduzieren und die Häufigkeit einiger Wörter steigern. Dazu können Verfahren zur Stammformreduktion (Stemming) aus dem Forschungsgebiet des Information Retrieval angewendet werden. Der Porter (1980) stemmer und Lovins (1968) stemmer basieren beispielsweise auf der Entfernung des Suffixes. In Hull (1996) und Frakes (1992) und wird eine Übersicht über verschiedene Stemming Verfahren gegeben. Ein anderes Verfahren zur Normalisierung ist die Lemmatisierung. Durch die Lemmatisierung werden Wörter in ihre Grundform zurückgeführt. In Uysal/Gunal (2014) betrachten die Autoren wie die Vorverarbeitung sich auf die Performanz der Text Klassifizierung im Allgemeinen auswirkt. Silva Conrado/Laguna Gutiérrez/Rezende (2012) evaluieren die Auswirkung verschiedener Normalisierungstechniken auf die Klassifizierung von portugiesischem Text.[25]

2.2.3 Spracherkennung

Zur Unterstützung einer mehrsprachigen Text Klassifizierung könnte für jede Sprache eine separate Trainingsmenge und damit einen separaten Klassifikator eingerichtet werden. Somit kann für die Klassifizierung eines neuen Dokumentes ein Klassifikator mit derselben Sprache des Dokumentes verwendet werden. Dazu müsste für jede zu unterstützende Sprache ein Klassifikator bzw. Modell trainiert werden. Zur Auswahl des richtigen Klassifikators wird

[23] Vgl. Grefenstette, Gregory/Tapanainen, Pasi: What is a word, What is a sentence? Problems of Tokenization. In Proceedings of the 3rd Conference on Computational Lexicography and Text Research. Budapest, 1994, S. 3 ff.

[24] Vgl. Carstensen, Kai-Uwe et al. (Hrsg.): Computerlinguistik und Sprachtechnologie: Eine Einführung (German Edition). 3. Auflage. Spektrum Akademischer Verlag, 11 2009, 736 Seiten, ISBN 9783827420237, S. 267 ff.

[25] Vgl. Hull, David A.: Stemming Algorithms: A Case Study for Detailed Evaluation. J. Am. Soc. Inf. Sci. 47 Januar 1996, Nr. 1 ⟨URL: http://dx.doi.org/10.1002/(SICI)1097-4571(199601)47:1<70::AID-ASI7>3.3.CO;2-Q⟩, ISSN 0002–8231, S. 2.

ein Verfahren zur automatisierten Erkennung der Sprache des Dokumentes benötigt. Eine Möglichkeit zur automatisierten Spracherkennung ist ein N-Gramm basierter Ansatz nach Cavnar/Trenkle (1994).[26]

Das Verfahren von Cavnar/Trenkle (1994) basiert dabei auf der Annahme, dass es in allen Sprachen eine Menge von Wörtern gibt, welche alle anderen Wörter hinsichtlich ihrer Häufigkeit übertrifft. Im ersten Schritt des Verfahrens wird das gegebene Dokument in die Worteinheiten tokenisiert. Ziffern sowie Satzzeichen werden dabei verworfen. Jedem Wort wird ein Unterstrich jeweils am Anfang und am Ende angefügt, damit die Übereinstimmung am Wortanfang und -ende unterschieden werden kann. Danach werden für jedes Wort mehrere N-Gramme mit $N = 1$ bis 5 generiert. Anschließend werden die Häufigkeiten der verschiedenen N-Gramme gebildet und absteigend sortiert. Das Ergebnis kann als ein N-Gramm Häufigkeitsprofil interpretiert werden. Die Abbildung 2.4 zeigt den idealtypischen Verlauf der Zipf'schen Verteilung anhand der Häufigkeit der N-Gramme mit $N = 1$ bis 5 für dieses Dokument. Cavnar/Trenkle (1994) beobachten ohne Formalitäten bezüglich der häufigsten etwa 300 N-Gramme eine hohe Korrelation zur Sprache. Demnach kann die Sprache eines Dokumentes automatisiert erkannt werden, indem die N-Gramm Häufigkeiten für das Dokument berechnet und mit den idealtypischen N-Gramm Häufigkeitsverteilung vordefinierter Sprachen verglichen wird.[27]

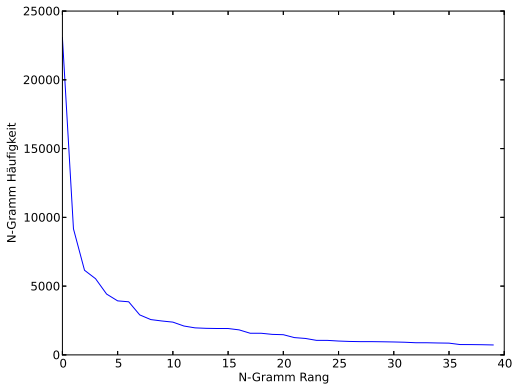
2.3 Methodisches Vorgehen

Die idealtypischen Schritte bei der automatisierten Klassifizierung von Textinhalten definieren den roten Faden für das Hauptkapitel Text Klassifizierung. Die Phasen der Klassifizierung von Textinhalten werden im Rahmen dieser Arbeit in den drei folgenden Phasen unterteilt. Der Prozess setzt voraus, dass eine Trainingsmenge in Form von Textdaten zur Verfügung steht.

Vorbereitung der Merkmale: Im Allgemeinen zielt die Vorbereitungsphase auf die Bereitstellung von optimalen Merkmalen für das Lernen von Modellen. Die Qualität bzw. Performanz der erlernten Modelle zur

[26] Vgl. Carstensen et al.: Computerlinguistik und Sprachtechnologie: Eine Einführung (German Edition), S. 267 f.

[27] Vgl. Cavnar, William B./Trenkle, John M.: N-Gram-Based Text Categorization. In In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. Las Vegas, US, 1994, S. 2 ff.



In Anlehnung an: Cavnar/Trenkle (1994), S. 3

Abbildung 2.4: Verteilung der absteigend sortierten N-Gramm Häufigkeiten dieses Dokuments

Klassifizierung hängt unmittelbar von der Qualität der dazu verwendeten Merkmale. Daher sollten die verwendeten Merkmale verschiedenen Zielklassen möglichst gut repräsentieren. Im Speziellen kann die Vorbereitung der Merkmale im Wesentlichen in drei weitere Themengebiete untergliedert werden. Im Rahmen der Merkmalsselektion werden irrelevante bzw. klassenübergreifend vorkommende Wörter entfernt. Mit der Merkmalsgewichtung können die aus der Merkmalsselektion resultierenden Merkmale unter verschiedenen Aspekten gewichtet werden. Die Merkmalstransformation beinhaltet Methoden zur Abbildung von Merkmalen

Lernen von Modellen: Mit der Anwendung von Lernverfahren können mathematische Modelle basierend auf eine kategorisierte Merkmalsmenge erlernt werden. Je nachdem welche Algorithmen dabei eingesetzt werden, können beispielsweise probabilisitische oder lineare Modelle erlernt werden.

Evaluierung der Modelle: In der Evaluierungsphase wird ein erlerntes Modell anhand einer Testmenge evaluiert. Dabei werden einzelne Exemplare aus der Testmenge klassifiziert. Die Performanz des Modells kann unter Verwendung der Metriken in Kapitel 2.1.3 analysiert werden.

Kapitel 3

Text Klassifizierung

3.1 Merkmale

Der erste Schritt der Text Klassifizierung liegt darin Text Exemplare der verschiedenen zu berücksichtigenden Klassen zu sammeln. Damit steht für das Lernen von Konzepten die Menge $X = (x_1, y_1), ..., (x_n, y_n)$ mit n Exemplaren zur Verfügung. Die Beschaffung der notwendigen Exemplare wird in dieser Arbeit nicht weiter betrachtet. Möglicherweise sind die benötigten Informationen nicht in Textform gegeben. Verfahren für die Extraktion von Textinhalten aus Bildern werden im Werk Optical Character Recognition von Mori/Nishida/Yamada (1999) beschrieben. Die Extraktion von Textinhalten aus Audioinhalten wird im Werk Speech and Language Processing von Jurafsky/Martin (2008) behandelt. Die beiden zuvor genannten Werke werden für weitere Ausführungen nicht verwendet.

3.1.1 Selektion

Der rechnerische Aufwand für einen induktiven Klassifizierer hängt von der Anzahl der Koeffizienten eines Vektors ab. Ein Vektor repräsentiert jeweils ein Dokument. Die Koeffizienten entsprechen den jeweiligen Merkmalen. Für die Reduzierung des Rechenaufwandes können Methoden aus der Merkmalsselektion verwendet werden. Damit kann aus der Menge der Merkmale M

eine kleinere Untermenge M' erhalten werden, welche die relevanteren bzw. nützlicheren Merkmale im Vergleich zu den anderen Merkmalen enthält.[1]

Für die weitere Betrachtung wird angenommen, dass das Vorkommen eines Wortes in einem Dokument jeweils ein Merkmal repräsentiert. Für die Merkmalsselektion wird ein Wert für jedes Wort zur Evaluation berechnet. Für die Berechnung des Evaluationswertes stehen verschiedene Funktionen zur Auswahl. Typischerweise hängt die Interpretation der Evaluationswerte von der verwendeten Evaluationsfunktion f ab. Nach der Berechnung der Werte werden anschließend die Begriffe ausgewählt, welche für f einen Wert über ein bestimmtes Maximum bzw. unter ein bestimmtes Minimum geben. Idealtypische Evaluationsfunktionen können in der Informationstheorie und der Statistik wiedergefunden verwendet.[2]

χ^2 Statistik

Die χ^2 Statistik wird in Experimenten dazu verwendet um unter Berücksichtigung der initialen Hypothese die Unabhängigkeit bzw. die Differenz zwischen den beobachteten und erwarteten Ergebnissen zu messen. Je niedriger der χ^2 Wert für ein Wort w und einer Klasse i ist, desto unabhängiger ist das Wort w von der Klasse i. Aus diesem Grunde werden die Wörter selektiert, welche einen hohen χ^2 Wert liefern. Im übertragenen Sinne kann die Unabhängigkeit zwischen einem Wort w und einer Klasse i untersucht werden.[3]

Sei n die Anzahl der Dokumente, $p_i(w)$ die Wahrscheinlichkeit der Klasse i für das Dokument, welches das Wort w enthält, P_i der globale Anteil der Dokumente, welche zur Klasse i gehören, und $F(w)$ der globale Anteil der Dokumente, welche das Wort w beinhalten. Dann kann die χ^2 Statistik für ein Wort zwischen dem Wort w und der Klasse i entsprechend der Gleichung 3.1 nach Aggarwal/Zhai (2012) berechnet werden.[4]

$$\chi_i^2(w) = \frac{n\,F(w)^2\,(p_i(w) - P_i)^2}{F(w)\,(1 - F(w))\,P_i\,(1 - P_i)} \tag{3.1}$$

[1] Vgl. Debole, Franca/Sebastiani, Fabrizio: Supervised Term Weighting for Automated Text Categorization. In Proceedings of the 2003 ACM Symposium on Applied Computing. New York, NY, USA: ACM, 2003, SAC '03 ⟨URL: http://dx.doi.org/10.1145/952532.952688⟩, ISBN 1–58113–624–2, S. 2.

[2] Vgl. a. a. O., S. 2 f.

[3] Vgl. a. a. O., S. 3.

[4] Vgl. Aggarwal, Charu C./Zhai, ChengXiang (Hrsg.): Mining Text Data. 2012. Auflage. Springer, 2 2012, 535 Seiten, ISBN 9781461432227, S. 170.

Gini Index

Der Gini Index kann zur Quantisierung der Diskriminierung eines Merkmals verwendet werden. Sei $p_i(w)$ die Wahrscheinlichkeit, dass ein Dokument, welches das Wort w enthält, zu einer Klasse i gehört. Dann repräsentieren $p_1(w)...p_k(w)$ die Wahrscheinlichkeiten für k verschiedene Klassen bezogen auf das Wort w. Es gilt dabei die Gleichung 3.2 bezüglich der Summe der Wahrscheinlichkeiten:[5]

$$\sum_{i=1}^{k} p_i(w) = 1 \tag{3.2}$$

Dann kann der Gini Index $G(w)$ hinsichtlich des Wortes w entsprechend der Gleichung 3.3 nach Aggarwal/Zhai (2012) definiert werden:[6]

$$G(w) = \sum_{i=1}^{k} p_i(w)^2 \tag{3.3}$$

Der ermittelte Wert liegt dabei im Interval $(1/k, 1)$. Je größer der Gini Index für ein Wort w ist, desto diskriminierender ist das Wort w. Für Dokumente, welche ein Wort w beinhalten und zu einer bestimmten Klasse gehören, wird für $G(w)$ der Wert 1 ermittelt. Kommt ein Wort w in Dokumenten aus k verschiedenen Klassen vor, so wird für $G(w)$ der Wert $1/k$ erwartet.[7]

Eine unausgewogene Verteilung der Klassen kann die Richtigkeit der Diskirimanz der Merkmale negativ beeinflussen. Durch die Verwendung von globalen Wahrscheinlichkeiten P_i kann die Genauigkeit der Reflektion der Diskriminanz eines Wortes w hinsichtlich der verschiedenen Klassen erhöht werden. Seien $P_1...P_k$ die globalen Vereilungen der Dokumente in den verschiedenen Klassen. Dann kann die normalisierte Wahrscheinlichkeit $p'_1(w)$ anhand der Gleichung 3.4 nach Aggarwal/Zhai (2012) berechnet werden:[8]

$$p'_i(w) = \frac{p_i(w)/P_i}{\sum_{j=1}^{k} p_j(w)/Pj} \tag{3.4}$$

[5] Vgl. Aggarwal/Zhai: Mining Text Data, S. 168.
[6] Vgl. a. a. O.
[7] Vgl. a. a. O.
[8] Vgl. a. a. O.

Dann kann der normalisierte Gini Index anhand der normalisierten Wahrscheinlichkeiten $p'_i(w)$ entsprechend der Gleichung 3.5 nach Aggarwal/Zhai (2012) ermittelt werden:[9]

$$G(w) = \sum_{i=1}^{k} p'_i(w)^2 \tag{3.5}$$

Informationsgewinn

Mit dem Informationsgewinn (engl. information gain, mutual information) wird gemessen, wie viel Information eine Zufallsvariable X über eine andere Zufallsvariable Y enthält. Es kann auch als ein Maß für die erwaretete Reduktion der Entropie $H(X)$ durch die enthaltene Information in Y gesehen werden. Im übertragenen Sinne wird für ein Wort w gemessen, wie viel Information w im Bezug auf eine Klasse i enthält. Dadurch können die Wörter selektiert werden, welche verstärkt auf eine bestimmte Klasse hindeuten. Der Informationsgewinn kann gemäß der Gleichung 3.6 berechnet werden.[10,11]

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ H(X) - H(X|Y) &= H(Y) - H(Y|X) \end{aligned} \tag{3.6}$$

Yang und Pedersen geben eine allgemeinere Form zur Berechnung an. Dadurch soll die Messung des Informationsgewinns für ein Wort unter Berücksichtigung aller Klassen erfolgen.[12]

Gegeben sei ein Dokument und das darin enthaltene Wort w. Seien P_i die globale Wahrscheinlichkeit der Klasse i und $p_i(w)$ die Wahrscheinlichkeit für die Klasse i bei einem Vorkommen des Wortes w. Seien k die Anzahl der Klassen und $F(w)$ der Anteil der Dokumente, welche das Wort w beinhalten.

[9] Vgl. Aggarwal/Zhai: Mining Text Data, S. 169.

[10] Vgl. Debole/Sebastiani: Supervised Term Weighting for Automated Text Categorization, S. 3.

[11] Vgl. Cover, T.M./Thomas, J.A.: Elements of Information Theory. Wiley, 2006, ISBN 9780471748816, S. 21.

[12] Vgl. Yang, Yiming/Pedersen, Jan O.: A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ICML '97 ⟨URL: http://dl.acm.org/citation.cfm?id=645526.657137⟩, ISBN 1–55860–486–3, S. 2.

Dann kann der Informationsgewinn IG entsprechend der Gleichung 3.7 nach Aggarwal/Zhai (2012) definiert werden.[13]

$$\begin{aligned} IG(w) = &- \sum_{i=1}^{k} P_i \log(P_i) + F(w) \sum_{i=1}^{k} p_i(w) \log(p_i(w)) \\ &+ (1 - F(w)) \sum_{i=1}^{k} (1 - p_i(w)) \log(1 - p_i(w)) \end{aligned} \tag{3.7}$$

Die Entropie $H(X)$ einer diskreten Zufallsvariable X kann entsprechend der Gleichung 3.8 von Shannon berechnet werden.[14,15]

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \tag{3.8}$$

Punktweise Transinformation

Die punktweise Transinformation $M_i(w)$ für ein Wort w und einer Klasse i definiert sich auf Grundlage der Kookkurrenz von w und i. Es vergleicht die gemeinsame Wahrscheinlichkeit der Kookkurrenz eines Wortes w und der Klasse i mit den unabhängigen Wahrscheinlichkeiten von w und i. Falls es einen echten Zusammenhang zwischen dem Wort w und der Klasse i geben sollte, wird $M_i(w) \gg 0$ sein. Falls es jedoch kein signifikanter Zusammenhang zwischen w und i besteht, wird $p_i(w) \approx P_i$ und damit $M_i(w) \approx 0$. Die punktweise Transinformation kann gemäß der Gleichung 3.9 nach Aggarwal/ Zhai (2012) definiert werden.[16,17]

$$M_i(w) = \log \left(\frac{F(w)\, p_i(w)}{F(w)\, P_i} \right) = \log \left(\frac{p_i(w)}{P_i} \right) \tag{3.9}$$

[13] Vgl. Aggarwal/Zhai: Mining Text Data, S. 169.

[14] Vgl. Cover/Thomas: Elements of Information Theory, S. 14.

[15] Vgl. Shannon, C. E.: A Mathematical Theory of Communication. Bell System Technical Journal, 27 1948, Nr. 3 ⟨URL: http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x⟩, ISSN 1538–7305, S. 393 f.

[16] Vgl. Xu, Yan et al.: A study on mutual information-based feature selection for text categorization. Journal of Computational Information Systems, 3 2007, Nr. 3 ⟨URL: http://doras.dcu.ie/16194/⟩, ISSN 1553–9105, S. 1007 f.

[17] Vgl. Aggarwal/Zhai: Mining Text Data, S. 170.

In den wissenschaftlichen Publikationen zur Merkmalsselektion lautet die Bezeichnung dieses Verfahrens oft mutual information (Transinformation). Unter dem Begriff mutual information werden jedoch die Formeln der pointwise mutual information (punktweise Transinformation) angegeben.[18,19] Dadurch entsteht eine Inkompatibilität der Definition von mutual information aus dem Forschungsbereich der Computerlinguistik und der Informationstheorie. Das unter mutual information angegebene Verfahren zur Merkmalsselektion sollte daher richtigerweise als pointwise mutual information bezeichnet werden.[20]

Wort Intensität

Die Wort Intensität (engl. word strength oder term strength) misst wie informativ ein Wort hinsichtlich der Identifizierung der zugehörigen Dokumente ist. Die Berechnung erfolgt auf Basis der Verteilung eines Wortes w in den zugehörigen Dokumenten. Für Paare von Dokumenten, welche eine große Anzahl von gemeinsamen Worten haben, wird die Existenz einer Relation zwischen den beiden Dokumenten angenommen. Falls die Dokumente d_1 und d_2 bereits in vektorisierter Form vorliegen kann die Cosinus Ähnlichkeit $d_1 \cdot d_2$ ermittelt werden. Anhand der Cosinus Ähnlichkeit und einem festgelegten kann Schwellenwert entschieden werden, ob zwei Dokumente zueinander in Beziehung stehen. Seien w ein Wort, y und x zwei beliebige voneinander unterschiedliche und zueinander in Beziehung stehende Dokumente, welche w beinhalten. Dann kann die Intensität eines Wortes w entsprechend der Gleichung 3.10 definiert werden.[21]

$$s(w) = P(w \in y | w \in x) \tag{3.10}$$

Seien die Anzahl an Dokumentpaaren, bei denen das Wort w in beiden Dokumenten vorkommt N_p und die Anzahl an Dokumentpaaren, bei denen das Wort w im ersten Dokument vorkommt N_e. Dann kann für die Wort Intensität gemäß der Gleichung 3.11 nach Yang/Wilbur (1996) definiert werden.

[18] Vgl. Yang/Pedersen: A Comparative Study on Feature Selection in Text Categorization, S. 3.

[19] Vgl. Aggarwal/Zhai: Mining Text Data, S. 169.

[20] Vgl. Manning, C.D./Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, 1999, ISBN 9780262133609, S. 68.

[21] Vgl. Yang, Yiming/Wilbur, John: Using Corpus Statistics to Remove Redundant Words in Text Categorization. J. Am. Soc. Inf. Sci. 47 Mai 1996, Nr. 5 ⟨URL: http://dx.doi.org/10.1002/(SICI)1097-4571(199605)47:5<357::AID-ASI3>3.3.CO;2-0⟩, ISSN 0002–8231, S. 359.

Für die Wort Intensität wird zusätzlich ein Schwellenwert festgelegt. Die Worte mit einer Intensität kleiner gleich dem Schwellenwert werden selektiert.[22]

$$s(w) \approx \frac{N_p}{N_e} \tag{3.11}$$

Anders als die Methoden Informationsgewinn und χ^2 Statistik wird die Relation zwischen einem Wort und einer Klasse bei der Wort Intensität nicht berücksichtigt.[23]

3.1.2 Gewichtung

Zur Erhöhung der Qualität der Text Klassifizierung werden Merkmale gewichtet. Viele der Verfahren zur Gewichtung stammen aus dem Forschungsbereich Information Retrieval ab und werden in der Regel in angepasster Form für die Text Klassifizierung verwendet. Als Basis dienen die Dokumente in Form des Vektorraummodells.

tf.idf

Die Werte der Koeffizienten eines Vektors, welches ein Dokument repräsentiert, können als Worthäufigkeiten interpretiert werden. Die Merkmale bzw. Worte wären somit durch die Angabe der Worthäufigkeit gewichtet. Das Problem bei der reinen Worthäufigkeit ist, dass alle Worte dieselbe Relevanz haben. Eine Möglichkeit ist die Skalierung einer Worthäufigkeit in Abhängigkeit des Vorkommens des jeweiligen Wortes innerhalb aller Dokumente. Die Idee dabei ist die Reduktion der Gewichtung um einen Faktor, welcher mit der Dokumenthäufigkeit steigt. Dieser Faktor wird inverse Dokumenthäufigkeit (inverse document frequency) genannt. Seien df_t die Dokumenthäufigkeit des Wortes t, N die Anzahl der Dokumente. Dann kann die inverse Dokumenthäufigkeit gemäß der Gleichung 3.12 definiert werden. Die Doku-

[22] Vgl. Yang/Wilbur: J. Am. Soc. Inf. Sci. 47 [1996], S. 359.

[23] Vgl. Yang/Pedersen: A Comparative Study on Feature Selection in Text Categorization, S. 3.

menthäufigkeit repräsentiert die Anzahl an Dokumenten, in denen das entsprechende Wort t vorkommt.[24,25,26]

$$idf_t = \log \frac{N}{df_t} \tag{3.12}$$

Die Wörter können dann entsprechend der Gleichung 3.13 mit der tf.idf Methode gewichtet werden.[27]

$$tf.idf_{t,d} = tf_{t,d} \cdot idf_t \tag{3.13}$$

Zobel und Moffat nennen drei Annahmen, welche möglicherweise in veränderter Form, in allen Gewichtungsmethoden vorkommen. In der $tf.idf$ Gewichtung lassen sich diese Annahmen ebenfalls wiederfinden.[28]

i) seltene Wörter sind nicht weniger wichtig als häufige Wörter (tf Annahme)

ii) mehrmaliges Vorkommen eines Wortes innerhalb eines Dokumentes ist nicht weniger wichtig als das einmalige Vorkommen (idf Annahme)

iii) bei derselben Anzahl an Vorkommen eines Wortes sind längere Dokumente nicht wichtiger als kürzere (Normalisierungsannahme)

Einige der im Rahmen dieser Arbeit dargestellten Verfahren zur Merkmalsgewichtung, wie zum Beispiel die Relevanzhäufigkeit und icf-based, verwenden Angaben über die Dokumenthäufigkeit aus der Tabelle 3.1. Mit a wird die Anzahl der zur Klasse c_i gehörenden Dokumente angegeben, welche das Wort t_k beinhalten. Die Anzahl der Dokumente, welche nicht zur Klasse c_i gehören und das Wort t_k beinhalten wird mit b angegeben. Die Anzahl der zur Klasse

[24] Jones, Karen Spärck: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 1972.

[25] Vgl. Robertson, Stephen: Understanding inverse document frequency: On theoretical arguments for IDF. Journal of Documentation, 60 2004, Nr. 5, S. 2.

[26] Vgl. Salton, Gerard/Buckley, Christopher: Term-weighting Approaches in Automatic Text Retrieval. Inf. Process. Manage. 24 August 1988, Nr. 5 ⟨URL: http://dx.doi.org/10.1016/0306-4573(88)90021-0⟩, ISSN 0306–4573, S. 516.

[27] Vgl. a. a. O.

[28] Vgl. Zobel, Justin/Moffat, Alistair: Exploring the Similarity Space. SIGIR Forum, 32 April 1998, Nr. 1 ⟨URL: http://dx.doi.org/10.1145/281250.281256⟩, ISSN 0163–5840, S. 3.

Tabelle 3.1: Kontingenztabelle für das Wort t_k und der Klasse c_i

	c_i	$\bar{c}_i$
t_k	a	b
$\bar{t}_k$	c	d

In Anlehnung an Liu/Loh/Sun (2009), S. 693

c_i gehörenden Dokumente, welche das Wort t_k nicht beinhalten, entspricht c. Die Anzahl der Dokumente, welche nicht zur Klasse c_i gehören und das Wort t_k nicht beinhalten, wird mit d repräsentiert.[29]

Die Gestaltung von Schemata für die Gewichtung von Wörtern besteht entsprechend der Annahmen aus drei Komponenten. Diese sind die Worthäufigkeits-, Dokumenthäufigkeits- und Normalisierungskomponente. Für jede Komponente kann jeweils ein eigener Ansatz verfolgt werden. Einige der Formel für Schemata zur Gewichtung von Wörtern werden in Tabelle 3.2 dargestellt.[30]

Überwachte Merkmalsgewichtung

Debole/Sebastiani (2003) kritisieren die Wiederverwendung von Evaluierungsfunktionen zur Merkmalsgewichtung aus dem Bereich der Information Retrieval. Im Kontext der Text Klassifizierung eignet sich demnach die inverse Dokumenthäufigkeit nicht. Im Kontext der Information Retrieval ist die Verwendung der inversen Dokumenthäufig plausibel. In diesem Zusammenhang ist ein Wort w, dass in vielen Dokumenten vorkommt, nicht für die Suche hilfreich wenn es darum geht relevante Dokumente von irrelevanten Dokumenten in Abhängigkeit eines Suchtextes q unterscheiden zu wollen. Anders als beim Information Retrieval ist q bei der Text Klassifizierung durch die Traningsdaten gegeben. Somit kann statt der dokumentbasierten Evaluierungsfunktion *idf* der bereits für die Merkmalsselektion berechnete Wert

[29] Vgl. Liu, Ying/Loh, Han Tong/Sun, Aixin: Imbalanced Text Classification: A Term Weighting Approach. Expert Syst. Appl. 36 Januar 2009, Nr. 1 ⟨URL: http://dx.doi.org/10.1016/j.eswa.2007.10.042⟩, ISSN 0957–4174, S. 693.

[30] Vgl. Lee, Joon Ho: Combining Multiple Evidence from Different Properties of Weighting Schemes. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM, 1995, SIGIR '95 ⟨URL: http://dx.doi.org/10.1145/215206.215358⟩, ISBN 0–89791–714–6, S. 3 f.

Tabelle 3.2: Komponenten von Schemata zur Gewichtung von Wörtern

Worthäufigkeit	
binär	Wenn das Wort vorhanden ist, sonst 0 (Worthäufigkeit wird ignoriert)
$tf_{t,d}$	Unveränderte Worthäufigkeit
$1 + \log(tf_{t,d})$	Reduziert die Wichtigkeit der Worthäufigkeit bei Dokumentsammlungen mit stark variierenden Dokumentlängen (logarithmisch)
Kollektionshäufigkeit	
1	Keine Veränderung
$\log \frac{N}{df_t}$	Inverse Dokumenthäufigkeit (idf)
$\log_2 \left(2 + \frac{a}{\max(1,c}\right)$	Relevanzhäufigkeit (rf)
$\log \left(1 + \frac{\lvert C\rvert}{cf_t}\right)$	Inverse Kategoriehäufigkeit (icf)
$\log \left(2 + \frac{a}{\max(1,c)} \cdot \frac{\lvert C\rvert}{cf_t}\right)$	icf-based (icf und rf)
Normalisierung	
1	Unveränderte Wort- und Kollektionshäufigkeit
$\sqrt{\sum_{i=1}^{n} g_i^2}$	Jede Gewichtung g wird durch die euklidische Norm dividiert (Cosinus Normalisierung).

In Anlehnung an: Lee (1995), S. 4

verwendet werden. Die Methoden, welche von der gegebenen Zuordnung zwischen Dokument und Kategorie Gebrauch machen werden zur Gruppe der überwachten Merkmalsgewichtung eingeordnet. Die Evaluierungsfunktionen zur Merkmalsselektion lassen sich auch durch die Informationselemente aus Tabelle 3.1 nach Liu/Loh/Sun (2009) ausdrücken.[31]

Es gibt jedoch keinen Beweis dafür, dass die überwachte Merkmalsgewichtung immer zu besseren Ergebnissen als $tf.idf$ führt. Deng et al. (2004) kommen beispielsweise aufgrund ihrer Experimente zu dem Schluss, dass $tf.\chi^2$ effektiver als $tf.idf$ ist. Die Ergebnisse von Debole/Sebastiani (2003) zeigen, dass χ^2 als überwachte Merkmalsgewichtung nicht stets effektiver als $tf.idf$ ist.[32,33]

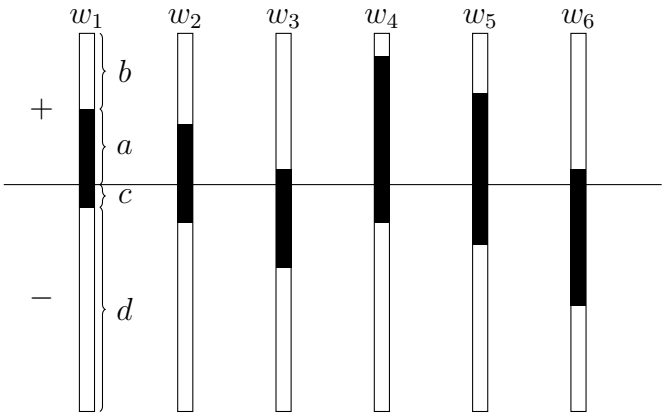
Relevanzhäufigkeit

Die soweit dargestellten Methoden zur überwachten Merkmalsgewichtung sind hinsichtlich der Einteilung in positive und negative Kategorien symmetrisch. Bei der Text Klassifizierung mit mehr als zwei Klassen kann die negative Klasse für eine beliebige Klasse i als alle Klassen ohne i verstanden werden. Der Korpus wird demnach für jede Klasse jeweils in eine positive und eine negative Klasse partitioniert. Die Abbildung 3.1 zeigt eine beispielhafte Verteilung von Dokumenten, welche die sechs Wörter $w_1, ..., w_6$ beinhalten in Bezug auf eine Klasse i. Wenn die Verteilung eines Wortes w_1 in den positiven und negativen Kategorien gleichverteilt wie ein anderes Wort w_3 ist, werden beide Wörter unter Verwendung von χ^2 oder *Informationsgewinn* gleichermaßen gewichtet. Es wird nicht berücksichtigt, wie groß der Anteil der Verteilung in der korrekten bzw. positiven Klasse ist. Je größer der Anteil der Verteilung in der positiven Klasse ist, desto größer ist der Beitrag des entsprechenden Wortes um die richtige Klasse zu finden. Basierend auf dieser Idee wird in Lan et al. (2009) für $tf.rf$ die Gleichung 3.14 angegeben. Die Konstante 2 soll dabei verhindern, dass Wörtern eine Gewichtung von 0

[31] Vgl. Debole/Sebastiani: Supervised Term Weighting for Automated Text Categorization, S. 2 f.

[32] Vgl. Deng, Zhi-Hong et al.: A Comparative Study on Feature Weight in Text Categorization. In Yu, JeffreyXu et al. (Hrsg.): Advanced Web Technologies and Applications. Band 3007, Springer Berlin Heidelberg, 2004 ⟨URL: http://dx.doi.org/10.1007/978-3-540-24655-8_64⟩, ISBN 978–3–540–21371–0, S. 596.

[33] Vgl. Debole/Sebastiani: Supervised Term Weighting for Automated Text Categorization, S. 5.



In Anlehnung an: Lan et al. (2009), S. 725

Abbildung 3.1: Beispielhafte Verteilung von Dokumenten für sechs verschiedene Wörter

gegeben wird. Desweiteren stellt die Verwendung der *max* Funktion sicher, dass im Fall von $C = 0$ eine Division mit 0 vermieden wird.[34]

$$tf.rf = tf \cdot \log_2 \left(2 + \frac{a}{\max(1, c)}\right) \tag{3.14}$$

Der Faktor rf wird als Relevanzhäufigkeit (relevance frequency) bezeichnet, da in der Gleichung nur die Häufigkeit der relevanten Dokumente betrachtet werden.[35] Die gleichen Autoren der $tf.rf$ Merkmalsgewichtung zeigen durch ihre Experimente, dass die Relevanzhäufigkeit andere Methoden zur Merkmalsgewichtung wie χ^2 und $Informationsgewinn$ stets übertrifft.[36]

Inverse Kategoriehäufigkeit

In Wang/Zhang (2013) wird eine neue Methode zur Merkmalsgewichtung vorgestellt. Die Verwendung der inversen Kategoriehäufigkeit wird durch die folgenden zwei Punkte motiviert.

[34] Vgl. Lan, Man et al.: Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 31 2009, Nr. 4, ISSN 0162–8828, S. 726.

[35] Vgl. a. a. O., S. 727.

[36] Vgl. a. a. O., S. 734.

- Die Verteilung eines Wortes sollte kategoriebasiert statt dokumentbasiert betrachtet werden.[37]
- Bei der Verwendung von $tf.rf$ wird ein Korpus in positive und negative Kategorien partitioniert. Dadurch verschwindet die Verteilung eines Wortes zwischen den Kategorien. Es wird angenommen, dass ein Wort mehr dazu beitragen kann zwischen Kategorien zu unterscheiden wenn es möglichst in wenigen Kategorien vorkommt. Dieser Faktor wird jedoch von der Relevanzhäufigkeit nicht berücksichtigt.[38]

Die Kategoriehäufigkeit cf gibt die Anzahl an Kategorien, in denen das Wort w_i vorkommt. Die inverse Kategoriehäufigkeit icf wird analog zur inversen Dokumenthäufigkeit definiert. Seien C die Menge der Klassen, t ein Wort und d ein Dokument. Die überwachte Merkmalsgewichtung $tf.icf$ kann dann entsprechend der Gleichung 3.15 definiert werden.[39]

$$tf.icf_{t,d} = tf_{t,d} \cdot \log\left(1 + \frac{|C|}{cf_t}\right) \tag{3.15}$$

In Wang/Zhang (2013) wird ein zusätzliches Schema icf-based vorgeschlagen. Die icf-based Methode wendet die Vorzüge der Relevanzhäufigkeit rf und der inversen Kategoriehäufigkeit icf an und wird gemäß der Gleichung 3.16 definiert. In den Experimenten von Wang/Zhang (2013) übertrifft icf-based acht existierende Methoden, wie zum Beispiel $tf.rf$ und $tf.idf$.[40,41]

$$\text{icf-based}_{t,d} = tf_{t,d} \cdot \log\left(2 + \frac{a}{\max(1,c)} \cdot \frac{|C|}{cf_t}\right) \tag{3.16}$$

Konfidenzintervall

In Soucy/Mineau (2005) wird eine weitere Methode $ConfWeight$ zur Gewichtung vorgeschlagen, welche auf Konfidenzintervalle basiert. Die

[37] Vgl. Wang, Deqing/Zhang, Hui: Inverse-Category-Frequency based Supervised Term Weighting Schemes for Text Categorization. J. Inf. Sci. Eng. 29 2013, Nr. 2, S. 212.
[38] Vgl. a. a. O., S. 213.
[39] Vgl. a. a. O., S. 214.
[40] Vgl. a. a. O.
[41] Vgl. a. a. O., S. 222.

Schätzung des Anteils der Dokumente, welche ein bestimmtes Wort t beinhalten, basiert auf der Wilson Anteilschätzung $\tilde{p}$. Seien x_t die Anzahl an Dokumenten im Korpus, welche das Wort t beinhalten und n die gesamte Anzahl an Dokumenten. Dann kann das Konfidenzintervall entsprechend der Gleichung 3.17 definiert werden.[42]

$$\tilde{p} \pm 1,96\sqrt{\frac{\tilde{p} \cdot (1-\tilde{p})}{n+3,84}} \tag{3.17}$$

$$\tilde{p} = \frac{x_t + 1,96}{n+3,84} \tag{3.18}$$

Für eine gegebene Kategorie können $\tilde{p}_+$ und $\tilde{p}_-$ durch Anwendung von $\tilde{p}$ auf die entsprechende positive bzw. negative Kategorie im Korpus erhalten werden. Die untere und obere Grenze des 95-prozentigen Konfidenzintervalls werden als $MinPos$ und $MaxNeg$ bezeichnet. Die Intensität des Wortes w für die Kategorie $c \in C$ sowie die Gewichtung $ConfWeight$ werden gemäß den Gleichungen 3.19 und 3.20 definiert. Statt der reinen Worthäufigkeit tf wird in Soucy/Mineau (2005) die logarithmische Variante der Worthäufigkeit verwendet. Die Experimente von Soucy/Mineau (2005) zeigen, dass $ConfWeight$ die Methoden $tf.idf$ und gain ratio übertrifft.[43,44,45]

$$str_{t,c} = \begin{cases} \log_2\left(2 - \frac{MinPos}{MinPos+MaxNeg}\right) & MinPos > MaxNeg, \\ 0 & MinPos \leq MaxNeg \end{cases} \tag{3.19}$$

$$ConfWeight_{t,d} = \log(tf_{t,d}+1) \cdot \max_{c \in C}(str_{t,c})^2 \tag{3.20}$$

[42] Vgl. Soucy, Pascal/Mineau, Guy W.: Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. In Proceedings of the 19th International Joint Conference on Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, IJCAI'05 ⟨URL: http://dl.acm.org/citation.cfm?id=1642293.1642474⟩, S. 2.

[43] Vgl. a. a. O., S. 2 f.

[44] Vgl. a. a. O., S. 6.

[45] Vgl. Lan et al.: Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 [2009], S. 724.

3.1.3 Transformation

Die Merkmalstransformation zielt im Gegensatz zur Merkmalsselektion auf die Erstellung einer neuen und möglicherweise kleineren Menge von Merkmale auf Basis der gegebenen Merkmale bzw. Wörter ab.

Latent semantische Indexierung

Die latent semantische Indexierung (LSI) kann zur Merkmalstransformation verwendet werden. Die LSI Methode transformiert einen Textraum bestehend aus Wörtern in einen approximierten und latent semantischen Raum mit möglicherweise weniger Dimensionen. Im Rahmen der Text Klassifizierung wird eine Wort-Dokument Matrix auf eine neue Niedrig-Rang Wort-Dokument Matrix approximiert. Für die Approximation wird die Singulärwertzerlegung verwendet. Es wird dabei angenommen, dass gemeinsam auftretende Wörter innerhalb einer Kollektion eine ähnliche Semantik aufweisen. In Golub/Loan (2012) wird die Singulärwertzerlegung für eine Matrix A entsprechend der Gleichung 3.21 definiert.[46,47]

$$A = U \Sigma V^T \tag{3.21}$$

Die Wort-Dokument Matrix A mit dem Rang r wird durch die Anwendung der Singulärwertzerlegung in drei Matrizen U, Σ und V^T zerlegt. Nach der Berechnung dieser Matrizen kann die Diagonalmatrix Σ angepasst werden um anschließend die approximierte Matrix A_k zu berechnen. Dazu werden die kleinsten $r - k$ Singulärwerte auf 0 gesetzt. Mit der neuen Wort-Dokument Matrix A_k werden die Dokumente im k Dimensionalen latent semantischem Raum repräsentiert. Die Interpretation der Komponenten der Singulärwertzerlegung im Rahmen der LSI ist in Tabelle 3.3 dargestellt. Die Abbildung 3.2 illustriert dabei die Zusammensetzung der approximierten Matrix A_k.[48]

Die LSI Methode funktioniert für Aufgaben aus dem Information Retrieval gut. In Zusammenhang mit der Text Klassifizierung ist die LSI Methode

[46] Vgl. Deerwester, Scott et al.: Indexing by latent semantic analysis. Journal Of The American Society For Information Science, 41 1990, Nr. 6, S. 1.

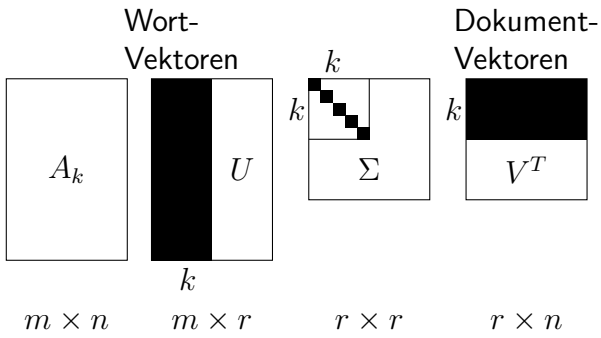
[47] Vgl. Golub, Gene H./Loan, Charles F. Van: Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences). fourth edition Auflage. Johns Hopkins University Press, 12 2012, 784 Seiten, ISBN 9781421407944, S. 76 ff.

[48] Vgl. Berry, Michael W./Dumais, Susan T./O'Brien, Gavin W.: Using Linear Algebra for Intelligent Information Retrieval. SIAM Rev. 37 Dezember 1995, Nr. 4 ⟨URL: http://dx.doi.org/10.1137/1037127⟩, ISSN 0036–1445, S. 574 ff.

Tabelle 3.3: Interpretation der Komponenten der Singulärwertzerlegung innerhalb der latent semantischen Indexierung

A_k = Approximierte Matrix	m = Anzahl verschiedener Wörter
U = Wort-Vektoren	n = Gesamte Anzahl an Dokumenten
Σ = Singulärwerte	k = Höchstmöglicher Rang der Matrix A_k
V = Dokument-Vektoren	r = Rang der Matrix A

In Anlehnung an: Berry/Dumais/O'Brien (1995), S. 576



In Anlehnung an: Berry/Dumais/O'Brien (1995), S. 576

Abbildung 3.2: Illustration der Singulärwertzerlegung

begrenzt hilfreich, da hierbei die vorliegenden Kategorien nicht in Betracht gezogen werden. Für die Text Klassifizierung wird daher in Sun et al. (2004) eine überwachte latent semantische Indexierung (SLSI) vorgeschlagen. Dabei wird für jede Klasse eine Wort-Dokument Matrix erstellt. Für jede Wort-Dokument Matrix wird dann die Singulärwertzerlegung angewendet.[49]

Die Experimente in Sun et al. (2004) zeigen, dass SLSI im Vergleich zu LSI für die Text Klassifizierung effektiver ist. Zusätzlich wurde in diesem Experiment eine Klassfizierung mit Support Vector Machines ohne SLSI bzw. LSI Merkmalstransformation durchgeführt. Nach Sun et al. (2004) können weder LSI noch SLSI die ursprüngliche Vektorraum Repräsentation übertreffen. Die Anwendung von SLSI führt allerdings nach Sun et al. (2004) durch die reduzierte Dimensionalität zu schnelleren Rechenzeiten.[50]

[49] Vgl. Sun, Jian-Tao et al.: Supervised Latent Semantic Indexing for Document Categorization. In Proceedings of the Fourth IEEE International Conference on Data Mining. Washington, DC, USA: IEEE Computer Society, 2004, ICDM '04 ⟨URL: http://dl.acm.org/citation.cfm?id=1032649.1033524⟩, ISBN 0–7695–2142–8, S. 1 f.

[50] Vgl. a. a. O., S. 4.

3.2 Algorithmen zur Klassifizierung

3.2.1 Entscheidungsbaum

Ein Entscheidungsbaum kann zur Klassifizierung von Daten verwendet werden. Die folgenden Erläuterungen beziehen sich auf die allgemeine Funktionsweise eines Entscheidungsbaums und beschränken sich auf die binäre Klassifizierung von Daten in Anlehnung an die Classification and Regression Tree (CART) Methode (Breiman et al., 1984). Zu den implementierenden Algorithmen gehören unter anderem ID3 (Quinlan, 1986) und C4.5 (Quinlan, 1993) und CART (Breiman et al., 1984), womit ein Entscheidungsbaum auf Grundlage einer Trainingsmenge erstellt werden kann.

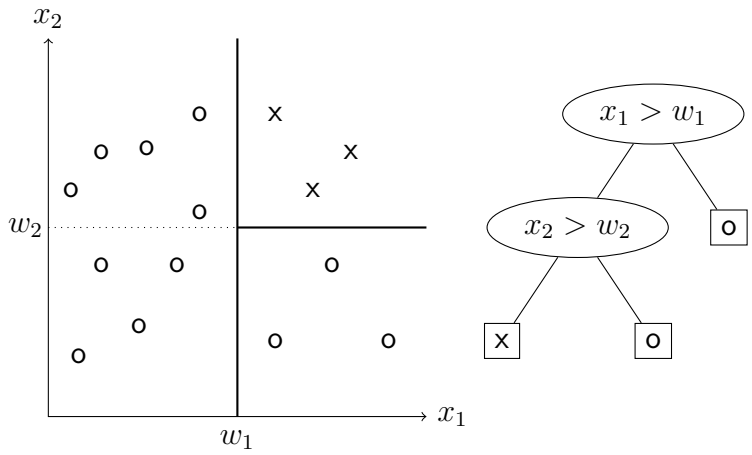
Die Abbildung 3.3 visualisiert die essentielle Idee der hierarchischen Zerlegung der Trainingsmenge. Das Ziel beim Zerlegen der Trainingsmenge ist, dass nach der Zerlegung alle Exemplare in einem Zweig zu einer einzigen Klasse gehören. Ein Knoten im Baum repräsentiert eine Bedingung. In Abhängigkeit der Wahrheit einer Bedingung wird eine Kante ausgewählt. Im Rahmen der Text Klassifizierung kann ein Knoten beispielsweise das Vorkommen eines bestimmten Wortes im Dokument prüfen. Die Kanten können als Entscheidungen interpretiert werden. Die Blätter bestimmen jeweils die Klasse. Für ein zu klassifizierendes Exemplar wird der Entscheidungsbaum von der Wurzel zu einem Blatt unter Berücksichtigung der Bedingungen in den Knoten traversiert.[51,52]

Für den Aufbau eines Entscheidungsbaums muss zunächst entschieden werden nach welchem Merkmal die Eingabemenge bzw. Trainingsmenge zerlegt werden soll. Das Merkmal sollte die Eingabemenge möglichst in zwei Gruppen zerlegen, welche jeweils nur Exemplare derselben Klasse beinhalten. Zur Auswahl eines Merkmals können die Methoden der Merkmalsselektion aus Kapitel 3.1.1 auf Seite 16 verwendet werden. Nach Hastie/Tibshirani/Friedman (2009) gehören beispielsweise Gini Index und Informationsgewinn zu den idealtypischen Methoden.[53]

[51] Vgl. Alpaydin, Ethem: Introduction to Machine Learning (Adaptive Computation and Machine Learning series). second edition Auflage. The MIT Press, 12 2009, 584 Seiten, ISBN 9780262012430, S. 185 f.

[52] Vgl. Aggarwal/Zhai: Mining Text Data, S. 176 f.

[53] Vgl. Hastie, T./Tibshirani, R./Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2. Auflage. Springer, 2009, Springer Series in Statistics, ISBN 9780387848587, S. 309.



In Anlehnung an: Alpaydin (2009), S. 186

Abbildung 3.3: Funktionsweise eines Entscheidungsbaums

Nach Aggarwal/Zhai (2012) kann im Falle von Blättern mit ungenauer Klassifizierung die Untermenge der Trainingsmenge im entsprechenden Pfad erneut anhand eines neuen Knotens bzw. einer neuen Bedingung in weitere zwei Gruppen zerlegt werden. Das betroffene Blatt kann mit dem neu erstellten Knoten ersetzt werden. So kann der Entscheidungsbaum sukzessive rekursiv vergrößert werden bis die benötigte Genauigkeit der Klassifizierung erreicht wird. Wenn die Zerlegung zu oft stattfindet wird der Entscheidungsbaum hinsichtlich der Trainingsmenge stark optimiert statt eine Verallgemeinerung aus der Trainingsmenge abzuleiten.[54]

Statt der binären Zerlegung könnte die Eingabemenge auch in mehr als zwei Gruppen zerlegt werden. In Hastie/Tibshirani/Friedman (2009) wird diesbezüglich auf mögliche Probleme hingewiesen. Es kann beispielsweise das Problem mit sich bringen, dass die Datenmengen zu schell fragmentiert werden, sodass beim nächsten Knoten eine unzureichende Datenmenge verfügbar ist. Zusätzlich kann es vorkommen, dass Daten zu oft zerlegt werden. Mit einem binären Baum kann die Trainingsmenge kontrollierter in weitere Gruppen aufgeteilt werden bis eine bestimmte Bedingung erfüllt wird. Aus diesen Gründen werden in der Regel binäre Zerlegungen bevorzugt.[55]

[54]Vgl. Aggarwal/Zhai: Mining Text Data, S. 176 f.

[55]Vgl. Hastie/Tibshirani/Friedman: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, S. 311.

3.2.2 Nächste-Nachbarn-Klassifikator

Der Kern eines Nächste-Nachbarn-Klassifikators liegt in der Ähnlichkeitsmessung. Es wird angenommen, dass Dokumente aus derselben Klasse eine gleiche Ähnlichkeit zueinander aufweisen. Die Idee beim k-Nächste-Nachbarn (kNN) Klassifikator ist es, einem Dokument die Klasse der k nächsten bzw. ähnlichsten Dokumente zuzuordnen (Fix/J.L. Hodges, 1951). Seien p das zu klassifizierende Dokument und C die Menge der Klassen, $Score$ die Bewertung der Klasse C_j bezüglich p, $sim(p, q_i)$ die Ähnlichkeit zwischen p und q_i aus der Trainingsmenge und $y(p_i, C_j) \in \{0, 1\}$ die Angabe ob q_i zur Klasse C_j gehört. Dann kann die Entscheidungsregel $f(p)$ für das kNN Verfahren entsprechend der Gleichung 3.22 formuliert werden (Jiang et al., 2012).[56]

$$f(p) = \arg\max_j Score(p, C_j) = \sum_{q_i \in kNN} sim(p, q_i)\, y(q_i, C_j) \tag{3.22}$$

Die Wahl eines Ähnlichkeitsmaßes für zwei Dokumente hängt von der vorliegenden Repräsentation eines Dokumentes ab. Für die Messung der Ähnlichkeit zwischen zwei Dokumentvektoren kann beispielsweise der Euklidische Abstand oder die Cosinus-Ähnlichkeit gebildet werden. Weitere mögliche Methoden sind Kullback-Leibler-Divergenz, L_1-Norm und Jensen-Shannon-Divergenz (Dagan/Pereira/Lee, 1994).[57] In der Regel misst eine Methode entweder die Ähnlichkeit s (engl. similarity) oder Unterschiedlichkeit d (engl. dissimilarity). Die Tabelle 3.4 gibt eine Übersicht über einige mögliche Methoden zur Messung der Ähnlichkeit zwischen zwei Dokumentvektoren $\vec{p}$ und $\vec{q}$. In (Cha, 2007) wird eine umfassende Übersicht über Ähnlichkeitsmaße gruppiert nach der Syntax und Semantik gegeben.[58]

Die Ergebnisse von Beyer et al. (1999) sollen zeigen, dass die Differenz zwischen der Distanz der nächsten und entferntesten Punkte nicht mit der Steigerung der Dimensionalität steigt. Mit Steigender Dimensionalität gehe die Distanz zwischen den nächsten und den entferntesten Punkten gegen Null.

[56] Vgl. Cover, T./Hart, P.: Nearest neighbor pattern classification. Information Theory, IEEE Transactions on, 13 1967, Nr. 1 ⟨URL: http://dx.doi.org/10.1109/TIT.1967.1053964⟩, ISSN 0018–9448, S. 22.

[57] Vgl. Manning/Schütze: Foundations of Statistical Natural Language Processing, S. 301 ff.

[58] Vgl. Cha, Sung-Hyuk: Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. International Journal of Mathematical Models and Methods in Applied Sciences, 1 2007, Nr. 4 ⟨URL: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.154.8446⟩, S. 301 ff.

Tabelle 3.4: Mögliche Ähnlichkeits- bzw. Unterschiedlichkeitsmaße

Skalarprodukt	$s_{SP} = \vec{p}\,\vec{q} = \sum_{i=1}^{n} p_i\, q_i$
Cosinus	$s_{cos} = \frac{\vec{p}\,\vec{q}}{\|\vec{p}\|\,\|\vec{q}\|}$
Euklidischer Abstand	$d_{Euc}\|\vec{p} - \vec{q}\| = \sum_{i=1}^{n}(p_i - q_i)^2$
Manhattan Abstand	$d_M = \sum_{i=1}^{n} \|p_i - q_i\|$
Kullback-Leibler-Divergenz	$d_{KL} = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$
Jensen-Shannon-Divergenz	$d_{JS} = \frac{1}{2}\left[\sum_{i=1}^{n} p_i \log\left(\frac{2\,p_i}{p_i+q_i}\right) + \sum_{i=1}^{n} q_i \log\left(\frac{2\,q_i}{p_i+q_i}\right)\right]$

In Anlehnung an: Cha (2007), S. 301 ff.

Dadurch wurde die Bedeutung der Suche nach den nächsten Punkten bei hoher Dimensionalität in Frage gestellt. In zwei Dimensionalen Räumen ist beispielsweise die Interpretation der Verteilung der Datenpunkte relativ einfach. Es kann ist relativ leicht angenommen werden, dass Metriken, wie zum Beispiel der euklidische Abstand, interpretierbare bzw. bedeutende Ergebnisse liefern. Dies sei jedoch in Räumen mit hoher Dimensionalität im Allgemeinen nicht wahr. Als Lösung zu dieser Problematik wird in Hinneburg/ Aggarwal/Keim (2000) eine allgemeinere Nächste-Nachbarn-Methode vorgeschlagen, welches dynamisch die relevanten Dimensionen bestimmt und diese für das Finden der nächsten Nachbarn verwendet.[59]

Eine Möglichkeit zur Bestimmung der k nächsten Nachbarpunkte eines zu klassifizierenden Textdokumentes p ist die Brute-Force-Methode. Für Vektoren mit einer großen Anzahl an Dimensionen kann die Berechnung der k nächsten Nachbarn jedoch Speicher- und Zeitintensiv werden. Daher wurden verschiedene Techniken entwickelt, um möglichen Limitierungen entgegen zu kommen. Beispielsweise können Exemplare, welche wenig zur Diskriminierung zwischen den Klassen beitragen, aus der Trainingsmenge entfernt wer-

[59] Vgl. Hinneburg, Alexander/Aggarwal, Charu C./Keim, Daniel A.: What Is the Nearest Neighbor in High Dimensional Spaces? In Proceedings of the 26th International Conference on Very Large Data Bases. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, VLDB '00 ⟨URL: http://dl.acm.org/citation.cfm?id=645926.671675⟩, ISBN 1–55860–715–3, S. 509 f.

den. Hierzu können die vorgestellten Verfahren aus den Kapiteln 3.1.1 und 3.1.3 verwendet werden. In Bhatia/Vandana (2010) wird eine Übersicht über verschiedene Nächste-Nachbarn Techniken gegeben. Die essentiellen Ideen der Ansätze sowie deren Vor- und Nachteile werden darin dargestellt.[60]

Der traditionelle Ansatz (Cover/Hart, 1967) für die kNN Methode verwendet alle Exemplare aus der Trainingsmenge, um neuen Dokumenten eine Klasse zuzuordnen. Der Aufwand für die Klassifizierung hängt von der Größe der Trainingsmenge ab. Zur Steigerung der Effizienz wird in Jiang et al. (2012) ein Verfahren vorgeschlagen, welches die kNN Methode mit einer Clusteranalyse kombiniert. Dabei wird für eine Klassifizierung die Ähnlichkeit zwischen dem zu klassifizierenden Dokument und der Cluster untersucht. Die Anzahl der Cluster soll kleiner sein als die Anzahl der Exemplare in der Trainingsmenge. Dadurch soll der Aufwand der Ähnlichkeitsanalyse verringert werden. Seien C_i^0 ein Cluster, $ClusterScore(p, C_j)$ die Bewertung des Dokuments p bezüglich der Klasse C_j, $sim(p, C_i^0)$ die Ähnlichkeit zwischen dem Dokument p und dem Cluster C_i^0 und $y(C_i^0, C_j) \in \{0, 1\}$ die Angabe ob das Cluster C_i^0 zur Klasse C_j gehört. Dann die Funktion $f(p)$ zur Klassifizierung eines Dokumentes p gemäß der Gleichung 3.23 definiert werden.[61]

$$f(p) = \arg\max_j ClusterScore(p, C_j) = \sum_{C_i^0 \in kNN} sim(p, C_i^0)\, y(C_i^0, C_j) \quad (3.23)$$

Ein anderer Ansatz für den effizienten Umgang mit hochdimensionalen Dokumentvektoren ist die Verwendung einer approximierten Nächste-Nachbarn Methode (Indyk/Motwani, 1998). Bei einer approximierten Nächste-Nachbarn Methode werden die annähernde nächsten Nachbarn gesucht. Ein Beispiel aus dieser Gruppe von Methoden ist die Locality-Sensitive-Hashing (LSH) Methode. Im Wesentlichen werden dabei Skalarprodukte von zufälligen Vektoren gebildet, um in einer großen Trainingsmenge möglichst schnell annähernd nahe Nachbarn zu finden.[62,63]

[60] Vgl. Bhatia, Nitin/Vandana: Survey of Nearest Neighbor Techniques. CoRR, abs/1007.0085 2010, S. 302 ff.

[61] Vgl. Jiang, Shengyi et al.: An Improved K-nearest-neighbor Algorithm for Text Categorization. Expert Syst. Appl. 39 Januar 2012, Nr. 1 ⟨URL: http://dx.doi.org/10.1016/j.eswa.2011.08.040⟩, ISSN 0957–4174, S. 1504 f.

[62] Vgl. Indyk, Piotr/Motwani, Rajeev: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. New York, NY, USA: ACM, 1998, STOC '98 ⟨URL: http://dx.doi.org/10.1145/276698.276876⟩, ISBN 0–89791–962–9, S. 605.

[63] Vgl. Slaney, M./Casey, M.: Locality-Sensitive Hashing for Finding Nearest Neighbors

3.2.3 Naive Bayes

Die naive Bayes Klassifizierung basiert auf der Annahme, dass alle Merkmale bzw. Wörter bedingt unabhängig voneinander sind. Damit wird die lineare Ordnung der Wörter innerhalb eines Dokuments ignoriert. Zudem ist die Okkurrenz eines Wortes unabhängig von der Okkurrenz eines anderen Wortes. Die getroffenen Annahmen scheinen für die Klassifizierung von natürlichsprachlichen Textinhalten unangemessen. In Domingos/Pazzani (1997) kann jedoch beobachtet werden, dass Bayes'sche Klassifikatoren selbst in Fällen mit starker Abhängigkeit zwischen Merkmalen eine gute Performanz erzielen können.[64,65]

Die Klassifizierung eines Dokumentes $d = \{w_1, ..., w_{|d|}\}$ kann durch die Anwendung der Bayes'schen Regel gemäß der Gleichung 3.24 durchgeführt werden. Dazu wird die maximale A-posteriori-Wahrscheinlichkeit $P(c|d)$ der verschiedenen Klassen aus der Menge C berechnet. Die Klasse c_{MAP} mit der maximalen A-posteriori-Wahrscheinlichkeit (MAP) wird schließlich gemäß der Gleichung 3.25 dem Dokument zugeordnet.[66]

$$P(c|d) = \frac{P(c)\,P(d|c)}{P(d)} \tag{3.24}$$

$$\begin{aligned} c_{MAP} &= \arg\max_{c \in C} P(c|d) \\ &= \arg\max_{c \in C} \frac{P(c)\,P(d|c)}{P(d)} \end{aligned} \tag{3.25}$$

Die Wahrscheinlichkeit $P(d)$ in der Gleichung 3.25 kann als Konstante betrachtet werden. Im Rahmen der Ermittlung der maximalen A-posteriori-Wahrscheinlichkeit $P(c|d)$ kann die Wahrscheinlichkeit $P(d)$ daher vernachlässigt werden. Dadurch kann die Klassifizierung entsprechend der Gleichung 3.26 vereinfacht ausgedrückt werden.[67]

[Lecture Notes]. Signal Processing Magazine, IEEE, 25 2008, Nr. 2 ⟨URL: http://dx.doi.org/10.1109/MSP.2007.914237⟩, ISSN 1053–5888, S. 128 f.

[64] Vgl. Maron, M. E.: Automatic Indexing: An Experimental Inquiry. J. ACM, 8 Juli 1961, Nr. 3 ⟨URL: http://dx.doi.org/10.1145/321075.321084⟩, ISSN 0004–5411, S. 409 f.

[65] Vgl. Lewis, David D.: Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In Proceedings of the 10th European Conference on Machine Learning. London, UK, UK: Springer-Verlag, 1998, ECML '98 ⟨URL: http://dl.acm.org/citation.cfm?id=645326.649711⟩, ISBN 3–540–64417–2, S. 4 ff.

[66] Vgl. Mitchell: Machine Learning, S. 157.

[67] Vgl. a. a. O., S. 157 f.

$$c_{MAP} \propto \arg\max_{c \in C} P(c)\, P(d|c) \tag{3.26}$$

Durch die Annahme der Unabhängigkeit zwischen den Merkmalen folgt die Gleichung 3.27. Die wahrscheinlichste Klasse für ein Dokument ist demnach die Klasse, welche das Produkt der bedingten Wahrscheinlichkeit $P(\{w_1, ..., w_{|d|}|c)$ und der Wahrscheinlichkeit $P(c)$ maximiert. Sei N die Anzahl der Dokumente und N_c die Anzahl der Dokumente in der Klasse c. Dann kann die A-priori-Wahrscheinlichkeit einer Klasse $P(c)$ kann durch die Gleichung 3.28 berechnet werden.[68]

$$\begin{aligned} c_{MAP} &\propto \arg\max_{c \in C} P(c)\, P(w_1, ..., w_{|d|}|c) \\ &\propto \arg\max_{c \in C} P(c) \prod_{i=1}^{|d|} P(w_i|c) \end{aligned} \tag{3.27}$$

$$P(c) = \frac{N_c}{\mathrm{N}} \tag{3.28}$$

Es gibt verschiedene Variationen der naive Bayes Methode, welche sich in der Regel in der Annahme über die Wahrscheinlichkeitsverteilung von $P(\{w \in V\}|c)$ unterscheiden lassen. In Nigam et al. (1998), Kalt (1998), Guthrie/Walker/Guthrie (1994) und Mitchell (1997) werden Variationen der Multinomialverteilung präsentiert. Ein anderes Ereignismodell für die naive Bayes Methode ist die Bernoulli-Verteilung (Lewis, 1998). Dieses Modell wird auch als Multi-variate Bernoulli Model bzw. Binary Independence Model bezeichnet.[69]

Im Allgemeinen wird bei der Bayes'schen Klassifizierung angenommen, dass die Textinhalte durch ein parametrisierbares Modell generiert wurden. Eine Klasse c bezieht sich dabei auf eine Verteilung mit den Parametern $\theta_c = (\theta_{c1}, ..., \theta_{cn})$. Die Anzahl an Wörtern im Vokabular wird mit $n = |V|$ gekennzeichnet. Basierend auf dieser Annahme werden anhand der Trainingsmenge optimale Werte für die Parameter des Modells geschätzt. Im Rahmen der Text Klassifizierung können theoretisch beliebige diskrete Wahrscheinlichkeitsverteilungen hinsichtlich der verschiedenen Klassen angenom-

[68] Vgl. Mitchell: Machine Learning, S. 177.

[69] Vgl. Lewis: Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval, S. 7 ff.

men werden. Durch Parametrisierung ist es ebenfalls möglich eine Mischverteilung anzunehmen, d.h. für jede Klasse kann eine unterschiedliche Wahrscheinlichkeitsverteilung angenommen werden. Zur Anwendung der multinomialen naive Bayes Methode können die Gleichungen 3.29 und 3.30 verwendet werden. Für die Anwendung des multivariaten Bernoulli-Modells können die Gleichungen 3.31 und 3.32 verwendet werden.[70]

Multinomialverteilung

Beim multinomialen Ereignismodell repräsentiert das Vorkommen eines Wortes ein Ereignis. Die Klassifizierung basiert dabei auf den Worthäufigkeiten innerhalb der Klassen. Wörter, die nicht im zu klassifizierenden Dokument vorhanden sind, werden innerhalb des multinomialen Modells nicht berücksichtigt. Sei M_j das multinomiale Merkmalsvektor für das Dokument d_j. Die multinomiale Wahrscheinlichkeit $P(d_j|c)$ entspricht dabei der Gleichung 3.29. Seien N_c die Anzahl der Wörter bzw. Merkmale innerhalb der Klasse c und N_{ci} die Anzahl des Vorkommens des Wortes w_i innerhalb der Klasse c. Dann kann die Wahrscheinlichkeit $P(w_i|c)$ durch den Parameter $\hat{\theta}_{ci}$ entsprechend der Gleichung 3.30 geschätzt werden.[71]

Ein Wort, das nicht im Vokabular der Trainingsmenge vorkommt, kann während der Klassifizierung dazu führen, dass die geschätzte Wahrscheinlichkeit 0 ergibt, sodass die gesamte A-posteriori-Wahrscheinlichkeit ebenfalls 0 ergibt. Als Lösung für das Problem, wird oft das Laplace- bzw. Add-1-Smoothing angewendet. In der Gleichung 3.30 wird das Laplace-Smoothing angewendet. Weitere anspruchsvollere Smoothing Verfahren werden in Chen/ Goodman (1996) detaillierter untersucht.[72,73,74]

[70] Vgl. McCallum, Andrew/Nigam, Kamal: A comparison of event models for Naive Bayes text classification. In AAAI-98 Workshop on Learning for Text Categorization. AAAI Press, 1998, S. 41.

[71] Vgl. a. a. O., S. 41 ff.

[72] Vgl. Carstensen et al.: Computerlinguistik und Sprachtechnologie: Eine Einführung (German Edition), S. 155 f.

[73] Vgl. Manning/Schütze: Foundations of Statistical Natural Language Processing, S. 202 ff.

[74] Vgl. Mitchell: Machine Learning, S. 179.

$$P(d_j|c) = \frac{|d|!}{\prod_i^{|V|} M_{ij}!} \prod_{i=1}^{|V|} P(w_i|c) \propto \prod_{i=1}^{|V|} P(w_i|c) \tag{3.29}$$

$$\hat{\theta}_{ci} = \hat{P}(w_i|c) = \frac{N_{ci} + 1}{N_c + |V|} \tag{3.30}$$

Multivariate Bernoulli-Verteilung

Im multivariaten Bernoulli Ereignismodell repräsentiert ein Dokument einen Ereignis. Für eine Klasse können die einzelnen Dokumente als voneinander unabhängige Bernoulli Experimente verstanden werden. Ein Dokument wird dabei, nicht wie beim multinomialen Modell durch die Worthäufigkeiten, sondern als einen Vektor mit den Werten 0 oder 1. Der Wert weist daraufhin ob ein Wort im Dokument enthalten ist. Für ein Wort w_i, welches mindestens einmal im Dokument enthalten ist, entspricht der Wert des Koeffizienten 1. Im Gegensatz zum multinomialen Modell wird beim multivaten Bernoulli Modell sowohl die Präsenz als auch die Absenz eines Wortes w_i berücksichtigt. Die Okkurrenz eines Wortes w_i innerhalb des Dokumentes d_j wird mit B_{ij} ausgedrückt. Seien $P(w_i|c)$ die Wahrscheinlichkeit für das Vorkommen des Wortes innerhalb der Klasse c und $(1 - P(w_i|c))$ die Wahrscheinlichkeit, dass das Wort w_i nicht innerhalb der Klasse c vorkommt. Dann kann die Wahrscheinlichkeit $P(d|c)$ entsprechend der Gleichung 3.31 definiert werden. Der Parameter $P(w_i|c)$ kann als der Anteil der Dokumente innerhalb einer Klasse, welche das Wort w_i enthalten, gemäß der Gleichung 3.32 geschätzt werden. An dieser Stelle kann, ähnlich wie im multinomialen Fall, das Laplace-Smoothing angewendet werden. N_{ci} entspricht dabei der Anzahl der Dokumente, in denen das Wort w_i beobachtet wurde, und N_c für die gesamte Anzahl der Dokumente innerhalb der Klasse c.[75]

$$P(d_j|c) = \prod_{i=1}^{|V|} (B_{ij}\, P(w_i|c) + (1 - B_{ij})\,(1 - P(w_i|c))) \tag{3.31}$$

[75] Vgl. McCallum/Nigam: A comparison of event models for Naive Bayes text classification, S. 42.

$$\hat{\theta}_{ci} = \hat{P}(w_i|c) = \frac{N_{ci} + 1}{N_c + 2} \tag{3.32}$$

3.2.4 Support Vector Machines

Initial wurden Support Vector Machines (SVM) von Cortes/Vapnik (1995) vorgestellt. Für die Beschreibung der Funktionsweise der SVMs wird eine binäre Klassifizierung betrachtet. Desweiteren wird für die folgenden Erläuterungen angenommen, dass die Trainingsmenge linear separierbar ist. Gegeben ist eine Trainingsmenge $L = \{x_i, y_i\}$ mit l Exemplaren. Ein Exemplar $x_i \in \mathbb{R}^d$ besteht dabei aus d Merkmalen und einer zugeordneten Klasse $y_i \in \{-1, 1\}$. Mit der Gleichung 3.33 werden alle möglichen Hyperebenen im $\mathbb{R}^d$ Raum ausgedrückt. Eine Hyperebene wird dabei durch einen Normalenvektor w und einer Konstante b beschrieben.[76]

$$w \cdot x + b = 0 \tag{3.33}$$

Wenn eine Hyperebene entsprechend der Gleichung 3.33 gegeben ist, kann ein beliebiges Exemplar anhand der Funktion $h(x)$ 3.34 zu einer Klasse aus einer vordefinierten Menge $Y \in \{-1, 1\}$ zugeordnet werden.[77]

$$h(x) = \text{sgn}(w \cdot x + b) \tag{3.34}$$

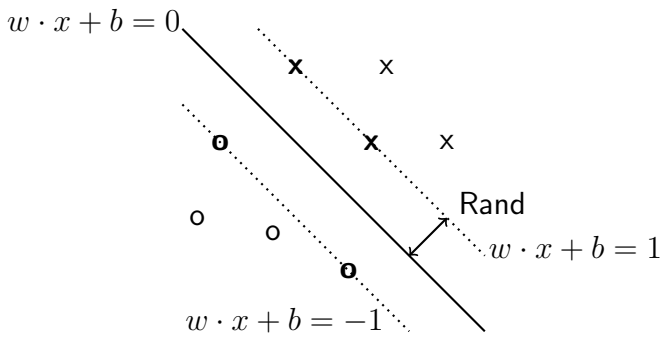
Es werden dabei die Hyperebenen betrachtet, welche die Bedingungen 3.35 erfüllen. Die Erfüllung der Bedingungen bedeutet für eine Hyperebene, dass diese x_i korrekt klassifiziert hat.[78]

$$x_i \cdot w + b \geq +1 \text{ für } y_i = +1 \tag{3.35}$$
$$x_i \cdot w + b \leq -1 \text{ für } y_i = -1 \tag{3.36}$$
$$y_i \cdot (x_i \cdot w + b) \geq 1, \forall i \tag{3.37}$$

[76] Vgl. Cortes, Corinna/Vapnik, Vladimir: Support-Vector Networks. Mach. Learn. 20 September 1995, Nr. 3 ⟨URL: http://dx.doi.org/10.1023/A:1022627411411⟩, ISSN 0885–6125, S. 277 f.

[77] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 68.

[78] Vgl. Cortes/Vapnik: Mach. Learn. 20 [1995], S. 277 f.



In Anlehnung an: Mohri/Rostamizadeh/Talwalkar (2012), S. 65

Abbildung 3.4: Kanonische Hyperebene und Rand-Hyperebenen

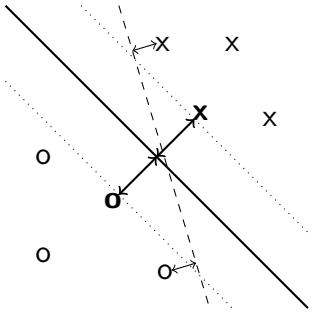
Die Hyperebene kann auch gleichermaßen durch alle Wertepaare $\{\lambda w, \lambda b\}$ für $\lambda \in \mathbb{R}^+$ ausgedrückt werden. Die Problematik dabei ist, dass nicht alle Wertepaare zur gleichen Distanz von der Hyperebene zu einem Exemplar x_i führt. Um eine geometrische Distanz zwischen der Hyperebene und einem Exemplar $x_i, i \in [1, l]$ zu erhalten muss die Distanz mit $\|w\|$ normalisiert werden. Die Abbildung 3.4 visualisiert die kanonische Hyperebene zur Trennung der Exemplare in die entsprechenden zwei Klassen. Zusätzlich sind die kanonischen Rand-Hyperebenen $w \cdot x + b = \pm 1$ abgebildet, welche parallel zur kanonischen Hyperebene sind. Die Distanz zwischen der Hyperebene und den nächsten Exemplaren wird als Rand (engl. margin) bezeichnet. Der Rand r wird durch die Gleichung 3.38 gegeben.[79,80]

$$r = \underset{(x,y)\in L}{min} \frac{|w \cdot x + b|}{\|w\|} = \frac{1}{\|w\|} \tag{3.38}$$

Für eine möglichst gute Verallgemeinerung anhand der Trainingsmenge soll der Rand maximiert werden. Die Maximierung des Randes r ist äquivalent zur Minimierung von $\|w\|$ bzw. $\frac{1}{2} \cdot \|w\|^2$. Dies kann durch die Anwendung der Langrange-Multiplikatoren erreicht werden. Schlussendlich kann das Problem entsprechend der Gleichung 3.39 nach Cortes/Vapnik (1995, S. 291 ff) und Burges (1998, S. 7 ff) formuliert werden. Die Abbildung 3.5 illustriert die Wahl einer Hyperebene, welche den Rand zu den nächsten Punkten zur Hy-

[79] Vgl. Cortes/Vapnik: Mach. Learn. 20 [1995], S. 278 f.

[80] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 65.



In Anlehnung an: Aggarwal/Zhai (2012), S. 194

Abbildung 3.5: Selektion einer optimalen Hyperebene

perebene maximiert.[81,82]

$$\begin{aligned} &\text{Minimiere} && W(\alpha) = -\sum_{i=1}^{l} \alpha_i + \frac{1}{2} \cdot \sum_{i=1}^{l} \sum_{j=1}^{l} y_i \, y_j \, \alpha_i \, \alpha_j \, (x_i \cdot x_j) \\ &\text{unter den NB} && \sum_{i=1}^{l} y_i \, \alpha_i = 0 \\ & && 0 \leq \alpha_i \leq C, \forall i \in [1, l]. \end{aligned} \tag{3.39}$$

Mit der quadratischen Programmierung kann α mit der gegebenen Matrix $y_i \, y_j (x_i \cdot x_j)$ berechnet werden. Die Lagrange Multiplikatoren α_i determinieren die optimale Hyperebene zur Separierung der Daten hinsichtlich der Klassen. Die Koeffizienten von α betragen nur für die Stützvektoren (engl. support vector) einen Wert größer als 0. Der Normalenvektor w kann gemäß der Gleichung 3.40 als eine Linearkombination der Trainingselemente betrachtet werden. Da die Stützvektoren auf den Rand-Hyperebenen liegen, kann die Konstante b mit Hilfe eines beliebigen Stützvektors gemäß der Gleichung 3.41 berechnet werden.[83]

$$w = \sum_{i}^{l} \alpha_i \, y_i \, x_i \tag{3.40}$$

[81] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 65.

[82] Vgl. a. a. O., S. 68.

[83] Vgl. a. a. O.

$$b = y_i - \sum_{j=1}^{l} \alpha_j \, y_j \, (x_j \cdot x_i) \tag{3.41}$$

Mit dem Normalenvektor w und der Konstante b kann die optimale Hypothese bzw. Hyperebene durch das Einsetzen von w und b in der Gleichung 3.34 determiniert werden.[84]

Es kann Szenarien geben, in denen eine Trainingsmenge störende bzw. unreine Exemplare enthält, sodass die lineare Separierbarkeit der Trainingsmange nicht gegeben ist. Für diese Fälle gibt es zwei mögliche Ansätze. Die eine Möglichkeit ist das Erlauben von Fehlern. Dazu kann die Konstante C in der Gleichung 3.39 verwendet werden. Für $C = \infty$ wird eine Hyperebene gesucht, welche die Trainingsdaten vollständig linear separiert. Je kleiner der Wert C ist, desto mehr werden Fehler in der Klassifizierung erlaubt. In diesem Zusammenhang wird auch oft von einem weichen Rand (soft margin) gesprochen. Der Wert für den Parameter C kann auch durch die Anwendung einer Kreuzvalidierung ermittelt werden. In Shawe-Taylor/Cristianini (2004, S. 219 ff) wird eine formale Herleitung des Parameters C gegeben.[85]

3.2.5 Kernel Methoden

Bei der Verwendung einer SVM wird angenommen, dass die Trainingsdaten linear separierbar vorliegen. Falls die Trainingsdaten jedoch nicht linear separierbar sind, können Kernel-Funktionen verwendet werden. Im Allgemeinen ist ein Kernel eine Funktion K zur Berechnung des Skalarprodukts für alle $x, x' \in X$ im Merkmalsraum M entsprechend der Gleichung 3.42. Mit der Funktion ϕ werden x und x' in den Merkmalsraum M abgebildet.[86]

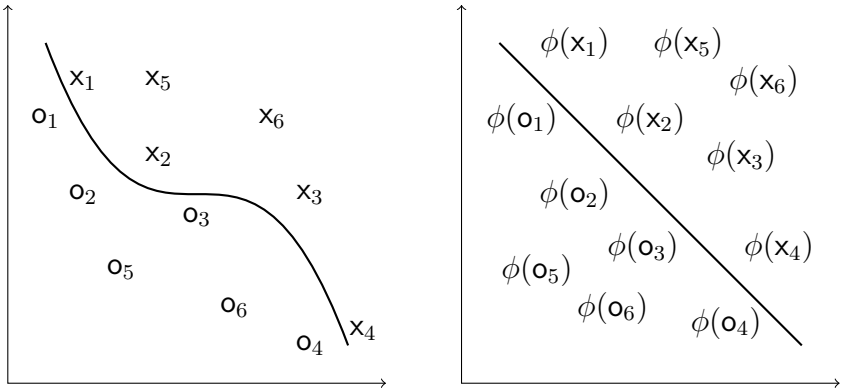
$$\forall x, x' \in X, K(x, x') = \langle \phi(x), \phi(x') \rangle \tag{3.42}$$

In Shawe-Taylor/Cristianini (2004) wird eine Kernel Methode in zweierlei Hinsicht definiert. Demnach besteht eine Kernel Methode aus einer Kernel-Funktion und einem Lernalgorithmus, wie zum Beispiel Support Vector Machines, zur linearen Separierung der Vektoren im Merkmalsraum M. Die

[84] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 68.

[85] Vgl. Shawe-Taylor/Cristianini: Kernel Methods for Pattern Analysis, S. 224 f.

[86] Vgl. a. a. O., S. 34.



In Anlehnung an: Shawe-Taylor/Cristianini (2004), S. 27

Abbildung 3.6: Idee der Abbildung von Daten in einen neuen Raum

Abbildung 3.6 illustriert die Idee hinter der Abbildung von Daten in einen neuen Merkmalsraum M durch die Funktion ϕ.[87]

Statt der Anwendung der quadratischen Programmierung auf der Matrix $y_i\, y_j(x_i \cdot x_j)$, wird die Matrix $y_i\, y_j(K(x_i, x_j))$ verwendet. Da die quadratische Programmierung die optimalen Werte α bezogen auf den Merkmalsraum M liefert, muss die Kernel-Funktion zur Berechnung des Normalenvektors w und der Konstante b berücksichtigt werden. Die Konstante b kann gemäß der Gleichung 3.43 für einen beliebigen Stützvektor x_i bestimmt werden. Somit ergibt sich für die Funktion des Klassifikators die Gleichung 3.44.[88,89]

$$b = y_i - \sum_{j=1}^{l} \alpha_j\, y_j\, K(x_j, x_i) \tag{3.43}$$

$$h(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i\, y_i\, K(x_i, x) + b\right) \tag{3.44}$$

[87] Vgl. Shawe-Taylor/Cristianini: Kernel Methods for Pattern Analysis, S. 25.

[88] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 100.

[89] Vgl. Siolas, Georges/Buc, Florence d'Alché: Support Vector Machines Based on a Semantic Kernel for Text Categorization. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5 - Volume 5. Washington, DC, USA: IEEE Computer Society, 2000, IJCNN '00 ⟨URL: http://dl.acm.org/citation.cfm?id=846232.849038⟩, ISBN 0–7695–0619–4, S. 207.

Tabelle 3.5: Symmetrische und positiv definite Kernel-Funktionen

Bezeichnung	Definition
Polynomiale Kernel-Funktion	Für eine beliebige Konstante $c > 0$ definiert sich eine polynomiale Kernel-Funktion des Grades $d \in \mathbb{N}$ mit $$\forall x, x' \in \mathbb{R}^N, K(x, x') = (x \cdot x' + c)^d.$$
Gaußsche Kernel-Funktion	Eine Gaußsche Kernel-Funktion definiert sich für beliebige Werte $\sigma > 0$ mit $$\forall x, x' \in \mathbb{R}^N, K(x, x') = \exp\left(-\frac{\|x' - x\|^2}{2\,\sigma^2}\right).$$
Sigmoidale Kernel-Funktion	Für beliebige relle Konstanten $a, b \geq 0$ definiert sich eine sigmoidale Kernel-Funktion mit $$\forall x, x' \in \mathbb{R}^N, K(x, x') = \tanh(a(x \cdot x') + b).$$

In Anlehnung an: Mohri/Rostamizadeh/Talwalkar (2012), S. 92 ff.

Für die Berechnung des Skalarprodukts im Merkmalsraum M durch die Kernel-Funktion K wird unter Berücksichtigung des Theorems von Mercer keine explizite Definition der Abbildungsfunktion ϕ benötigt. Wenn eine Kernel-Funktion die Bedingung von Mercer (Mercer, 1909) erfüllt, wird die Existenz einer Abbildung ϕ der Daten in den Merkmalsraum M garantiert. In diesem Fall kann das Skalarprodukt mit der Kernel-Funktion K ohne die Abbildung ϕ implizit berechnet werden. Dies wird in der Literatur als Kernel-Trick bezeichnet. Die Tabelle 3.5 zeigt einige Kernel-Funktionen, welche die Bedingung von Mercer erfüllen.[90]

[90] Vgl. Mohri/Rostamizadeh/Talwalkar: Foundations of Machine Learning (Adaptive Computation and Machine Learning series), S. 91.

Tabelle 3.6: Enthaltene Kategorien im Twenty Newsgroups Korpus

1	rec.sport.baseball	8	talk.religion.misc	15	talk.politics.misc
2	comp.graphics	9	alt.atheism	16	comp.os.ms-windows.misc
3	sci.electronics	10	rec.motorcycles	17	rec.autos
4	comp.windows.x	11	comp.sys.ibm.pc.hardware	18	comp.sys.mac.hardware
5	rec.sport.hockey	12	talk.politics.guns	19	soc.religion.christian
6	sci.crypt	13	sci.med	20	talk.politics.mideast
7	sci.space	14	misc.forsale		

In Anlehnung an: Mitchell (1997), S. 184

3.3 Evaluation

Unterschiedliche Lernalgorithmen wurden im Kontext der Text Klassifizierung bereits von Forschern evaluiert. Zur Evaluation werden Standard Korpora, wie zum Beispiel *Reuters-21578* und *Twenty Newsgroups*, für die Forschung zur Verfügung gestellt. Zur Demonstration der beschriebenen Methoden und Lernalgorithmen wird der Korpus *Twenty Newsgroups*[91] verwendet. Der Korpus ist kategorisiert aufgebaut und eignet sich hinsichtlich der Struktur für die Evaluation. Je eine Klasse existiert ein Ordner mit Textdateien. Insgesamt beinhaltet der Korpus 19 Kategorien mit jeweils 1000 Textdokumenten und einer Kategorie mit 997 Textdokumente statt 1000. Damit stehen insgesamt 19997 Textdokumente für die Evaluation zur Verfügung. Die Tabelle 3.6 listet die enthaltenen Kategorien auf.[92]

Zur Anwendung der Lernalgorithmen wird die Python Bibliothek *Scikit-learn* verwendet. Diese Bibliothek implementiert eine Vielzahl von Lernalgorithmen sowohl für überwachte als auch unüberwachte Lernprobleme. Zur Merkmalsgewichtung ist nur die Implementierung der inversen Dokumenthäufigkeit (Kapitel 3.1.2) verfügbar. Desweiteren sind Implementierungen der Lernalgorithmen Naive Bayes und SVM vorhanden. Die in diesem Kapitel dargestellten Listings bzw. Quellcodes lehnen sich an Beispiele aus der Scikit-learn Bibliothek.[93]

[91] Vgl. Mitchell, Tom: Twenty Newsgroups Data Set. ⟨URL: http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups⟩ – Zugriff am 2014-01-19.

[92] Vgl. Mitchell: Machine Learning, S. 182 f.

[93] Vgl. Pedregosa, F. et al.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12 2011, S. 2826 ff.

Listing 3.1: Idealtypischer Aufbau eines Textdokuments aus dem Twenty Newsgroups Korpus

```
Newsgroups: <newsgroups>
Path: <path>
From: <email>
Subject: <subject>
Message-ID: <message-id>
Sender: <sender>
Organization: <organization>
Date: <date>
Lines: <amount of lines>

<content>
```

Listing 3.2: Bereinigung der Header Informationen in den Textdokumenten

```
import os
import sys
path = sys.argv[1]
separator = '\n\n'
folders = os.listdir(path)
for folder in folders:
    docs = os.listdir(path + '/' + folder)
    for doc in docs:
        fp = path + '/' + folder + '/' + doc
        f = open(fp, 'r')
        raw = f.read()
        f.close()
        f = open(fp, 'w')
        f.write(raw.split(separator,1)[1])
        f.close()
```

Vor der Evaluation sollte die Struktur und der Inhalt des *Twenty Newsgroups* Korpus vorbereitet werden. Der idealtypische Aufbau eines Textdokuments wird in Listing 3.1 dargestellt. Die Textdokumente fangen mit einem Header an, welcher unter anderem Informationen zur Newsgroup, zum Thema und zur Absenderadresse enthält. Für die Klassifizierung werden die Header aus allen Textdokumenten mit dem Python Skript aus Listing 3.2 entfernt.[94]

Das Listing 3.3 zeigt eine mögliche Lösung für das Auslesen der Textinhalte. Die Methoden lehnen sich stark an die Struktur des Twenty Newsgroups Korpus an. Die Klasse `CorpusReader` kann für Korpora mit einer Ordnerstruktur, welches dem Twenty Newsgroups Korpus ähnelt, verwen-

[94] Vgl. Wang/Zhang: J. Inf. Sci. Eng. 29 [2013], S. 216.

det werden. Nach dem Auslesen müssen die Textinhalte in eine Dokument-Wort Matrix transformiert werden. Die Dokument-Wort Matrix entspricht der transponierten Wort-Dokument Matrix aus dem Kapitel 2.1.4 auf Seite 12. Die Scikit-learn Bibliothek bietet dazu die Klasse `CountVectorizer` an. Für eine gegebene Instanz der `CountVectorizer` Klasse und dem Pfad zum Korpus, kann die Transformation mit dem Aufruf der Methode `extr` erfolgen. Das Ergebnis der Transformation resultiert in den Variablen `X`, `y` und `y_names`.

Variable `X`: Diese Variable beinhaltet die Dokument-Wort Matrix. Das Format entspricht einer *Compressed Sparse Column* Matrix. Mit dieser Datenstruktur kann eine dünnbesetzte Matrix in komprimierter Form dargestellt werden. Dazu wird in Scikit-learn die Bibliothek *SciPy*[95] verwendet. Wenn beispielsweise ein Wort w_i nur im Dokument d_j vorkommt, entsprechen die Koeffizienten der Dokumente $d_k, k \neq j$ in der Spalte i dem Wert Null. In einem solchen Fall eignet sich die Verwendung von Methoden zur Speicherung von dünnbesetzten Matrizen. Die *Compressed Sparse Row Matrix* ist beispielsweise eine weitere mögliche Datenstruktur für den Umgang mit dünnbesetzten Matrizen.[96]

Variable `y`: Diese Variable enthält korrespondierenden Klassen der Dokumente in `X`. Mit dem Index j für ein Dokument d_j in der Matrix `X` kann die korrespondierende Klasse aus dem Array `y` enthalten werden. Die Anzahl der Dokumente bzw. Zeilen der Matrix `X` entspricht der Länge des Arrays `y`. Statt den tatsächlichen Bezeichnungen der Klassen werden Zahlen gespeichert. Anhand einer Zahl kann die zugehörige Bezeichnung der Klasse erhalten werden.

Variable `y_names`: In diesem Array werden die Bezeichnungen der Klassen gespeichert. Eine Zahl aus dem Array `y` kann als Index verwendet werden, um die dazugehörige Bezeichnung der Klasse in `y_names` nachzuschlagen.

Die Verarbeitungsschritte einer `CountVectorizer` Instanz können über Parameter eingestellt werden. Im Rahmen der Evaluation wird lediglich die Liste der Stoppwörter spezifiziert. Dazu wird die bereits in der Scikit-learn

[95] Vgl. Jones, Eric et al.: SciPy: Open source scientific tools for Python. ⟨URL: http://www.scipy.org⟩ – Zugriff am 2014-01-19.

[96] Vgl. Davis, Timothy A.: Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms). Society for Industrial and Applied Mathematic, 9 2006, 217 Seiten, ISBN 9780898716139, S. 7 ff.

Listing 3.3: Auslesen der Textinhalte aus dem Twenty Newsgroups Korpus

```
import os

CATEGORY = 'category'
DOCS = 'documents'

class CorpusReader:
    def read_file(self, fname):
        return unicode(open(fname).read(), errors='ignore')

    def read_files(self, path):
        os.chdir(path)
        return [self.read_file(fname)
                for fname in os.listdir('.')]

    def read(self, path):
        os.chdir(path)
        return [{CATEGORY : c,
                 DOCS : self.read_files(path + c)}
                for c in os.listdir('.')]

    def categories(self, corpus):
        categories = {}
        y_names = []
        i = 0
        for c in corpus:
            categories[c[CATEGORY]] = i
            y_names.append(c[CATEGORY])
            i += 1
        return categories, y_names

    def extr(self, vectorizer, path):
        corpus = self.read(path)
        categories, y_names = self.categories(corpus)
        documents = self.documents(corpus)
        X = vectorizer.fit_transform(documents)
        y = [y for c in corpus for y
             in [categories[c[CATEGORY]]] * len(c[DOCS])]
        return X, y, y_names
```

Bibliothek vorhandene Liste von Stoppwörtern verwendet. Im Quellcode der Scikit-learn Bibliothek wird als Quelle für die Liste der Stoppwörter die Glasgow Information Retrieval Group angegeben.[97] Für alle weiteren Parameter werden die Standard Einstellungen verwendet. Zum Standard gehörend werden alle Wörter kleingeschrieben. Für die Tokenisierung verwendet Scikit-learn standardmäßig den regulären Ausdruck `\b\w\w+\b`. Ein Vokabular wird nicht angegeben. Stattdessen wird das Vokabular durch den `CountVectorizer` anhand des Korpus automatisiert aufgebaut. Das Listing 3.4 zeigt die beispielhafte Durchführung der Evaluation eines Klassifikators mit der Bibliothek Scikit-learn. Der Quellcode lehnt sich dabei an das Beispiel der Scikit-learn Bibliothek an.[98] Zu Anfang werden die Textinhalte mithilfe der Klassen `CorpusReader` und `CountVectorizer` für die Evaluation vorbereitet. Anschließend wird die Menge der Daten in eine Trainings- und Testmenge zerlegt. 40 Prozent der Datenmenge entspricht der Testmenge und die anderen 60 Prozent der Trainingsmenge. Danach wird ein Klassifikator trainiert, mit dem die Textdokumente aus der Testmenge klassifiziert werden. Das Ergebnis der Klassifikation entspricht einem Array mit den vorausgesagten Klassen. Mit den vorausgesagten `y_pred` und den tatsächlichen Klassen `y_test` können abschließend Metriken zur Messung der Performanz ermittelt werden.

Die Tabelle 3.7 zeigt die Performanz einer Auswahl von Lernalgorithmen in Abhängigkeit zur Vorverarbeitung der Merkmale. Für die Evaluation wurden die F_1-Makro Metrik ausgewertet. In allen Evaluationsdurchläufen wurden die Dokument-Wort Matrizen mit der euklidischen Norm normalisiert. Die kNN Methode erreicht in der Gesamtbetrachtung die schlechteste Performanz. Es wurde eine Kreuzvalidierung mit 3 Partitionen mit der kNN Methode durchgeführt, um den optimalen Parameter k zu finden. Dazu wurden die Werte 1 bis 10 ausprobiert. Die beste Performanz wurde mit $k = 1$ erzielt. Die Ergebnisse der Bernoulli Variante der Naive Bayes Methode können durch das zugrunde liegende Ereignismodell erklärt werden. Durch das Bernoulli Ereignismodell reagiert die Methode reagiert lediglich auf die Präsenz bzw. Absenz von Wörtern. Demnach hat eine Merkmalsgewichtung für diese Methode keine Auswirkung. Das Entfernen der Stoppwörter aus der Merkmalsmenge steigerte in diesem Experiment die Performanz der Naive Bayes und SVM Methode. Für die Naive Bayes Methode wurde Laplace Smoothing

[97] Vgl. Group, Glasgow Information Retrieval: Stop word list. ⟨URL: http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words⟩ – Zugriff am 2014-01-19.

[98] Vgl. Prettenhofer, Peter et al.: Classification of text documents using sparse features. ⟨URL: http://scikit-learn.org/stable/auto_examples/document_classification_20newsgroups.html⟩ – Zugriff am 2014-01-20.

Listing 3.4: Evaluation von Klassifikatoren am Beispiel des multinomialen Naive Bayes

```
import sys
import numpy as np
from evaluation import CorpusReader
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction import stop_words
from sklearn.naive_bayes import MultinomialNB
from sklearn import cross_validation as cv
from sklearn.metrics import *

path = sys.argv[1]
v = CountVectorizer(stop_words=stop_words.ENGLISH_STOP_WORDS)
X, y, y_names = CorpusReader().extr(v, path)
X = preprocessing.normalize(X.astype(np.double), norm='l2')

X_train, X_test, y_train, y_test = cv.train_test_split(X, y,
    test_size=0.4)
clf = MultinomialNB()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
precision = precision_score(y_test,y_pred)
recall = recall_score(y_test,y_pred)
f1_micro = f1_score(y_test, y_pred, average='micro')
f1_macro = f1_score(y_test, y_pred, average='macro')
```

angewendet. Der Parameter C für die SVM Methode wurde bei dem Standard Wert von 1 belassen. Die beste Performanz erzielte die SVM Methode.

Eine weitere Möglichkeit zur Betrachtung der Richtigkeit von Klassifikatoren sind Konfusionsmatrizen. Die Spalten stellen die Entscheidung eines Klassifikators für eine Klasse dar. Die Zeilen stellen die tatsächlich korrekte Klasse dar. Die Anzahl der Zeilen und Spalten entsprechen der Anzahl der Klassen. Die Reihenfolge der Klassen entlang der Zeilen sowie Spalten sind gleich. In der Hauptdiagonalen kann beobachtet werden, für wie viele Textdokumente ein Klassifikator die tatsächlich korrekte Klasse vorausgesagt hat. Die Konfusionsmatrizen in der Abbildung 3.7 basieren auf den Ergebnissen der zuvor beschriebenen Evaluation mit Entfernung der Stoppwörter und der Anwendung der tf.idf Merkmalsgewichtung. Die Konfusionsmatrizen wurden auf Grundlage des Beispiels der Scikit-learn Bibliothek generiert.[99] Die Zuord-

[99] Vgl. Scikit-learn: Confusion matrix. ⟨URL: http://scikit-learn.org/

Tabelle 3.7: Auswertung der F_1-Makro Metrik für eine Auswahl von Klassifikatoren anhand des Twenty Newsgroups Korpus

	1NN	Naive Bayes (Bernoulli)	Naive Bayes (multinomial)	SVM (linear)
nur Worthäufigkeiten	53,51 %	66,17 %	70,93 %	76,57 %
Stoppwörter	44,21 %	68,12 %	79,35 %	82,11 %
tf.idf	44,67 %	66,17 %	79,32 %	84,59 %
Stoppwörter und tf.idf	41,60 %	68,12 %	82,31 %	84,94 %

nung der Indizes zu den Klassen kann in der Tabelle 3.6 gefunden werden.[100]

3.4 Beurteilung

Die durchgeführte Evaluation beschränkt sich stark auf ein spezielles Korpus und eine beschränkten Menge an Lernalgorithmen. Es kann keine eindeutige Entscheidung auf Basis der Evaluation getroffen, dass ein bestimmter Lernalgorithmus stets besser ist. Die Performanz sowie das Verhalten der Lernalgorithmen kann sich möglicherweise je nach Korpus ändern. Der Mehrwert der Evaluation liegt in der Demonstration der zuvor beschriebenen Methoden in der Umsetzung. Dies soll helfen, das Verständnis sowie die Intuition hinsichtlich der Lernalgorithmen zu steigern.

Die Auswahl eines Klassifikators mit der besten Performanz wird möglicherweise nicht stets erwartete Klassifizierungen treffen. Daher ist es wichtig zu wissen, welche möglichen Anpassungen getroffen werden können um die Performanz eines Klassifikators zu verbessern. Beispielsweise könnten weitere Merkmale hinzugefügt werden. Eine andere Möglichkeit wäre durch die Methoden der Merkmalsselektion die Merkmalsmenge zu reduzieren. Eine weitere Möglichkeit wäre weitere Trainingsexemplare zu sammeln. Eine Empfehlung in diesem Zusammenhang ist es, den Bias-Varianz Trade-off zu analysieren.Ein hoher Bias liegt vor, wenn die Merkmale unzureichend waren. In

stable/auto_examples/plot_confusion_matrix.html⟩ – Zugriff am 2014-01-20.

[100] Vgl. Manning/Schütze: Foundations of Statistical Natural Language Processing, S. 374 f.

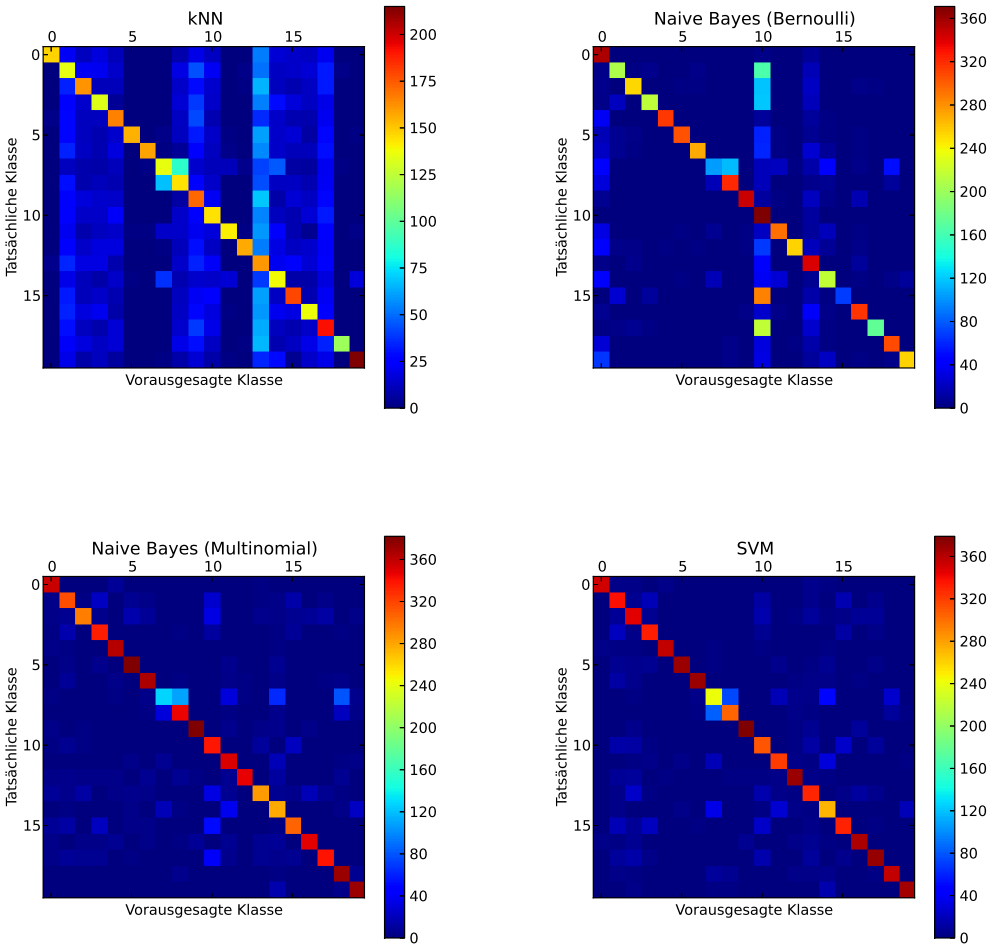
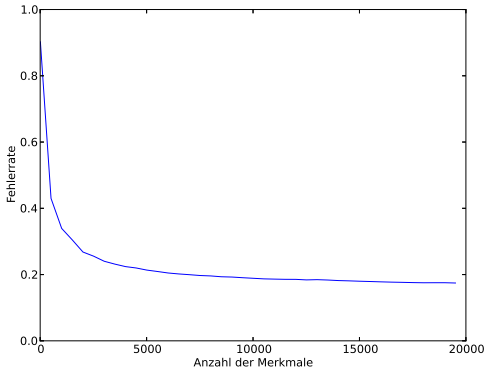


Abbildung 3.7: Konfusionsmatrizen bezüglich der Evaluation der Klassifikatoren



In Anlehnung an Pekalska/Duin (2005), S. 158

Abbildung 3.8: Verhältnis zwischen der Fehlerrate und der Anzahl der Merkmale

diesem Fall hilft es nicht die Trainingsmenge durch neue Exemplare anzureichern. Ein hoher Bias kann beobachtet werden, wenn mit dem Hinzufügen neuer Trainingsexemplare die Fehlerrate bzw. der Anteil der falschen Klassifizierungen nicht weiter sinkt. Die Abbildung 3.8 veranschaulicht das Verhältnis zwischen der Fehlerrate und der Anzahl der Merkmale. Die Fehlerrate entspricht dem Anteil der falsch klassifizierten Exemplare. Mit zunehmender Anzahl an Merkmal verringert sich die Fehlerrate immer weniger. Dieses Experiment wurde mit dem multinomialen Naive Bayes Lernalgorithmus durchgeführt. Zur Behebung eines hohen Bias können neue Merkmale in Betracht gezogen werden, welche zu einer besseren Performanz des Klassifikators führen könnten. Im Falle einer hohen Varianz jedoch sollten weitere Trainingsexemplare gesammelt werden, um eine breiteren Menge an Exemplaren zu berücksichtigen. Dadurch kann möglicherweise eine bessere Generalisierung erreicht werden. Die Quintessenz liegt in der Durchführung einer Diagnose und der gezielten Anwendung von Maßnahmen statt zufälliger Maßnahmen. Dadurch kann die Performanz eines Klassifikators gezielt optimiert werden. Weitere Details zu diesem Thema können in Hastie/Tibshirani/Friedman (2009) und Pekalska/Duin (2005) gefunden werden.[101,102]

[101] Vgl. Hastie/Tibshirani/Friedman: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, S. 37 f.

[102] Vgl. Pekalska, Elzbieta/Duin, Robert P. W.: The Dissimilarity Representation for Pattern Recognition: Foundations And Applications (Machine Perception and Artificial Intelligence). World Scientific Pub Co Inc, 12 2005, 636 Seiten, ISBN 9789812565303, S. 157 ff.

Eine Empfehlung für die Durchführung einer Text Klassifizierung ist, mit einer sehr einfachen Lösung zu beginnen. Der intiale Klassifikator kann beispielsweise für den Anfang ohne jegliche Merkmalsselektion oder -gewichtung trainiert werden. Der resultierende Klassifikator kann unter Verwendung von Performanzmaße analysiert werden. Anschließend können weitere Methoden der Merkmalsverarbeitung getestet werden. Die Anwendung verschiedener Methoden zur Vorverarbeitung kann zeitintensiv sein. Bei qualitativ hochwertigen Kopora wird möglicherweise keine bzw. nur wenige Vorverarbeitung benötigt. Dadurch kann die Geschwindigkeit der Klassifizierung gesteigert werden.

Kapitel 4

Schlussbetrachtung

4.1 Fazit

In dieser Arbeit wurden verschiedene Methoden entlang der idealtypischer Phasen der automatisierten Text Klassifizierung dargestellt. Zur Merkmalsverarbeitung wurden Verfahren der Selektion, Gewichtung und Transformation von Merkmalen beschrieben. Anschließend wurden verschiedene Lernalgorithmen präsentiert. Die Gemeinsamkeit der Lernalgorithmen liegt dabei in der automatisierten Generierung von Modellen bzw. Klassifikatoren. Mit einem Klassifikator lassen sich zuvor unbekannte Dokumente klassifizieren. Im Rahmen der Evaluation wurden die vorgestellten Methoden in der Anwendung demonstriert. Zur Demonstration wurden Standard Korpora verwendet, welche üblicherweise in der Forschung verwendet werden, um die Vergleichbarkeit der Ergebnisse sicherzustellen. Mit der Beurteilung wurden die Methoden, durch die Erkenntnisse aus der Evaluation, erneut durchleuchtet. Zudem wurden Empfehlungen hinsichtlich der Anwendung der Methoden gegeben.

4.2 Ausblick

Ein relevantes und weiterführendes Thema bezüglich der Text Klassifizierung ist das Anwenden von Lernalgorithmen in verteilten Systemen (Silva/ Ribeiro, 2009). In Silva/Ribeiro (2009) wird diese Thematik detaillierter betrachtet. Durch die Anwendung einer verteilten Text Klassifizierung soll dabei die Geschwindigkeit der Klassifizierung gesteigert werden. Die Autoren

gehen dabei detaillierter auf das Task Scheduling mit gerichteten azyklischen Graphen ein. Zunächst wird ein Entwurf über eine verteilte Text Klassifizierung vorgestellt. Abschließend wird der vorgeschlagene Entwurf der verteilten Text Klassifizierung anhand der Korpora Reuters-21578 und Reuters-RCV1 getestet. Eine Übersicht über verschiedene Methoden für die verteilte Anwendung von Lernalgorithmen wird in Peteiro-Barral/Guijarro-Berdiñas (2013) gegeben.[1]

Ein weiteres Thema ist die Implementierung der im Rahmen dieser Arbeit beschriebenen Methoden zur Merkmalsverarbeitung für die Python Bibliothek Scikit-learn. Für die Evaluierung standen lediglich Implementierungen für die Transformation der Textdokumente in das Vektorraummodell und die tf.idf Merkmalsgewichtung zur Verfügung. Es könnten Routinen zur Ermittlung der benötigten Statistiken für die Merkmalsgewichtung implementiert werden. Dazu zählt beispielsweise die Ermittlung der Kategoriehäufigkeiten der einzelnen Wörter im Korpus. Diese Werte können anschließend abgespeichert werden, damit diese später während der Gewichtung der Wörter verwendet werden können.[2]

Die Kombination verschiedener Lernalgorithmen ist durch Meta-Algorithmen möglich. In Aggarwal/Zhai (2012) wird eine Übersicht über verschiedene Meta-Algorithmen, wie zum Beispiel AdaBoost oder Ensemble Learning, gegeben.[3]

[1] Vgl. Silva, C./Ribeiro, B.: Inductive Inference for Large Scale Text Classification: Kernel Approaches and Techniques. Springer, 2009, Studies in Computational Intelligence, ISBN 9783642045325, S. 93 ff.

[2] Vgl. Wang/Zhang: J. Inf. Sci. Eng. 29 [2013], S. 216.

[3] Vgl. Aggarwal/Zhai: Mining Text Data, S. 209 ff.

Literaturverzeichnis

Aggarwal, Charu C./Zhai, ChengXiang (Hrsg.): Mining Text Data. 2012. Auflage. Springer, 2 2012, 535 Seiten, ISBN 9781461432227

Alpaydin, Ethem: Introduction to Machine Learning (Adaptive Computation and Machine Learning series). second edition Auflage. The MIT Press, 12 2009, 584 Seiten, ISBN 9780262012430

Berry, Michael W./Dumais, Susan T./O'Brien, Gavin W.: Using Linear Algebra for Intelligent Information Retrieval. SIAM Rev. 37 Dezember 1995, Nr. 4, 573–595 ⟨URL: http://dx.doi.org/10.1137/1037127⟩, ISSN 0036–1445

Beyer, Kevin S. et al.: When Is "Nearest Neighbor" Meaningful? In Proceedings of the 7th International Conference on Database Theory. London, UK, UK: Springer-Verlag, 1999, ICDT '99 ⟨URL: http://dl.acm.org/citation.cfm?id=645503.656271⟩, ISBN 3–540–65452–6, 217–235

Bhatia, Nitin/Vandana: Survey of Nearest Neighbor Techniques. CoRR, abs/1007.0085 2010, 302–305

Breiman, Leo et al.: Classification and Regression Trees. 1. Auflage. Chapman and Hall/CRC, 1 1984, 368 Seiten, ISBN 9780412048418

Burges, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. Data Min. Knowl. Discov. 2 Juni 1998, Nr. 2, 121–167 ⟨URL: http://dx.doi.org/10.1023/A:1009715923555⟩, ISSN 1384–5810

Carstensen, Kai-Uwe et al. (Hrsg.): Computerlinguistik und Sprachtechnologie: Eine Einführung (German Edition). 3. Auflage. Spektrum Akademischer Verlag, 11 2009, 736 Seiten, ISBN 9783827420237

Cavnar, William B./Trenkle, John M.: N-Gram-Based Text Categorization. In In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. Las Vegas, US, 1994, 161–175

Cha, Sung-Hyuk: Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. International Journal of Mathematical Models and Methods in Applied Sciences, 1 2007, Nr. 4, 300–307 ⟨URL: http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.154.8446⟩

Chen, Stanley F./Goodman, Joshua: An Empirical Study of Smoothing Techniques for Language Modeling. In Proceedings of the 34th Annual Meeting on Association for Computational Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996, ACL '96 ⟨URL: http://dx.doi.org/10.3115/981863.981904⟩, 310–318

Cortes, Corinna/Vapnik, Vladimir: Support-Vector Networks. Mach. Learn. 20 September 1995, Nr. 3, 273–297 ⟨URL: http://dx.doi.org/10.1023/A:1022627411411⟩, ISSN 0885–6125

Cover, T./Hart, P.: Nearest neighbor pattern classification. Information Theory, IEEE Transactions on, 13 1967, Nr. 1, 21–27 ⟨URL: http://dx.doi.org/10.1109/TIT.1967.1053964⟩, ISSN 0018–9448

Cover, T.M./Thomas, J.A.: Elements of Information Theory. Wiley, 2006, ISBN 9780471748816

Dagan, Ido/Pereira, Fernando/Lee, Lillian: Similarity-based Estimation of Word Cooccurrence Probabilities. In Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, ACL '94 ⟨URL: http://dx.doi.org/10.3115/981732.981770⟩, 272–278

Davis, Timothy A.: Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms). Society for Industrial and Applied Mathematic, 9 2006, 217 Seiten, ISBN 9780898716139

Debole, Franca/Sebastiani, Fabrizio: Supervised Term Weighting for Automated Text Categorization. In Proceedings of the 2003 ACM Symposium on Applied Computing. New York, NY, USA: ACM,

2003, SAC '03 ⟨URL: http://dx.doi.org/10.1145/952532.952688⟩, ISBN 1–58113–624–2, 784–788

Deerwester, Scott et al.: Indexing by latent semantic analysis. Journal Of The American Society For Information Science, 41 1990, Nr. 6, 391–407

Deng, Zhi-Hong et al.: A Comparative Study on Feature Weight in Text Categorization. In **Yu, JeffreyXu et al. (Hrsg.):** Advanced Web Technologies and Applications. Band 3007, Springer Berlin Heidelberg, 2004 ⟨URL: http://dx.doi.org/10.1007/978-3-540-24655-8_64⟩, ISBN 978–3–540–21371–0, 588–597

Dietterich, Thomas G./Bakiri, Ghulum: Solving Multiclass Learning Problems via Error-correcting Output Codes. J. Artif. Int. Res. 2 Januar 1995, Nr. 1, 263–286 ⟨URL: http://dl.acm.org/citation.cfm?id=1622826.1622834⟩, ISSN 1076–9757

Domingos, Pedro/Pazzani, Michael: On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss. Mach. Learn. 29 November 1997, Nr. 2-3, 103–130 ⟨URL: http://dx.doi.org/10.1023/A:1007413511361⟩, ISSN 0885–6125

Fix, Evelyn/J.L. Hodges, Jr.: Discriminatory Analysis: Nonparametric Discrimination: Consistency Properties. 1951, Nr. Project 21-49-004, Report Number 4, 261–279

Forsyth, RichardS.: New Directions in Text Categorization. In **Gammerman, Alex (Hrsg.):** Causal Models and Intelligent Data Management. Springer Berlin Heidelberg, 1999 ⟨URL: http://dx.doi.org/10.1007/978-3-642-58648-4_11⟩, ISBN 978–3–642–63682–0, 151–185

Frakes, William B.: Stemming Algorithms. In Information Retrieval: Data Structures & Algorithms. Prentice Hall, 1992, ISBN 9780134638379, 131–160

Fuhr, N./Knorz, G. E.: Retrieval Test Evaluation of a Rule Based Automatic Indexing (AIR/PHYS). In Proc. Of the Third Joint BCS and ACM Symposium on Research and Development in Information Retrieval. New York, NY, USA: Cambridge University Press, 1984 ⟨URL: http://dl.acm.org/citation.cfm?id=2999.3031⟩, ISBN 0–521–26865–6, 391–408

Golub, Gene H./Loan, Charles F. Van: Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences). fourth edition Auflage. Johns Hopkins University Press, 12 2012, 784 Seiten, ISBN 9781421407944

Gordon, Michael/Kochen, Manfred: Recall-precision Trade-off: A Derivation. J. Am. Soc. Inf. Sci. 40 Mai 1989, Nr. 3, 145–151 ⟨URL: http://dx.doi.org/10.1002/(SICI)1097-4571(198905)40:3<145::AID-ASI1>3.0.CO;2-I⟩, ISSN 0002–8231

Grefenstette, Gregory/Tapanainen, Pasi: What is a word, What is a sentence? Problems of Tokenization. In Proceedings of the 3rd Conference on Computational Lexicography and Text Research. Budapest, 1994, 79–87

Group, Glasgow Information Retrieval: Stop word list. ⟨URL: http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words⟩ – Zugriff am 2014-01-19

Guthrie, Louise/Walker, Elbert/Guthrie, Joe: Document Classification by Machine: Theory and Practice. In Proceedings of the 15th Conference on Computational Linguistics - Volume 2. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994, COLING '94 ⟨URL: http://dx.doi.org/10.3115/991250.991322⟩, 1059–1063

Hastie, T./Tibshirani, R./Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2. Auflage. Springer, 2009, Springer Series in Statistics, ISBN 9780387848587

Hinneburg, Alexander/Aggarwal, Charu C./Keim, Daniel A.: What Is the Nearest Neighbor in High Dimensional Spaces? In Proceedings of the 26th International Conference on Very Large Data Bases. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, VLDB '00 ⟨URL: http://dl.acm.org/citation.cfm?id=645926.671675⟩, ISBN 1–55860–715–3, 506–515

Hull, David A.: Stemming Algorithms: A Case Study for Detailed Evaluation. J. Am. Soc. Inf. Sci. 47 Januar 1996, Nr. 1, 70–84 ⟨URL: http://dx.doi.org/10.1002/(SICI)1097-4571(199601)47:1<70::AID-ASI7>3.3.CO;2-Q⟩, ISSN 0002–8231

Indyk, Piotr/Motwani, Rajeev: Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In Proceedings of the

Thirtieth Annual ACM Symposium on Theory of Computing. New York, NY, USA: ACM, 1998, STOC '98 ⟨URL: http://dx.doi.org/10.1145/276698.276876⟩, ISBN 0–89791–962–9, 604–613

Jackson, P./Moulinier, I.: Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization. John Benjamins Pub., 2002, Natural language processing, ISBN 9789027249890

Jiang, Shengyi et al.: An Improved K-nearest-neighbor Algorithm for Text Categorization. Expert Syst. Appl. 39 Januar 2012, Nr. 1, 1503–1509 ⟨URL: http://dx.doi.org/10.1016/j.eswa.2011.08.040⟩, ISSN 0957–4174

Jones, Eric et al.: SciPy: Open source scientific tools for Python. ⟨URL: http://www.scipy.org⟩ – Zugriff am 2014-01-19

Jones, Karen Spärck: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28 1972, 11–21

Jurafsky, Dan/Martin, James H.: Speech and Language Processing. 2. Auflage. Prentice Hall, 4 2008, 1024 Seiten, ISBN 9780135041963

Kalt, T.: A New Probabilistic Model of Text Classification and Retrieval TITLE2:. Amherst, MA, USA: University of Massachusetts, 1998 – Technischer Bericht

Kessler, Brett/Numberg, Geoffrey/Schütze, Hinrich: Automatic Detection of Text Genre. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, 1997, ACL '98 ⟨URL: http://dx.doi.org/10.3115/976909.979622⟩, 32–38

Lan, Man et al.: Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 31 2009, Nr. 4, 721–735, ISSN 0162–8828

Lee, Joon Ho: Combining Multiple Evidence from Different Properties of Weighting Schemes. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM, 1995, SIGIR '95 ⟨URL: http://dx.doi.org/10.1145/215206.215358⟩, ISBN 0–89791–714–6, 180–188

Lewis, David D.: Evaluating Text Categorization. In Proceedings of the Workshop on Speech and Natural Language. Stroudsburg, PA, USA: Association for Computational Linguistics, 1991, HLT '91 ⟨URL: http://dx.doi.org/10.3115/112405.112471⟩, 312–318

Lewis, David D.: Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In Proceedings of the 10th European Conference on Machine Learning. London, UK, UK: Springer-Verlag, 1998, ECML '98 ⟨URL: http://dl.acm.org/citation.cfm?id=645326.649711⟩, ISBN 3–540–64417–2, 4–15

Liu, Ying/Loh, Han Tong/Sun, Aixin: Imbalanced Text Classification: A Term Weighting Approach. Expert Syst. Appl. 36 Januar 2009, Nr. 1, 690–701 ⟨URL: http://dx.doi.org/10.1016/j.eswa.2007.10.042⟩, ISSN 0957–4174

Lovins, J. B.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics, 11 1968, 22–31

Manning, C.D./Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, 1999, ISBN 9780262133609

Maron, M. E.: Automatic Indexing: An Experimental Inquiry. J. ACM, 8 Juli 1961, Nr. 3, 404–417 ⟨URL: http://dx.doi.org/10.1145/321075.321084⟩, ISSN 0004–5411

McCallum, Andrew/Nigam, Kamal: A comparison of event models for Naive Bayes text classification. In AAAI-98 Workshop on Learning for Text Categorization. AAAI Press, 1998, 41–48

Mercer, J.: Functions of positive and negative type, and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society, London, 209 1909, 415–446

Mitchell, Tom: Twenty Newsgroups Data Set. ⟨URL: http://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups⟩ – Zugriff am 2014-01-19

Mitchell, Tom M.: Machine Learning. 1. Auflage. McGraw-Hill Science/Engineering/Math, 3 1997, 432 Seiten, ISBN 9780070428072

Mohri, Mehryar/Rostamizadeh, Afshin/Talwalkar, Ameet: Foundations of Machine Learning (Adaptive Computation and Machine Learning series). The MIT Press, 8 2012, 432 Seiten, ISBN 9780262018258

Mori, Shunji/Nishida, Hirobumi/Yamada, Hiromitsu: Optical Character Recognition (Wiley Series in Microwave and Optical Engineering). 1. Auflage. Wiley-Interscience, 4 1999, 560 Seiten, ISBN 9780471308195

Nigam, Kamal et al.: Learning to Classify Text from Labeled and Unlabeled Documents. In Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998, AAAI '98/IAAI '98 ⟨URL: http://dl.acm.org/citation.cfm?id=295240.295806⟩, ISBN 0–262–51098–7, 792–799

Pedregosa, F. et al.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12 2011, 2825–2830

Pekalska, Elzbieta/Duin, Robert P. W.: The Dissimilarity Representation for Pattern Recognition: Foundations And Applications (Machine Perception and Artificial Intelligence). World Scientific Pub Co Inc, 12 2005, 636 Seiten, ISBN 9789812565303

Peteiro-Barral, Diego/Guijarro-Berdiñas, Bertha: A survey of methods for distributed machine learning. Progress in Artificial Intelligence, 2 2013, Nr. 1, 1–11 ⟨URL: http://dx.doi.org/10.1007/s13748-012-0035-5⟩, ISSN 2192–6352

Porter, Martin: An algorithm for suffix stripping. Program: electronic library and information systems, 14 1980, Nr. 3, 130–137

Prettenhofer, Peter et al.: Classification of text documents using sparse features. ⟨URL: http://scikit-learn.org/stable/auto_examples/document_classification_20newsgroups.html⟩ – Zugriff am 2014-01-20

Quinlan, J. R.: Induction of Decision Trees. Mach. Learn. 1 März 1986, Nr. 1, 81–106 ⟨URL: http://dx.doi.org/10.1023/A:1022643204877⟩, ISSN 0885–6125

Quinlan, J. Ross: C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, ISBN 1–55860–238–0

Robertson, Stephen: Understanding inverse document frequency: On theoretical arguments for IDF. Journal of Documentation, 60 2004, Nr. 5, 503–520

Salton, G./Wong, A./Yang, C. S.: A Vector Space Model for Automatic Indexing. Commun. ACM, 18 November 1975, Nr. 11, 613–620 ⟨URL: http://dx.doi.org/10.1145/361219.361220⟩, ISSN 0001–0782

Salton, Gerard/Buckley, Christopher: Term-weighting Approaches in Automatic Text Retrieval. Inf. Process. Manage. 24 August 1988, Nr. 5, 513–523 ⟨URL: http://dx.doi.org/10.1016/0306-4573(88)90021-0⟩, ISSN 0306–4573

Scikit-learn: Confusion matrix. ⟨URL: http://scikit-learn.org/stable/auto_examples/plot_confusion_matrix.html⟩ – Zugriff am 2014-01-20

Sebastiani, Fabrizio: Machine Learning in Automated Text Categorization. ACM Comput. Surv. 34 März 2002, Nr. 1, 1–47 ⟨URL: http://dx.doi.org/10.1145/505282.505283⟩, ISSN 0360–0300

Shannon, C. E.: A Mathematical Theory of Communication. Bell System Technical Journal, 27 1948, Nr. 3, 379–423 ⟨URL: http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x⟩, ISSN 1538–7305

Shawe-Taylor, John/Cristianini, Nello: Kernel Methods for Pattern Analysis. Cambridge University Press, 6 2004, 474 Seiten, ISBN 9780521813976

Silva, C./Ribeiro, B.: Inductive Inference for Large Scale Text Classification: Kernel Approaches and Techniques. Springer, 2009, Studies in Computational Intelligence, ISBN 9783642045325

Silva Conrado, Merley/Laguna Gutiérrez, VíctorAntonio/Rezende, SolangeOliveira: Evaluation of Normalization Techniques in Text Classification for Portuguese. In **Murgante, Beniamino et al. (Hrsg.):** Computational Science and Its Applications – ICCSA 2012. Band 7335, Springer Berlin Heidelberg, 2012 ⟨URL: http://dx.doi.org/10.1007/978-3-642-31137-6_47⟩, ISBN 978–3–642–31136–9, 618–630

Siolas, Georges/Buc, Florence d'Alché: Support Vector Machines Based on a Semantic Kernel for Text Categorization. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 5 - Volume 5. Washington, DC, USA: IEEE Computer Society, 2000, IJCNN '00 ⟨URL: http://dl.acm.org/citation.cfm?id=846232.849038⟩, ISBN 0–7695–0619–4, 5205–

Slaney, M./Casey, M.: Locality-Sensitive Hashing for Finding Nearest Neighbors [Lecture Notes]. Signal Processing Magazine, IEEE, 25 2008, Nr. 2, 128–131 ⟨URL: http://dx.doi.org/10.1109/MSP.2007.914237⟩, ISSN 1053–5888

Soucy, Pascal/Mineau, Guy W.: Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. In Proceedings of the 19th International Joint Conference on Artificial Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, IJCAI'05 ⟨URL: http://dl.acm.org/citation.cfm?id=1642293.1642474⟩, 1130–1135

Sriram, Bharath et al.: Short Text Classification in Twitter to Improve Information Filtering. In Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM, 2010, SIGIR '10 ⟨URL: http://dx.doi.org/10.1145/1835449.1835643⟩, ISBN 978–1–4503–0153–4, 841–842

Sun, Jian-Tao et al.: Supervised Latent Semantic Indexing for Document Categorization. In Proceedings of the Fourth IEEE International Conference on Data Mining. Washington, DC, USA: IEEE Computer Society, 2004, ICDM '04 ⟨URL: http://dl.acm.org/citation.cfm?id=1032649.1033524⟩, ISBN 0–7695–2142–8, 535–538

Turney, Peter D./Pantel, Patrick: From Frequency to Meaning: Vector Space Models of Semantics. J. Artif. Int. Res. 37 Januar 2010, Nr. 1, 141–188 ⟨URL: http://dx.doi.org/10.1109/MSP.2007.914237⟩, ISSN 1076–9757

Tzeras, Kostas/Hartmann, Stephan: Automatic Indexing Based on Bayesian Inference Networks. In Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York, NY, USA: ACM, 1993, SIGIR '93

⟨URL: http://dx.doi.org/10.1145/160688.160691⟩, ISBN 0–89791–605–0, 22–35

Uysal, Alper Kursat/Gunal, Serkan: The Impact of Preprocessing on Text Classification. Inf. Process. Manage. 50 Januar 2014, Nr. 1, 104–112 ⟨URL: http://dx.doi.org/10.1016/j.ipm.2013.08.006⟩, ISSN 0306–4573

Van Rijsbergen, C.J.: Information retrieval. Butterworths, 1979, ISBN 9780408709293

Wang, Deqing/Zhang, Hui: Inverse-Category-Frequency based Supervised Term Weighting Schemes for Text Categorization. J. Inf. Sci. Eng. 29 2013, Nr. 2, 209–225

Xu, Yan et al.: A study on mutual information-based feature selection for text categorization. Journal of Computational Information Systems, 3 2007, Nr. 3, 1007–1012 ⟨URL: http://doras.dcu.ie/16194/⟩, ISSN 1553–9105

Yang, Yiming: An Evaluation of Statistical Approaches to Text Categorization. Inf. Retr. 1 Mai 1999, Nr. 1-2, 69–90 ⟨URL: http://dx.doi.org/10.1023/A:1009982220290⟩, ISSN 1386–4564

Yang, Yiming/Pedersen, Jan O.: A Comparative Study on Feature Selection in Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ICML '97 ⟨URL: http://dl.acm.org/citation.cfm?id=645526.657137⟩, ISBN 1–55860–486–3, 412–420

Yang, Yiming/Wilbur, John: Using Corpus Statistics to Remove Redundant Words in Text Categorization. J. Am. Soc. Inf. Sci. 47 Mai 1996, Nr. 5, 357–369 ⟨URL: http://dx.doi.org/10.1002/(SICI)1097-4571(199605)47:5<357::AID-ASI3>3.3.CO;2-0⟩, ISSN 0002–8231

Zobel, Justin/Moffat, Alistair: Exploring the Similarity Space. SIGIR Forum, 32 April 1998, Nr. 1, 18–34 ⟨URL: http://dx.doi.org/10.1145/281250.281256⟩, ISSN 0163–5840