UNIVERSITÄT PADERBORN

Die Universität der Informationsgesellschaft

Fakultät für Elektrotechnik, Informatik und Mathematik
Institut für Informatik

Hierarchische Textklassifikation als Verfahren zur Topic Identification

Studienarbeit zur
Erlangung des Grades
Bachelor of Science in Information Systems
für den Studiengang Informatik

vorgelegt am 18. Juli 2006 bei
Prof. Dr. Benno Stein
und Prof. Dr. Wilhelm Schäfer

von

Maik Anderka
Leipziger Str. 1
34454 Bad Arolsen
manderka@upb.de

Nedim Lipka
Remser Weg 60
33428 Harsewinkel
lippi@upb.de

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum Unterschrift

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

Ort, Datum Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Im Informationszeitalter ist das WWW allgegenwärtig. Es ist möglich, von fast jedem Ort und zu jeder Zeit eine Vielzahl von Informationen abzurufen. Der rasante Zuwachs von, vor allem textuellen Daten, bringt jedoch ein allgemeines Problem mit sich. Die große Anzahl von unorganisierten Dokumenten erschwert dem Benutzer zum einen den Umgang mit diesen Datenmengen und zum anderen das Auffinden der für ihn relevanten Dokumente. Dies führt dazu, dass das Organisieren von Dokumenten sowie deren Präsentation dem Benutzer gegenüber zunehmend an Bedeutung gewinnt.

Moderne Suchmaschinen wie z. B. AIsearch begegnen diesen Problemen, indem sie die Suchergebnisse in Kategorien organisieren. Die Kategorien beinhalten Dokumente gleichen Inhaltes. Auf Grund der Kategorisierung wird eine semantische und somit effiziente Suche möglich. Problematisch ist allerdings, für die Kategorien aussagekräftige Label automatisiert zu generieren, die den Inhalt der darin enthaltenen Dokumente gut wiedergeben. Problemstellungen dieser Art werden als Topic Identification bezeichnet. Die Vorteile, die eine solche Kategorisierung bietet, sind jedoch stark von der Qualität der Label abhängig.

In Kapitel 2 wird der Problembereich genauer beschrieben. Weiterhin werden verschiedene Ansätze zur Topic Identification erläutert und das in dieser Arbeit vorgestellte Verfahren präsentiert.

Kapitel 3 beschreibt die wichtigsten Techniken aus dem Bereich der Textklassifikation, die in dieser Arbeit angewandt werden. In Kapitel 4 wird die Hierarchische Textklassifikation vorgestellt sowie verschiedene hierarchische Klassifikationsverfahren. Außerdem wird eine Übersicht über frühere Arbeiten zur Hierarchischen Textklassifikation gegeben.

Im Rahmen dieser Arbeit sind verschiedene Softwarewerkzeuge und ein Framework entwickelt worden. Diese werden in Kapitel 5 erläutert. Kapitel 6 zeigt die Experimente, die durchgeführt worden sind sowie deren Ergebnisse. Eine Zusammenfassung dieser Arbeit und ein Ausblick werden in Kapitel 7 gegeben.

Die Arbeit ist eine Gruppenarbeit mit der Aufteilung:

Maik Anderka	Nedim Lipka
1 Einleitung	
	2.0 Topic Identification
	2.1 Die Zeit vor dem Semantic Web
	2.2 Ansätze zur Topic Identification
	2.3 Hierarchische Textklassifikation als Verfahren zur Topic Identification
2.4 DMoz	
	3.0 Textklassifikation
	3.1 Dokument Indexierung
	3.2 Evaluierung
3.3.1 Lineare Klassifizierer	
3.3.2 Probabilistische Klassifizierer	3.3.2 Probabilistische Klassifizierer
	3.3.3 Klassifiziererensembles
	4.0 Hierarchische Textklassifikation
	4.1 Methodik
4.2 Frühere Arbeiten	
4.3 Shrinkage	
	4.4 Subsumption
4.5 Multi-Pfad-Klassifikation	
5.0 Software zur Evaluierung	
5.1 Framework	
5.2 Naïve-Bayes-Klassifikationsmodell	
	5.3 Weka-Klassifikationsmodell
	5.4 Subsumption-Klassifikationsmodell
	5.5 Evaluierung
	5.6 Indexierung
	5.7 Grafische Benutzungsschnittstelle
	6.0 Experimente und Ergebnisse
	6.1 Vorbereitung
6.2 Allgemeine Parameter	6.2 Allgemeine Parameter
	6.3 Domänenspezifische Kategorisierung
6.4 Abstrakte Kategorisierung	
7 Zusammenfassung und Ausblick	7 Zusammenfassung und Ausblick

2. Topic Identification

Topic Identification ist das maschinelle Erstellen von Überschriften, so genannte Labeln, für Dokumente. Topic-Identification-Verfahren können einerseits einem einzelnen Dokument ein Label zuweisen, aber auch einer Menge von Dokumenten mit ähnlichem Inhalt.

Ein Label wird erstellt, indem der Inhalt eines Dokumentes analysiert und zusammengefasst wird. Maschinen können die Struktur eines Dokumentes analysieren, jedoch nicht die Semantik des Inhaltes. Das Semantic Web soll dies ermöglichen, ist aber erst in Zukunft verfügbar. Andere Verfahren müssen verwendet werden, die die Dokumentstruktur zur Topic Identification verwenden.

Dieses Kapitel unterteilt Topic-Identification-Verfahren und nennt hierfür Beispiele. Ein neues Verfahren wird vorgestellt: „Hierarchische Textklassifikation als Verfahren zur Topic Identification“.

2.1. Die Zeit vor dem Semantic Web

Die Inhalte und Dokumente im WWW sind von Menschen für Menschen gemacht. Eine maschinelle Verarbeitung von Semantik ist nicht eingeplant.

Wir befinden uns im Wandel zu einem von Maschinen lesbaren Web, dem Semantic Web. Mit dem Semantic Web lassen sich Anwendungen implementieren, die den Menschen in vielen Bereichen unterstützen. Tim Berners-Lee beschreibt in einem Artikel der Scientific American von seiner neuen Version des Webs: Softwareagenten managen Aufgaben wie das Finden eines guten Arztes, um dort einen Termin einzutragen [BLHL01]. Um eine grobe Vorstellung vom Aufbau des Semantic Webs zu bekommen, folgt eine knappe Umschreibung.

Die wichtigsten Komponenten des Semantic Webs sind die Extensible Markup Language (XML), das Resource Description Framework (RDF) und die Ontologien. Unter einer Ontologie versteht man im Bereich der Wissensrepräsentation ein formal definiertes System von Begriffen und/oder Konzepten und Relationen zwischen diesen Begriffen. Die XML erlaubt es,

Dokumente zu strukturieren, bietet jedoch nicht die Möglichkeit, den XML Strukturen eine Semantik zu geben. Das RDF ermöglicht es, Inhalte in Bezug zueinander zu setzen. Ein RDF Statement ist ein Tripel von Objekten, die dem Subjekt, Prädikat und Objekt der natürlichen Sprache ähneln. Die Tripel können mit XML-Strukturen beschrieben werden. Inhalte des Tripels werden mit Universal Resource Identifiern (URIs) referenziert. Jeder ist somit in der Lage Neues zu definieren, indem eine neue URI angelegt wird. Damit ein Programm verschiedene URIs mit gleicher Semantik vergleichen kann, benötigt es Weltwissen über die Semantiken. Zeigt beispielsweise eine URI auf den Begriff „Auto“, so hat dieser Begriff die gleiche Semantik wie „PKW“. Für das Semantic Web sind Ontologien notwendig, die eine solche Semantik repräsentieren können. Ontologien sind in der Lage, Relationen zwischen Begriffen zu beschreiben. Darüber hinaus können Ontologien Inferenzregeln bereitstellen, mit denen man weitere Informationen aus den Relationen der Einträge in der Ontologie ableiten kann. Eine Voraussetzung für Anwendungen, die miteinander Daten austauschen, ist, dass sie die gleiche Ontologie verwenden.

Mit dem Semantic Web ließen sich komplexe Anwendungen realisieren, die bisher nicht möglich sind. Das Semantic Web ist auf eine weite Verbreitung und auf die Erfüllung seiner Spezifikationen auf den Webseiten angewiesen. Darüber hinaus muss eine allgemein zugängliche Ontologie mit genügend Weltwissen existieren. Weitere Problemstellungen liegen beim Entwurf einer Webseite für das Semantic Web. Der Entwickler hat zusätzlichen Aufwand zu tragen, indem er die Vorgaben des neuen Webs erfüllen muss, wie beispielsweise das Setzen der RDF-Klauseln. Ferner ist zunächst nicht überprüfbar, ob die Referenzen des Inhaltes zu einer Ontologie korrekt sind, solange man keine Möglichkeit hat Semantik zu erschließen und zusammenzufassen.

Das Semantic Web kann die Topic-Identification-Aufgabe für Dokumente im WWW lösen. Der Inhalt eines Dokumentes kann über die Referenzen zu einer Ontologie ermittelt werden.

2.2. Ansätze zur Topic Identification

Ein Beispiel für die Anwendung von Topic Identification findet sich in Suchmaschinen. Der Benutzer einer Suchmaschine wünscht sich eine gute Übersicht der Ergebnisse seiner Anfrage. Der Benutzer muss in der Darstellung der Ergebnisse schnell explorieren können, um das Relevante finden zu können. Eine Strategie, die eine breite Akzeptanz gefunden hat, ist mit Google[1] implementiert worden: Den Ergebnissen einer Suchanfrage wird mit einem Ranking-

[1] http://www.google.com

Algorithmus ein Wert für die Relevanz zugewiesen. Die Ergebnisse werden sortiert ausgegeben. Die Semantik wird dabei nicht berücksichtigt, sodass Dokumente mit unterschiedlichen Semantiken vermischt auftreten. Moderne Suchmaschinen vermeiden dies. Beispiele sind die Meta-Suchmaschinen[2] AIsearch[3] oder Vivisimo[4]. Die Suchergebnisse werden in Gruppen gleichen Inhaltes aufgeteilt, so genannte Cluster. Verfahren der Topic Identification können Label für die Cluster erzeugen, die deren Inhalt beschreiben. Der Benutzer der Suchmaschine sieht in einer ersten Näherung eine Liste von Labeln und kann entscheiden, welches Cluster er genauer betrachten möchte. Ein schnelles Explorieren wird in diesem Fall mit aussagekräftigen Labeln ermöglicht. Deshalb ist für Anwendungen dieser Art das Hauptkriterium eines Topic-Identification-Verfahrens die Aussagekraft des Labels.

Topic-Identification-Verfahren lassen sich in zwei Kategorien einteilen:

- interne Topic Identification und
- externe Topic Identification

Mit interner Topic Identification ist gemeint, dass aus den vorhandenen Wörtern eines Dokumentes ein Label generiert wird. Das Label kann keine zusätzlichen Wörter enthalten, die evtl. nötig wären, um ein aussagekräftiges Label zu erzeugen. Es ist mit einem internen Verfahren nicht möglich, einen abstrakten Oberbegriff in einem Label zu verwenden, wenn dieser nicht im zu labelnden Dokument vorhanden ist. Mit externer Topic Identification wäre dies möglich, denn hierbei geht man von einer externen Quelle von Wörtern aus, die der Eingabe zugeordnet werden.

2.2.1. Interne Topic Identification

Häufig werden Verfahren verwendet, die einzelne Wörter aus den zu labelnden Dokumente zu einem Label zusammensetzen. Wegen der Verwendung ausschließlich im Dokument vorhandener Wörter, wird dieses Vorgehen interne Topic Identification genannt. Die Probleme sind, dass Wörter ausgewählt werden können, die zusammen keinen Sinn ergeben, den Inhalt nicht korrekt zusammenfassen bzw. wenig über den Inhalt aussagen oder mehrdeutig sind. Trotz dieser Schwierigkeiten handelt es sich bei den meisten Topic-Identification-Algorithmen um interne Verfahren.

[2]Eine Meta-Suchmaschine fasst die Ergebnisse unterschiedlicher Suchmaschinen zusammen.
[3]http://www.aisearch.de
[4]http://vivisimo.com

Der Topic-Identification-Algorithmus *Weighted Centroid Covering* aus der Arbeit [SM04] wird an dieser Stelle exemplarisch dargestellt. In der genannten Arbeit werden einige formale Eigenschaften definiert, die zu erfüllen sind, um die Qualität eines Labels sicherzustellen. Dazu gehören, dass sich die Label verschiedener Cluster unterscheiden müssen (Unique), dass jedes Dokument eines Clusters im Label erfasst sein muss (Summarizing), dass das häufigste Wort eines Dokumentes im Label (Expressive) enthalten ist und weitere.

Algorithmus 1 : Weighted Centroid Covering (WCC)

Input : Clustering C. Vocabular Dic
l. Anzahl der Terme für ein Label.
k. Anzahl des Vorkommens eines Terms in verschiedenen Clustern.
Output : Labelfunktion τ

foreach $w \in Dic$ **do**
    Bestimme für w das 1 bis k häufigste Vorkommen über alle Cluster und speichere (w, C_i) im Vektor V.
end

Sortiere V nach Häufigkeit von $w \in V$.

for 0 *bis* l **do**
    (Round-Robin Strategie)
    Füge zu jedem Label $\tau(C_i)$ das häufigste w und lösche es in V, wenn gilt:
        1. $(w, C_i) \in V$
        2. w wurde in der aktuellen Iteration noch nicht verwendet.
end

return τ;

Der Algorithmus stellt die Eigenschaften Unique und Summarizing sicher. Durch die Round-Robin-Strategie lässt sich auch die Expressive-Eigenschaft erfüllen, jedoch nur, wenn die ersten beiden Eigenschaften nicht verletzt werden.

Ein weiteres internes Verfahren wird in der Arbeit [FPW+99] beschrieben. Es wird versucht, Schlüsselphrasen zu extrahieren, die den Inhalt eines Dokumentes bzw. mehrerer Dokumente treffend beschreiben. Die Extrahierung wird als binäre Klassifikationsaufgabe formuliert: Ob eine Phrase entweder eine Schlüsselphrase ist oder nicht, wird durch einen Klassifizierer entschieden. Die Klassifikationsmerkmale sind der Abstand der Phrase zum Anfang des Dokumentes und ein Maß für die Häufigkeit der Phrase. Für die Evaluierung dieses Verfahrens und das Generieren eines Klassifizierers muss a priori bekannt sein, welche Schlüsselphrasen für ein Dokument in Frage kommen. Aus diesem Grund werden Dokumente verwendet, zu denen der Autor Schlüsselphrasen angegeben hat.

Beide genannten internen Verfahren besitzen unterschiedliche Herangehensweisen. Sie haben

die Gemeinsamkeit, dass ausschließlich Terme, die literal in den Dokumenten vorhanden sind, als Label in Frage kommen.

2.2.2. Externe Topic Identification

Die Ausgabe der externen Topic Identification ist ein Label, das aus einer externen Quelle entnommen wird. Der Nachteil interner Verfahren, dass kein abstrakter Begriff verwendet werden kann, lässt sich durch ein externes Verfahren umgehen. Externes Wissen kann für das Label verwendet werden.

Grundlage der Label-Erstellung durch externe Topic Identification ist eine existierende Deskriptorenmenge, die für Label in Frage kommt. Man beachte, dass die Menge der Deskriptoren groß genug sein muss und sich über möglichst viele Themen erstrecken sollte. Zusätzlich sollten die Deskriptoren aussagekräftig sein und die Eigenschaft besitzen, Inhalte zusammenzufassen.

In dem folgenden Abschnitt wird beschrieben, wie ein Model zur externen Topic Identification aufgebaut werden kann.

2.3. Hierarchische Textklassifikation als Verfahren zur Topic Identification

In dieser Arbeit wird ein Topic-Identification-Modell vorgestellt, dass existierende Label der Eingabe zuweist, vgl. Abschnitt 2.2.2. Es wurde bereits herausgestellt, dass die Auswahl einer großen, aussagekräftigen Deskriptorenmenge die Qualität eines Topic-Identification-Verfahrens mitbestimmt.

Die Überschriften eines Verzeichnisses, dessen Einträge nach gleichem Inhalt gruppiert sind, haben die Anforderung, diese Inhalte treffend zu beschreiben. Dies kann sichergestellt werden, indem die Inhalte und Überschriften durch Experten erstellt werden. Je nach Wissensgebiet kann so ein Verzeichnis unterschiedliche Größen annehmen. Es gibt allerdings auch Verzeichnisse, wie beispielsweise Yahoo oder DMoz, die versuchen, für jedes Thema eine passende Kategorie bereitzustellen. Es wird angenommen, dass sich die Überschriften eines solchen Verzeichnisses besonders gut als Deskriptoren einer Topic Identification eignen, da die Menge sich über viele Themengebiete erstreckt und Inhalte gut zusammenfassen kann.

2.3.1. Ontologie-basierte Topic Identification

Man versteht unter Ontologie-basierter Topic Identification ein Topic-Identification-Verfahren, das zwei Eigenschaften einer Ontologie nutzt:

1. die Anordnung der Kategorien und
2. die Deskriptoren der Kategorien.

Die Namen von Kategorien werden als Deskriptoren verwendet. Im konkreten Szenario der Arbeit wird das DMoz-Verzeichnis benutzt. Das Verzeichnis bietet eine Vielzahl an hierarchisch angeordneten Kategorien, die einen Großteil der existierenden Wissensbereiche erschließen. Die Kategorienamen sind von Menschen gemacht worden und die Unterkategorien sind spezialisierend. Damit eignen sich die Namen als Deskriptorenmenge für eine externe Topic Identification. Die Struktur der Ontologie wird in zweierlei Hinsicht genutzt; zum einen ermöglicht sie ein Abstrahieren von Begriffen, und zum anderen ermöglicht sie eine hierarchische Textklassifikation.

In einer Ontologie wie dem DMoz-Verzeichnis sind die Kategorien so angeordnet, dass sie eine Hierarchie bilden. Eine Kategorie, die eine andere umfasst, ist allgemeiner. Diese Beziehung ist transitiv. Eine Kategorie kann als Abstrahierung von Dokumenten bezeichnet werden, da sie deren Inhalte zusammenfasst. Genauso fasst der Vorgänger mehrerer Kategorien deren Gemeinsamkeiten zusammen. Beginnend von einem Dokument in der Tiefe n einer Hierarchie, sind die Kategorie des Dokumentes und alle Vorgänger Abstrahierungen unterschiedlichen Grades. Diese Eigenschaft lässt sich nutzen, um mit einem Topic-Identification-Verfahren abstrakte Label zu erstellen. Dabei wird ein Dokument einer Kategorie zugeordnet. Je nach Abstrahierungsgrad wird der Deskriptor der Kategorie oder einer Vorgängerkategorie als Label verwendet.

Die Herausforderung liegt in der Zuordnung der Topic-Identification-Eingabe zu einer Kategorie in der Ontologie. Mit Hilfe hierarchischer Textklassifikation wird versucht, die Zuordnung in den Griff zu bekommen. Eine Herausforderung ist die große Anzahl von Kategorien. In der Arbeit werden Verfahren gezeigt und zum Teil in Experimenten erprobt, mit denen man möglichst gute Ergebnisse für eine Zuordnung erzielt.

2.3.2. Vorgehen

Um eine Vorstellung über hierarchische Textklassifikation zu gewinnen, wird das Vorgehen an dieser Stelle informell beschrieben.

Beginnend bei der Wurzel eines ausgewählten Teilbaumes der Ontologie, entscheidet ein Klassifizierer zu welchen Nachfolger die Eingabe gehört. Die Klassifikationsaufgabe wird also in ein kleineres Problem zerlegt. Rekursiv wird weiter entschieden, welcher nächste Nachfolger ausgewählt wird, bis dieser ein Blatt im Teilbaum ist. Die Label der Kategorien, die während einer Klassifizierung ausgewählt werden, werden zusammengefasst und der Eingabe zugeordnet.

Algorithmus 2 : Hierarchisches Klassifizieren

Input : $document, label, subtree$
Output : $label$

$classifier = subtree.$`getClassifier(`$label$`)`;

while $classifier \neq NULL$ **do**
    $label = classifier.$`classify(`$document$`)`;
    $classifier = subtree.$`getClassifier(`$label$`)`;
end

return *label*;

Beispiel: Sei T ein Teilbaum einer Ontologie der durch das UML-Klassendiagramm in **Abb. 2.1** dargestellt wird, wobei die abgebildeten Klassen den Kategorien der Ontologie entsprechen. Die Eingabe für das Beispiel sei nun eine Sammlung von Dokumenten über

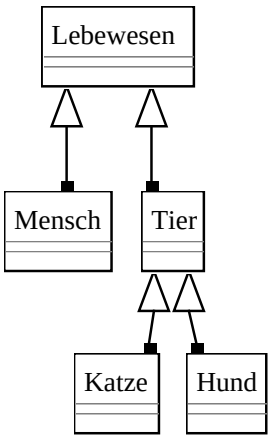


Abb. 2.1.: *Beispiel für einen Teilbaum einer Ontologie. Ein Dokument über Hunde fällt in die Kategorie „Hund“, die die Kategorie „Tier“ spezialisiert. „Tier“ ist eine Unterkategorie der Kategorie „Lebewesen“.*

Hunde, die durch einen Vektor *doc* repräsentiert wird. Der Aufruf des **Algorithmus 2** würde lauten: $Hierarchisches\ Klassifizieren(doc,$ „$Lebewesen$“, $T)$

Der erste Klassifizierer würde entscheiden, ob die Eingabe zur Kategorie „Mensch“ oder „Tier“ gehört. Für „Tier“ würde im nächsten Schritt ein Klassifizierer zwischen „Hund“

und „Katze“ wählen. Anschließend gibt es keine spezialisierendere Kategorie und die Ausgabe würde „Lebewesen.Tier.Hund“ lauten.

Wie werden die Klassifizierer für die Topic Identification erstellt? Die Elemente einer Kategorie im DMoz-Verzeichnis sind Links, die auf Webseiten zeigen und von Menschenhand der jeweiligen Kategorie zugeordnet sind. Ein Teil dieser Webseiten dient als Trainingsmenge, um Klassifizierer zu erstellen und ein anderer Teil zum Testen. Eine Software wählt eine festgelegte Anzahl an Webseiten zufällig aus und lädt diese herunter.

Damit ein Klassifizierer die heruntergeladenen Dokumente verwenden kann, müssen die Dokumente durch einen Indexierungsprozess aufbereitet werden. Erst dann können die Klassifizierer generiert und getestet werden.

Abschließend lassen sich die Ergebnisse einer Klassifikation zur Topic Identification verwenden, indem der Deskriptor der ausgewählten Kategorie als Label für die Eingabe der Topic Identification verwendet wird.

Das gesamte Vorgehen sieht wie folgt aus:

1. Auswahl eines Teilbaumes aus der Ontologie
2. Herunterladen von Dokumenten zum Trainieren und Evaluieren der Klassifikationsverfahren
3. Indexierung der heruntergeladenen Dokumente
4. Klassifizierererstellung
5. Topic Identification

2.4. DMoz

Bei dem Open Directory Project[5], auch bekannt als *DMoz* (**D**irectory at **Moz**illa), handelt es sich um eine freie, sehr umfangreiche und manuell erstellte Dokumentensammlung. Diese Sammlung wurde von einer Gemeinschaft, bestehend aus über 73 000 freiwilligen Editoren[6] erstellt und wird ständig erweitert und aktualisiert. Ziel ist es, das WWW zu organisieren, indem alle Dokumente eindeutig in Kategorien eingeteilt werden. Die Kategorien sind in einer hierarchischen Struktur organisiert. Die Dokumente werden von den Editoren auf Grund

[5] http://www.dmoz.org
[6] Daten von Juli 2006 (http://www.dmoz.org)

ihrer Inhalte manuell in die Ontologie einsortiert. Diese manuelle Zuordnung garantiert eine inhaltlich sinnvolle Organisation und ist daher von höherer Qualität als automatisch erstellte Ontologien.

2.4.1. Aufbau der Ontologie

Die DMoz-Ontologie besteht aus einer Menge von Kategorien, die in einer hierarchischen Struktur organisiert sind. Man kann sich die Ontologie als Baum vorstellen. Die Knoten des Baumes repräsentieren die Kategorien. Für jeden Knoten gilt, dass sein Elternknoten ein allgemeineres Konzept darstellt, wobei seine Nachfolger Spezialisierungen sind. Die Wurzel des Baumes stellt das allgemeinste Konzept dar und umfasst inhaltlich alle anderen. Die **Abb. 2.2** zeigt einen Ausschnitt der DMoz-Ontologie.

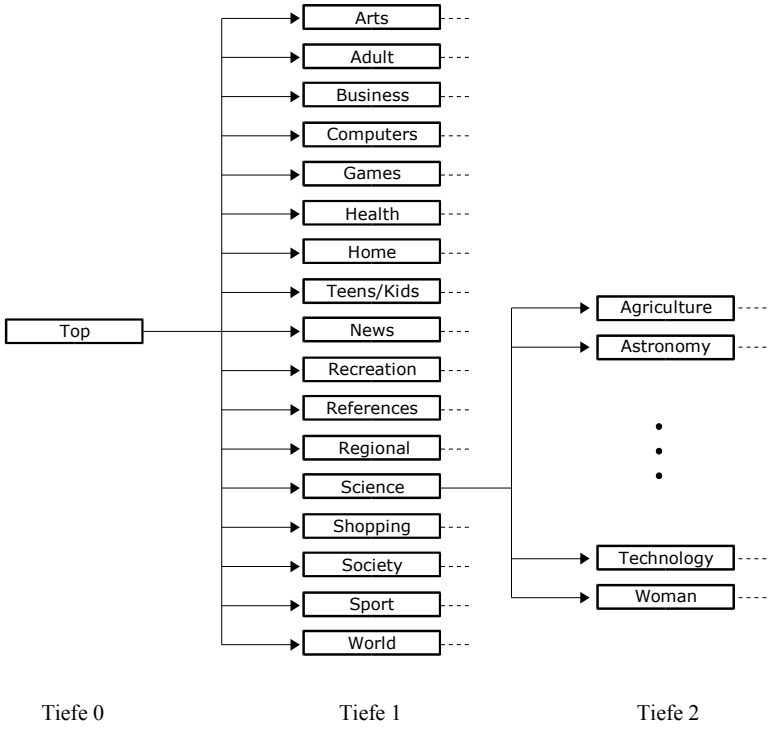


Abb. 2.2.: *Ausschnitt der DMoz-Ontologie. Dargestellt sind die Wurzel, alle Kategorien in Tiefe 1 sowie ein Auszug der Unterkategorien von Science, in Tiefe 2.*

Jede Kategorie besitzt einen in der gesamten Ontologie eindeutigen Deskriptor, der die Kategorie beschreibt. Der Deskriptor einer Kategorie c setzt sich aus den Namen der Kategorien zusammen, die auf dem Pfad von der Wurzel zu Kategorie c vorkommen. Beispielsweise ist der

Deskriptor der Kategorie Astronomy, eine Unterkategorie der Kategorie Science in Abb. 2.2, Top/Science/Astronomy.

Den Kategorien sind Dokumente zugeordnet. Jedes Dokument befindet sich in genau einer Kategorie. Im Fall der DMoz-Ontologie handelt es sich bei den Dokumenten um Webseiten. Eine Kategorie beinhaltet Dokumente gleichen Inhaltes. Beispielsweise handeln alle Dokumente, die sich in der Kategorie Top/Science/Astronomy befinden, von Astronomie. Je tiefer sich eine Kategorie in der Hierarchie befindet, desto spezialisierter ist sie. Die Dokumente sollten so tief wie möglich eingeordnet werden. Ein Dokument d sei beispielsweise der Kategorie Top/Science/Astronomy zugeordnet. Es handelt sich also um ein wissenschaftliches Dokument über Astronomie. Folglich wäre eine Zuordnung des Dokumentes d in die Kategorie Top/Science auch korrekt. Die Information, dass das Dokument von Astronomie handelt, würde in diesem Fall allerdings verloren gehen.

Bei dem Open Directory Projekt ist man um eine möglichst gute Kategorisierung bemüht. Deshalb befinden sich in der DMoz-Ontologie die meisten Dokumente in den Blättern. Die Kategorien, die durch die Blätter repräsentiert werden, sind die am meisten spezialisierten Kategorien. Die wenigen Dokumente, die den inneren Kategorien zugeordnet sind, wurden in dieser Arbeit ignoriert.

Die DMoz-Ontologie wird im RDF-Format vom Open Directory Projekt zum Download bereitgestellt[7]. Hierbei handelt es sich um die Datei content.rdf.u8.gz, die die gesamten Informationen der Ontologie enthält. In der **Abb. 2.3** ist ein Auszug der Datei zu sehen. Der <`Topic`>-Tag beschreibt hier eine Kategorie. Das Attribut `r:id` enthält den Deskriptor der Kategorie, z. B. Top/Arts. Mit dem <`catid`>-Tag wird jeder Kategorie eine eindeutige Id zugeordnet. Blattkategorien, z. B. Top/Arts/Movies/Titles/1/10_Rillington_Place, können außerdem <`link`>-Tags besitzen. Die <`link`>-Tags enthalten als Attribut `r:resource` die URL zu einem, der Kategorie zugeordneten, Dokument. Jedem dieser Verweise ist ein <`ExternalPage`>-Tag über das Attribut `about` zugeordnet, der einen Titel (<`d:Title`>) und eine Beschreibung (<`d:Description`>) zu dem Dokument enthält, auf das der Verweis zeigt. Weiterhin enthält der Tag den Deskriptor der Kategorie, der das Dokument zugeordnet ist (<`topic`>). Detailierte Informationen zu RDF können der W3-Spezifikation entnommen werden[8].

[7] http://rdf.dmoz.org/rdf/archive
[8] http://www.w3.org/RDF/

```
<?xml version='1.0' encoding='UTF-8' ?>
<RDF xmlns:r="http://www.w3.org/TR/RDF/" xmlns:d="http://purl.org/dc/elements/1.0/"
     xmlns="http://dmoz.org/rdf">
<!-- Generated at 2006-06-06 01:16:09 GMT on dust -->

<Topic r:id="Top">
  <catid>1</catid>
</Topic>

<ExternalPage about="">
  <topic>Top/</topic>
</ExternalPage>

<Topic r:id="Top/Arts">
  <catid>2</catid>
</Topic>
...

<Topic r:id="Top/Arts/Movies/Titles/1/10_Rillington_Place">
  <catid>205108</catid>
  <link r:resource="http://www.britishhorrorfilms.co.uk/rillington.shtml"/>
  <link r:resource="http://www.shoestring.org/mmi_revs/10-rillington-place.html"/>
  ...
</Topic>

<ExternalPage about="http://www.britishhorrorfilms.co.uk/rillington.shtml">
  <d:Title>British Horror Films: 10 Rillington Place</d:Title>
  <d:Description>Review which looks at plot especially the shocking features of it.</d:Description
      >
  <topic>Top/Arts/Movies/Titles/1/10_Rillington_Place</topic>
</ExternalPage>
...

</RDF>
```

Abb. 2.3.: *Auszug aus der Datei content.rdf.u8.gz. Die Datei enthält die DMoz-Ontologie im RDF-Format. Dargestellt ist der Header und die Definition von den drei Kategorien: Top, Top/Arts, Top/Arts/Movies/Titles/1/10_Rillington_Place.*

2.4.2. Größenordnung

Bei der DMoz-Ontologie handelt es sich um eine sehr umfangreiche Dokumentensammlung. Die in dieser Arbeit verwendete Version vom 11.06.2006 besteht aus 709 679 Kategorien und beinhaltet 4 766 935 Dokumente. Die entsprechende RDF-Datei ist komprimiert 308 MB groß und entpackt ca. zwei GB. Die Ontologie bzw. der Baum den die Ontologie darstellt besitzt Tiefe 14. Für den weiteren Verlauf dieser Arbeit wird festgelegt, dass die Wurzel immer Tiefe 0 besitzt. In den Diagrammen in **Abb. 2.4** ist die Größe und Beschaffenheit der DMoz-Ontologie grafisch dargestellt. Für detaillierte Daten siehe Anhang A.

Ein Großteil der Kategorien befindet sich in den mittleren Ebenen der Ontologie, etwa in Tiefe 4 bis 10. In diesem Bereich liegen auch ca. 94% der Dokumente. Der durchschnittliche Knotengrad pro Ebene zeigt, dass der Baum schnell sehr breit wird. Ab Tiefe 7 verzweigt sich der Baum jedoch kaum noch. Lediglich einzelne Pfade reichen noch bis in tiefere Ebenen.

Eine semantische Sicht auf die DMoz-Ontologie bieten die Diagramme, die kumulierte Werte enthalten. Hier wurde berücksichtigt, dass jede Kategorie inhaltlich alle Nachfolgekategorien umfasst. Beispielsweise berechnet sich die kumulierte Anzahl der Kategorien in Tiefe k, wie folgt:

$$\#\,\text{Kategorien in Tiefe k} + \sum (\#\,\text{Kategorien in Tiefe größer k})$$

Die Kategorien in geringer Tiefe beinhalten sehr viele andere Kategorien. Folglich sind die Kategorien in den oberen Ebenen sehr allgemein. Eine Spezialisierung ist erst ab den mittleren Ebenen vorhanden. Dies ist gut an der durchschnittlichen Anzahl der kumulierten Kategorien pro Knoten zu sehen. Der Wert stellt die durchschnittliche Anzahl der Kategorien dar, die ein Knoten in der entsprechenden Tiefe inhaltlich umfasst.

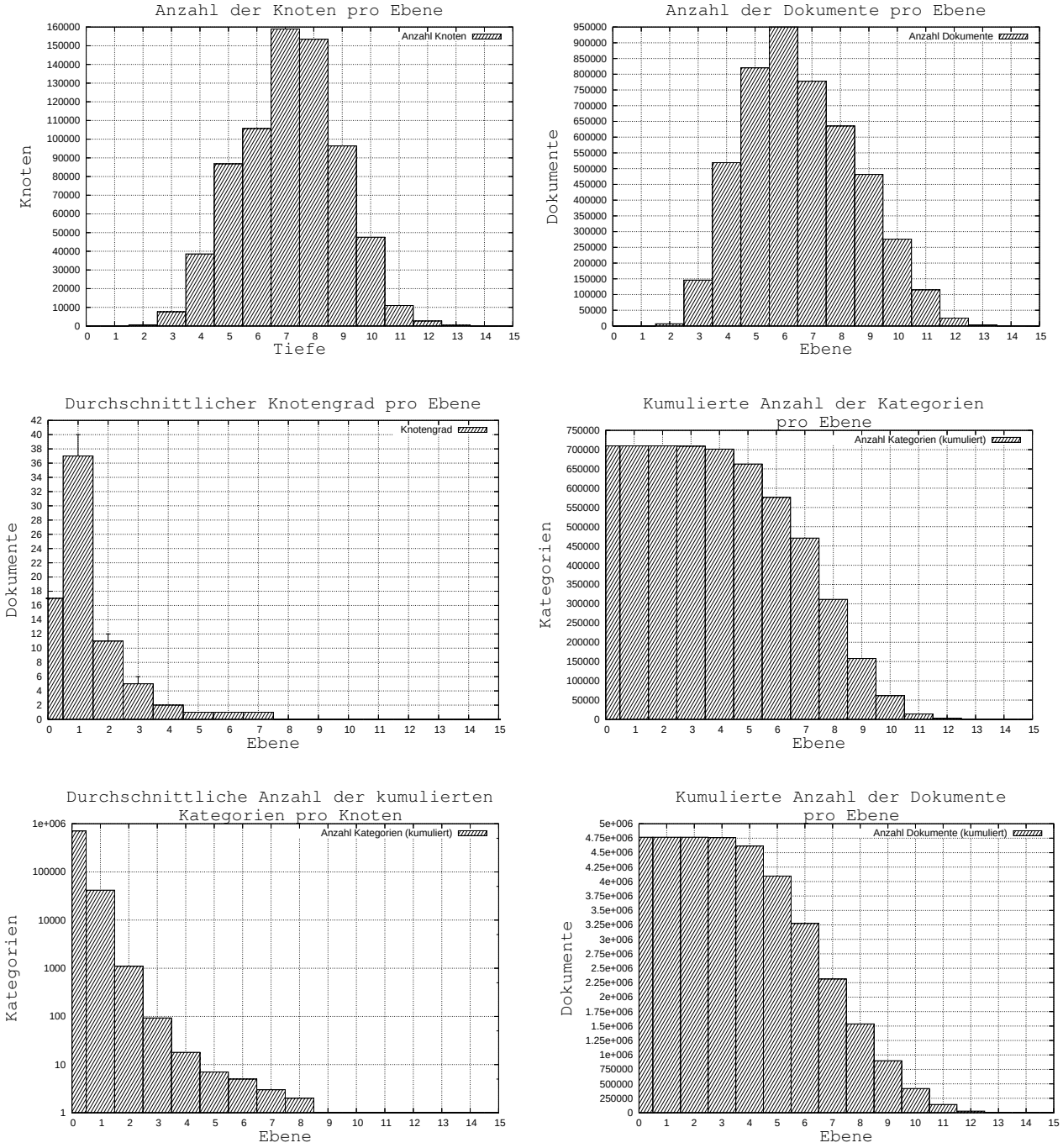


Abb. 2.4.: *Größenordnung und Beschaffenheit der DMoz-Ontologie. Es wird dargestellt: Die Anzahl der Knoten pro Ebene, die Anzahl der Dokumente pro Ebene, der durchschnittliche Knotengrad eines Knotens pro Ebene, die kumulierte Anzahl der Kategorien pro Ebene, die kumulierte Anzahl der Dokumente pro Ebene und pro Ebene die durchschnittliche kumulierte Anzahl der Kategorien pro Knoten.*

3. Textklassifikation

Textklassifikation ist ein vielfach motivierter Prozess. In den oft zitierten Arbeiten von Dunja Mladenić, Michael Granitzer, Fabrizio Sebastiani spricht man von Anwendungen wie Spamfilter, die Generierung von Metadaten, die Organisation von Dokumenten etc. [Mla98], [Gra03], [Seb02]. In dieser Arbeit wird Textklassifikation als Verfahren zur externen Topic Identification verwendet. Die Herausforderung liegt in der großen Anzahl an Kategorien, die als Basis für die Klassifikation dienen. Die folgenden Definitionen geben ein genaueres Verständnis.

Definition (Dokumente, Kategorien, Label). *Ein Dokument d_i besteht aus einer Menge von Wörtern. Es sei $D = \{d_1, \ldots, d_n\}$ eine Menge von Dokumenten, $C = \{c_1, \ldots, c_m\}$ eine Menge von Kategorien. Eine Funktion $f : D \to C$ heißt Kategorisierung von D bezüglich C. Für Kategorien gilt: $c_i \subseteq D$. Ein Label $l(c_i)$ ist ein Bezeichner (Deskriptor) einer Kategorie c_i. Entspricht c_i einem Knoten in einem Baum, so gehören weiterhin alle Label auf dem Pfad von c_i bis zur Wurzel zu diesem Knoten.*

Definition (Textklassifizierer). *Sei D eine Menge von Dokumenten und C eine Menge von Kategorien. Einem Tupel $(d_i, c_j) \in D \times C$ wird der Wert T zugeordnet, falls d_i in Kategorie c_j gehört, andernfalls F. Diese Zuordnung nennen wir Zielfunktion $\phi : D \times C \to \{T, F\}$. Ein Klassifizierer bzw. eine Hypothese ist eine approximierte Funktion $\hat{\phi} : D \times C \to \{T, F\}$, sodass ϕ und $\hat{\phi}$ sich möglichst ähnlich sind.*

3.1. Dokument Indexierung

Der Ansatz der Textklassifikation in dieser Arbeit ist, vorhandenes textuelles Wissen aus den Kategorien einer Ontologie zu nutzen, um Dokumente bzw. Dokumentengruppen in diese Ontologie einzuordnen. Die Idee ist, aus den implizit vorhandenen Informationen der Ontologie Klassifizierer zu generieren. Die Informationen werden extrahiert, indem Merkmale eines Textes ausgewählt werden. Merkmale, die in der Textklassifikation genutzt werden, können Worte

oder Wortgruppen sein, zu denen zusätzlich semantische, sowie statistische Informationen über ihr Vorkommen geführt werden können.

Im Folgenden beschreibt der Abschnitt Dokumentmodell, welche semantischen und statistischen Informationen eines Textes für die Erzeugung von Klassifizierern genutzt werden können. Der Abschnitt Merkmalsauswahl geht darauf ein, wie Merkmale ausgewählt werden. Das Ziel einer Merkmalauswahl ist, dass der erzeugte Klassifizierer wenig Rechenzeit und Arbeitsspeicher benötigt sowie ein möglichst genaues und richtiges Ergebnis liefert.

3.1.1. Dokumentmodell

Klassifizierer können rohe Dokumente nicht als Eingabe verwenden. Eine einheitliche Repräsentation der Dokumente ist erforderlich, die durch den Indexierungsvorgang erstellt wird. Die am Häufigsten verwendete Methode ist, gewichtete Merkmale in Vektoren zusammenzufassen. Alle vorkommenden Merkmale f_i bilden das Vokabular *Dic*. Für jedes Dokument d_i wird ein Vektor $\vec{d_i} = \left[w(f_1), \ldots, w(f_{|Dic|})\right]$ mit einer Gewichtsfunktion w berechnet. Für w gibt es unterschiedliche Ansätze. Zunächst wird festgelegt, um was es sich bei den Merkmalen handelt.

Werden Wörter als Merkmale verwendet, so spricht man auch von einer bag-of-words-Repräsentation. Es lassen sich einzelne Wörter, aber auch Wortgruppen verwenden. Die Wörter können durch einen Reduktionsprozess in ihre Grundform gebracht werden. Ein häufig verwendeter Algorithmus nennt sich *Porter Stemmer* [Por97]. Das Verfahren entfernt in mehreren Schritten Endungen, wie sie beispielsweise beim Plural auftreten. Der Algorithmus gewährleistet nicht, dass das Wort am Ende in seiner korrekten Grundform ist. Das Vorgehen ist dennoch für Textklassifikation legitim, da es deterministisch ist und immer zum gleichen Ergebnis führt.

In vielen Anwendungen werden Wörter, die unabhängig von konkreten Kategorien sind, so genannte Stoppwörter (the, is, he, she, it, ...), entfernt.

Eine Reihe von Gewichtsfunktionen, die sich für das Dokumentmodell gut eignen, werden an dieser Stelle definiert. Zwei einfache Funktion sind die Termhäufigkeit in einem Dokument d_i in **Gleichung 3.1** und die binäre Darstellung von Merkmalen in **Gleichung 3.2**.

$$w(f_k) = \#f_k \; mit \; f_k \in d_i \tag{3.1}$$

$$w(f_k) = \begin{cases} 1 \text{ wenn} f_k \in d_i \\ 0 \text{ (sonst)} \end{cases} \tag{3.2}$$

Auch die relative Häufigkeit eines Mermales f_k in Bezug auf die Häufigkeit aller Terme eines Dokumentes lässt sich als Gewichtsfunktion verwenden, siehe **Gleichung 3.3**.

$$w(f_k) = tf\,(f_k) = \frac{\#f_k}{\sum_j^{|Dic|} \#f_j} \; mit \; f_k, f_j \in d_i \tag{3.3}$$

Die inverse Dokumenthäufigkeit setzt die Gewichtung eines Merkmales in Relation mit der Anzahl der Dokumente, in denen das Merkmal vorkommt. Sei $|D|$ die Gesamtzahl der Dokumente im Verzeichnis und sei $|D_{f_k}|$ die Anzahl der Dokumente, in denen das Merkmal f_k vorkommt. Für die Berechnung der $tfidf$-Werte ergibt sich die **Gleichung 3.4**.

$$w(f_k) = tfidf\,(f_k) = tf\,(f_k) \cdot log\frac{|D|}{|D_{f_k}|} \tag{3.4}$$

In den Experimenten der Arbeit wird die einfache Termhäufigkeit in Gleichung 3.1 verwendet.

3.1.2. Merkmalsauswahl

Bei der Auswahl von Merkmalen gibt es unterschiedliche Vorgehensweisen. Das Einfachste ist die Auswahl aller vorkommenden Merkmale in das Vokabular. Das kann jedoch dazu führen, dass das Vokabular sehr groß wird und dann schwer handhabbar ist. Das Ziel einer guten Merkmalsauswahl ist, dass ein möglichst kleines Vokabular erstellt wird, ohne dass die Genauigkeit beim späteren Klassifizieren durch einen zu hohen Informationverlust verschlechtert wird.

Y. Yang und J. Pedersen vergleichen in ihrer Arbeit [YP97] verschiedene Kriterien für automatisierte Verfahren zur Merkmalsauswahl:

Document Frequency Thresholding (DF): Für jeden Term wird die Anzahl der Dokumente bestimmt, in denen er vorkommt. Die Terme, die über einen festgelegten Grenzwert liegen, werden für das Vokabular ausgewählt. Dies ist das einfachste Verfahren und es kommt mit einer linearen Laufzeit aus.

Information Gain (IG): Mit **Gleichung 3.5** wird $G(t)$ für jeden Term t berechnet. Es werden die Terme für das Vokabular übernommen, deren Werte oberhalb vorher festgelegten Grenze liegen. Seien die Kategorien $c_1 \dots c_m \in C$ alle Kategorien der Trainingsmenge. Die Abwesenheit eines Terms t ist mit $\bar{t}$ gekennzeichnet.

$$\begin{aligned} G(t) = & -\sum_{i=1}^{|C|} P(c_i) log P(c_i) \\ & + P(t) \sum_{i=1}^{|C|} P(c_i|t) log P(c_i|t) \\ & + P(\bar{t}) \sum_{i=1}^{|C|} P(c_i|\bar{t}) log P(c_i|\bar{t}) \end{aligned} \tag{3.5}$$

χ^2 (CHI): Die Funktion $\chi^2(t, c)$ berechnet die stochastische Unabhängigkeit eines Terms t und einer Kategorie c. Der Wert Null bedeutet, dass t und c stochastisch unabhängig sind. $\chi^2(t, c)$ wird mit **Gleichung 3.6** berechnet. Sei A die Anzahl der Dokumente in c mit t, sei B die Anzahl der Dokumente außerhalb von c mit t, sei C die Anzahl der Dokumente in c ohne t, sei D die Anzahl der Dokumente außerhalb von c ohne t, und sei N die Anzahl aller Dokumente.

$$\chi^2(t, c) = \frac{N(AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{3.6}$$

Für jeden Term wird das Maximum $\chi^2(t)$ bestimmt, indem die Werte jeder Kategorie $c_1 \dots c_m$ miteinander verglichen werden, siehe **Gleichung 3.7**.

$$\chi^2_{max}(t) = \max_{i=1}^{|C|} \chi^2(t, c_i) \tag{3.7}$$

Die Merkmale, deren Werte aus der Gleichung 3.7 über einen festgelegten Grenzwert liegen, werden in das Vokabular eingepflegt.

Yang und Pedersen haben die Genauigkeit von verschiedenen Klassifizierern mit automatisch generierten Vokabularen untersucht. Zum Vergleich hat man ein Vokabular gebildet, dass alle Merkmale verwendet, die in dem Korpus vorkommen. Die Verfahren IG und CHI verbessern die Genauigkeit der verwendeten Klassifizierer bei einer Merkmalreduzierung von 98%. Das Verfahren DF liefert eine vergleichbare Performanze bei 90% Merkmalreduzierung. Andere Kriterien wie Mutual Information oder Term Strength schneiden weniger gut ab [YP97]. In der Arbeit [RY02] werden weitere Versuche mit anderen Klassifizierern und anderen Korpora

gemacht, die diese Ergebnisse bestätigen.

Die Kombination von mehreren Kriterien zeigt sich als Verbesserung. In Experimenten schneidet die Kombination von CHI_max und IG gut ab [RY02]. Es werden für jeden Term t mit Gleichung 3.7 und Gleichung 3.5 die Werte für $\chi^2_{max}(t)$ und $G(t)$ berechnet und normalisiert, sodass ein Vergleich beider Werte möglich wurde. Anschließend wird der Term t mit dem höheren Wert in das Vokabular übernommen.

Diese Ergebnisse werde mit einem Klassifizierer (k-Nearest-Neighbour bzw. Linear Least Squares Fit) erzielt, der über alle vorhandenen Kategorien klassifiziert. Mladenić kommt bei der Verwendung von eines Naïve-Bayes-Klassifizierers und einer hierarchischen Klassifikation zu anderen Ergebnissen [Mla98]. Diese zeigen, dass sich das Verfahren Odds Ratio in **Gleichung 3.8** besser eignet.

$$\begin{aligned} OR(t, c_i) &= \frac{P(t|c_i)(1 - P(t|\bar{c_i}))}{(1 - P(t|c_i))P(t|\bar{c_i})} \\ OR_{sum}(t) &= \sum_{i=1}^{|C|} OR(t, c_i) \end{aligned} \tag{3.8}$$

Bei der Erstellung eines Vokabulares unterscheidet man zwischen einer lokalen und einer globalen Merkmalsauswahl:

Die globale Merkmalsauswahl nimmt als Eingabe für die oben genannten Kriterien, alle Terme aus dem gesamten Verzeichnis, um daraus ein globales Vokabular zu erstellen. Ein globales Vokabular ist ein Vokabular, in dem alle ausgewählten Terme mit einem eindeutigen Index gespeichert werden.

Teilt man die Kategorien in Gruppen, lässt sich eine lokale Merkmalsauswahl anwenden. Die lokale Merkmalsauswahl erstellt pro Gruppe ein eigenes Vokabular, wobei die Eingabe der Merkmalsauswahl aus den Termen der jeweiligen Kategorie besteht. Der Nachteil bei diesem Verfahren ist, dass man durch mehrere Vokabulare zusätzlichen Aufwand mit ihrer Verwaltung hat. Ein Dokument muss durch jedes erstellte Vokabular repräsentiert werden, damit es in einer Gruppe klassifiziert werden kann. Zusätzlich muss vor einer Klassifikation das richtige Vokabular ausgewählt werden.

Die lokale Merkmalsauswahl ist die Grundlage für die hierarchische Merkmalsauswahl. Grundvoraussetzung ist eine hierarchische Anordnung der Kategorien. Für die Kategorien, deren Nachfolger Blätter in der Hierarchie sind, wird mit der lokalen Merkmalsauswahl jeweils ein

Vokabular gebildet. Rekursiv wird aus den bereits erzeugten Vokabularen ein neues Vokabular für den Vorgänger erzeugt: Die Merkmale mit den besten Werten in den Merkmalsauswahlkriterien werden zu gleichen Teilen aus den Nachfolgerknoten ausgewählt.

3.2. Evaluierung

Das Evaluieren eines Klassifizierers soll Aufschluss geben, wie gut dieser ist. Das heißt in erster Linie, es ist festzustellen, wie viele Fehler er beim Klassifizieren macht.

Nimmt man eine Menge von Dokumenten, von denen man bereits weiß, zu welcher Kategorie sie gehören, lässt sich die Fehlerrate berechnen. Sie ist das Verhältnis der Anzahl falsch klassifizierter Dokumente zur Anzahl aller klassifizierten Dokumente. Bei Dokumenten, die bereits zum Training der Klassifizier verwendet werden, sind die zugehörigen Kategorien bekannt, jedoch stellt sich die Frage, ob diese zur Evaluierung geeignet sind. Die meisten Anwendungen ordnen unbekannte Dokumente in Kategorien ein. Sei ein unbekanntes Dokument ein Dokument, welches nicht bei der Erstellung eines Klassifiziers verwendet wird, analog dazu sei ein bekanntes Dokument ein Dokument, das zum Erstellen dient.

Für die vorangestellte Frage ist entscheidend, wie sehr die Fehlerrate von Trainingsdokumenten mit der Fehlerrate von neuen Dokumenten zusammenhängt. Schließlich ist es so, dass man keine Aussagen über einen Zusammenhang treffen kann. Die Fehlerrate von Dokumenten, die in Zukunft klassifiziert werden, lässt sich nur durch unbekannte Dokumente bestimmen. Diese Dokumente bilden die Testmenge. Test- und Trainingsmenge müssen also disjunkt sein. Im weiteren Verlauf der Arbeit wird die Erfolgsrate untersucht, die auch der Genauigkeit eines Klassifizierers entspricht.

Typische Maße zur Bewertung von Klassifizierern sind Recall, Precision und F, die in vielen Arbeiten verwendet werden [Gra03], [Mla98], [DC00]. Recall und Precision sind externe Maße, da sie Wissen über eine von Menschen gemachte Klassifikation benötigen, die als Referenz dient [SMW03]. F_1 wird aus Recall und Precision berechnet.

Sei D eine Menge von Dokumenten. Sei $c_1^*, \ldots, c_m^*$ eine Kategorisierung von D mit $c_i^* \subseteq D$, die von Menschen gemacht wurde. Ein Klassifizierer ordne D in die Kategorien $c_1, \ldots, c_n$ mit $c_i \subseteq D$. Der Recall einer Kategorisierung i in Bezug auf die Kategorie j ist $rec(i,j) = |c_j \cap c_i^*| \,/\, |c_i^*|$. Die Precision einer Kategorisierung i in Bezug auf die Kategorie j ist $prec(i,j) = |c_j \cap c_i^*| \,/\, |c_j|$.

Beide Werte liegen zwischen 0 und 1. Der Recall gibt an, ob die Dokumente aus der von Menschen gemachten Kategorie c_i^* in der erstellten Kategorie c_j vorkommen; sind alle Dokumente enthalten, so ist $rec(i,j) = 1$. Für die Precision gilt $prec(i,j) = 1$, wenn in Kategorie c_j nur Dokumente vorkommen, die sich auch in der tatsächlichen Kategorie c_i^* befinden; es spielt hierbei keine Rolle, ob c_j alle oder weniger Dokumente aus c_i^* enthält.

Strebt man einen möglichst hohen Recall an, so verschlechtert sich meist die Precision. Umgekehrt ist es leichter eine hohe Precision bei einem kleineren Recall zu erreichen. Deshalb ist es wichtig, bei einer Bewertung beide Maße anzugeben. Das F-Maß ist eine Kombination aus Recall und Precision:

$$F(i) = \frac{2}{1/prec(i) + 1/rec(i)}$$

3.3. Klassifikationstechniken

Die folgen Abschnitte beschreiben die Klassifikationstechniken:

- Lineare Klassifizierer
- Probabilistische Klassifizierer
- Klassifiziererensembles

3.3.1. Lineare Klassifizierer

Eine der wichtigsten Klassifiziererfamilien im Bereich der Textklassifikation ist die der linearen Klassifizierer. Auf Grund ihrer einfachen Natur besitzen sie einen gut fundierten theoretischen Hintergrund.

Ein linearer Klassifizierer ist eine Linearkombination aller Merkmale f_k eines Vokabulares Dic. Ein linearer Klassifizierer $h(\vec{d_i}) \rightarrow \{1, -1\}$, der entscheidet, ob ein Dokument d_i einer Kategorie c zugeordnet wird oder nicht, kann wie folgt beschrieben werden:

$$h(\vec{d_i}) = sign(\vec{w} \cdot \vec{d_i} - \Theta) = sign(\sum_{k=1}^{|Dic|} w_k \cdot d_{k,i} - \Theta)$$

Wobei w_k das Gewicht von Merkmal f_k ist. $d_{k,i}$ ist der Wert von Merkmal f_k in Dokument d_i, z. B. die Häufigkeit oder der tfidf-Wert. Somit ist jede Kategorie c_j durch einen Gewichtsvektor

$\vec{w_j}$ repräsentiert, der ein Dokument $\vec{d_i}$ einer Kategorie c_j zuordnet, wenn das innere Produkt $\vec{w_j} \cdot \vec{d_i}$ einen Grenzwert Θ_j überschreitet.

Ein Beispiel eines linearen Klassifizierers im zweidimensionalen Fall zeigt die **Abb. 3.1**. Die Gleichung $\vec{w} \cdot \vec{d} = \Theta$ definiert eine Hyperebene in einem $|Dic|$-dimensionalen Raum, die die positiven von den negativen Trainingsbeispielen trennt. Der Gewichtsvektor $\vec{w}$ ist der Normalvektor auf der Hyperebene. Der Abstand von der Hyperebene zum Ursprung ist

$$\frac{\theta}{||\vec{w}||}.$$

Der Abstand eines Trainingsbeispiels zur Hyperebene ist gegeben durch

$$r = \frac{\vec{w} \cdot \vec{d} - \Theta}{||\vec{w}||}.$$

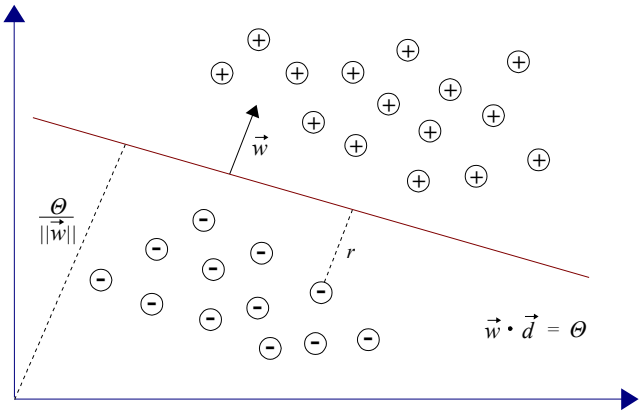


Abb. 3.1.: *Zweidimensionales Beispiel eines linearen Klassifizierers. Die Hyperebene separiert die negativen und die positiven Trainingsbeispiele. r ist der Abstand von der Hyperebene zu einem Trainingsbeispiel. $\vec{w}$ ist der Normalvektor auf der Hyperebene.*

Lineare Klassifizierer können auf verschiedene Arten lernen. Eine Methode ist der Perceptron-Algorithmus. Der Algorithmus kann verwendet werden, um einen linearen Klassifizierer im linear separierbaren Fall zu trainieren. Eine Alternative zu dem Perceptron-Algorithmus wird im Folgenden kurz beschrieben, die sogenannten Support Vector Machines.

Support Vector Machines (SVM)

Die Klassifikation mit einer SVM beruht auf dem Finden einer optimalen Hyperebene. Eine optimale Hyperebene trennt die positiven und die negativen Trainingsbeispielen so, dass

der Randabstand von der Hyperebene zum am nächsten gelegenen negativen und zum am nächsten gelegenen positiven Trainingsbeispiel maximal ist. In **Abb. 3.2** ist ein zweidimensionales, linear separierbares Beispiel dargestellt. Die Trainingsbeispiele, die der Hyperebene am nächsten liegen werden als Support-Vektoren bezeichnet. In dem linear separierbaren Fall kann die Maximierung des Randabstandes als Optimierungsproblem ausgedrückt werden:

$$minimiere\ \frac{1}{2}||\vec{w}||^2$$

Mit der Bedingung:

$$y_i(\vec{w} \cdot \vec{d_i} + \theta) \geq 1, \forall i \tag{3.9}$$

Der Vektor $\vec{w}$ ist der Normalvektor auf der Hyperebene. $\vec{d_i}$ ist die Wortvektorrepräsentation des i-ten Trainingsdokumentes. $y_i \in \{+1, -1\}$ besagt, ob das Dokument einer Kategorie zugeordnet ist oder nicht. **Ungleichung 3.9** fordert, dass alle Trainingsbeispiele korrekt klassifiziert wurden.

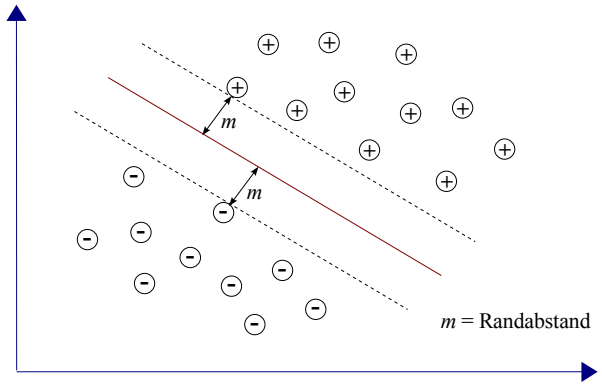


Abb. 3.2.: *Zweidimensionales Beispiel einer linearen SVM. Die Hyperebene separiert die negativen und die positiven Trainingsbeispiele mit maximalem Randabstand. Der Randabstand m ist der Abstand von der Hyperebene zum am nächsten gelegenen Trainingsbeispiel.*

Für detaillierte Informationen zu SVM's siehe [Bur98]. Für Informationen über SVM's angewandt im Bereich der Textklassifikation siehe [Joa98].

3.3.2. Probabilistische Klassifizierer

Probabilistische Klassifizierer entscheiden, ob ein Dokument $\vec{d}$ einer Kategorie $c_i \in C$ zugeordnet wird, indem sie die bedingte Wahrscheinlichkeit $P(c_i|\vec{d})$ berechnen. $P(c_i|\vec{d})$ ist die

Wahrscheinlichkeit, dass Kategorie c_i ausgewählt wird, wenn Dokument $\vec{d}$ gegeben ist. Die Zuordnung von Dokument $\vec{d}$ zu Kategorie c_i erfolgt, falls die Wahrscheinlichkeit über einem bestimmten Grenzwert liegt. Zur Berechnung der bedingten Wahrscheinlichkeiten wird das Theorem von Bayes benutzt:

$$P(c_i|\vec{d_j}) = \frac{P(c_i) \cdot P(\vec{d_j}|c_i)}{P(\vec{d_j})} \tag{3.10}$$

$P(c_i)$ ist die Wahrscheinlichkeit für Kategorie c_i. Die Wahrscheinlichkeit kann über die Anzahl der Dokumente bestimmt werden: $P(c_i) = \frac{|\{d_j|d_j \in c_i\}|}{|D|}$.
Aus dem Satz der totalen Wahrscheinlichkeit[1] folgt:

$$P(\vec{d_j}) = \sum_{i=1}^{|C|} P(c_i)P(\vec{d_j}|c_i) \tag{3.11}$$

Die Bestimmung von $P(\vec{d_j}|c_i)$ ist problematisch, da die Anzahl der möglichen Vektoren $\vec{d_j}$ für eine effiziente Berechnung zu hoch ist. Aus diesem Grund wird angenommen, dass alle Merkmale stochastisch unabhängig sind. Damit folgt:

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|Dic|} P(f_{kj}|c_i) \tag{3.12}$$

$P(f_{kj}|c_i)$ ist die bedingte Wahrscheinlichkeit, dass bei einer gegebenen Kategorie c_i Merkmal $f_{kj} \in d_j$ ausgewählt wird. Wegen der Einfachheit dieser Annahme wird der Ansatz Naïve Bayes genannt.

Durch Einsetzen der **Gleichung 3.12** in die **Gleichung 3.11** und die resultierende Gleichung dann in die **Gleichung 3.10** sowie durch Einsetzen der Gleichung 3.12 in die Gleichung 3.10 erhält man folgende Formel zur Berechnung der bedingten Wahrscheinlichkeiten:

$$P(c_i|\vec{d_j}) = \frac{P(c_i) \cdot \prod_{k=1}^{|Dic|} P(f_{kj}|c_i)}{\sum_{i=1}^{|C|} P(c_i) \cdot \prod_{k=1}^{|Dic|} P(f_{kj}|c_i)} \tag{3.13}$$

Trainiert wird ein Naïve-Bayes-Klassifizierer, indem die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$ für jedes Merkmal $f_k \in Dic$ und für jede Kategorie $c_i \in C$ auf Grund der Trainingsdaten bestimmt werden. Ein Ansatz, um die bedingten Wahrscheinlichkeiten zu berechnen ist

[1] Satz der totalen Wahrscheinlichkeit: Seien $A_1, ..., A_n$ disjunkte Ereignisse, mit $\sum_{i=1}^{n} P(A_i) = 1$, dann gilt für ein Ereignis B: $P(B) = \sum_{i=1}^{n} P(A_i)P(B|A_i)$

folgender:

$$P(f_k|c_i) = \frac{1 + FF(f_k, c_i)}{|Dic| + \sum_{j=1}^{|Dic|} FF(f_j, c_i)} \tag{3.14}$$

$FF(f_k, c_i)$ ist die Häufigkeit von Merkmal f_k in Dokumenten aus Kategorie c_i in der Trainingsmenge. Die Addition von 1 im Zähler und $|Dic|$ im Nenner ist als Laplace-Glättung bekannt. Ohne Laplace-Glättung würde ein Merkmal, das in keinem Dokument der Trainingsmege vorkommt, allerdings später in einem zu klassifizierenden Dokument d_j vorhanden ist, dafür sorgen, dass $P(\vec{d_j}|c_i) = 0$ ist, egal ob das Dokument zu dieser Kategorie gehört oder nicht.

Complement Naïve Bayes

Eine leichte Abänderung des Naïve-Bayes-Klassifizierers ist das Complement Naïve-Bayes-Verfahren. Die Wahrscheinlichkeit für das Vorkommen eines Merkmales f_i in Kategorie c_i wird geschätzt. Sie sei das Komplement der Wahrscheinlichkeit, dass f_i nicht in c_i ist. Die **Gleichung 3.12** wird demnach ersetzt durch:

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|Dic|} P(f_{kj}|c_i) = \prod_{k=1}^{|Dic|} (1 - P(f_{kj}|\bar{c_i})) \tag{3.15}$$

Bis auf diese Abweichung entspricht das Vorgehen dem aus 3.3.2.

Naïve Bayes Multinomial

Eine weitere Naïve-Bayes-Variante ist Naïve Bayes Multinomial ([KFPH04]). Wie schon bei Complement Naïve Bayes wird gegenüber Naïve Bayes die Wahrscheinlichkeit, dass Dokument d_j in c_i liegt (vgl. 3.12), mit einer anderen Gleichung bestimmt. Für Naïve Bayes Multinomial:

$$P(\vec{d_j}|c_i) = (\sum_{k=1}^{|Dic|} |f_{kj}|)! \prod_{k=1}^{|Dic|} \frac{P(f_{kj}|c_i)^{|f_{kj}|}}{|f_{kj}|!} \tag{3.16}$$

$P(f_{kj}|c_i)$ wird wie in **Gleichung 3.14** ermittelt. Die Terme $(\sum_{k=1}^{|Dic|} |f_{kj}|)!$ und $\prod_{k=1}^{|Dic|} |f_{kj}|!$ sind unabhängig von c_i. Das hat zur Folge, dass die Klassifikationsergebnisse nach Entfernen der

Terme gleich bleiben. Daraus resultiert mit α als Konstante:

$$P(\vec{d_j}|c_i) = \alpha \prod_{k=1}^{|Dic|} P(f_{kj}|c_i)^{|f_{kj}|} \tag{3.17}$$

3.3.3. Klassifiziererensembles

Dieser Abschnitt zeigt verschiedene Methoden zur Verbesserung der Klassifikationsgenauigkeit durch Ensembles. Ein Ensemble ist eine Menge von Klassifizierern mit den selben Zielkategorien. Ähnlich wie bei einem Expertenteam wird unter Berücksichtigung aller Meinungen ein Mehrheitsentscheid getroffen. Jedes Mitglied kann eine unterschiedliche Meinung haben. Verschiedene Arbeiten zeigen das Verbesserungspotential eines Ensembles [PNTK05], [Sch01], [Seb02]. Ensemblemethoden werden auch Klassifiziererkomittee oder Klassifiziererkombination genannt.

Die Klassifizierer in einem Ensemble werden als Basisklassifizierer bezeichnet. Jeder Basisklassifizierer erhält eine eigene Trainingsmenge. Wie ein Ensemble erzeugt wird, stellt **Abb. 3.3** schematisch dar. Das Ensembleverfahren erstellt für n Basisklassifizierer n verschiedene Trainingsmengen aus der ursprünglichen Trainingsmenge. Die Dokumente der ursprünglichen Trainingsmenge können —je nach Strategie— in mehreren Trainingsmengen vorkommen oder nur in höchstens einer. Das Ensembleverfahren trainiert jeden Basisklassifizierer mit seiner eigenen Trainingsmenge. Die Kombination aller Basisklassifzierer ist das Ensemble.

Nach [PNTK05] gibt es vier Methoden zur Ensemblekonstruktion. Zwei Methoden, die bezüglich des Experimentaufbaus dieser Arbeit geeignet wären, sind die folgenden:

- Manipulation der Kategoriedeskriptoren
- Manipulation der Trainingsmenge

Das Vorgehen der Manipulation der Kategoriedeskriptoren sieht folgendermaßen aus. Eine gleichverteilte Zufallsfunktion teilt die Deskriptoren der Kategorien in zwei disjunkte Mengen A_0 und A_1. Die Dokumente der Kategorien, deren Deskriptor in A_0 bzw. in A_1 ist, gehören zu A_0 bzw. A_1. Die neu aufgeteilten Dokumente trainieren einen binären Basisklassifizierer, der seine Eingabe A_0 oder A_1 zuordnet. Dieses Vorgehen erstellt mehrere binäre BasisKlassifizierer, indem es vor jeder Erstellung A_0 und A_1 zufällig erzeugt. Die binären Basisklassifizierer bilden das Ensemble. Das Ensemble trifft einen Mehrheitsentscheid, indem jeder Basisklassifizierer die Eingabe entweder A_0 oder A_1 zuordnet. Die Deskriptoren in A_0 bzw. A_1 erhalten für eine Zuordnung eine Stimme. Der Deskriptor mit den meisten Stimmen wird ausgewählt.

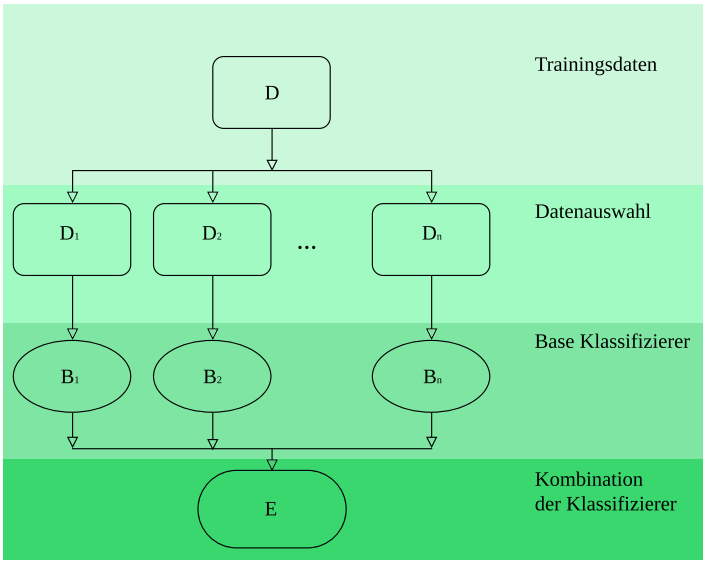


Abb. 3.3.: *Schema zur Erstellung eines Klassifiziererensembles. Die Trainingsmenge wird in n Mengen aufgeteilt, mit denen jeweils ein Basisklassifizierer trainiert wird. Die Kombination der Klassifizierer ergibt das Ensemble E.*

Beispiel: Das Beispiel bezieht sich auf vier Kategorien mit den Deskriptoren: „Biologie“, „Chemie“, „Physik“ und „Astronomie“. In drei Runden wird jeweils ein Basisklassifizierer B_i erstellt. Eine zufällige Aufteilung ordnet den Basisklassifizierer die Deskriptoren zu. Die Zuordnung ist in **Tab. 3.1** zu sehen. Das Ensembleverfahren trainiert jeden Basisklassifizierer mit den Dokumenten der zugeordneten Kategorien.

Basisklassifizierer	A_0	A_1
B_0	Biologie, Astronomie	Chemie, Physik
B_1	Biologie, Chemie	Astronomie, Physik
B_2	Chemie, Astronomie	Biologie, Physik

Tab. 3.1.: *Aufteilung der Deskriptoren für jeden Basisklassifizierer*

Sei d ein Dokument welches klassifiziert werden soll. Entscheide der Basisklassifizierer B_0, dass d zu B_0/A_0 gehört. Alle Deskriptoren in B_0/A_0 erhalten eine Stimme. Die Klassifizierer B_1 und B_2 treffen die Zuordnungen B_1/A_1 (Schritt 2) und B_2/A_0 (Schritt 3). Die Ergebnisse sind in **Tab. 3.2** zu sehen.

Das Ergebnis der Klassifikation durch das Ensemble ist „Astronomie“, da diese Kategorie die meisten Stimmen hat.

Die Ensembleerstellung durch Manipulation der Trainingsmenge trainiert Basisklassifizierer

Anzahl der Stimmen	Schritt 1 Klassifikation B_0	Schritt 2 Klassifikation B_1	Schritt 3 Klassifikation B_2
vote(Astronomie)	1	2	3
vote(Biologie)	1	1	1
vote(Chemie)	0	0	1
vote(Physik)	0	1	1

Tab. 3.2.: *Abstimmung der Basisklassifizierer*

mit zufällig erstellten Trainingsmengen. Die Methoden, die nach diesem Prinzip vorgehen, unterscheiden sich in

- der Erstellung der Trainingsmengen und in
- der Erstellung des Mehrheitsentscheids.

Zwei Verfahren, die sich bezüglich der Erstellung des Mehrheitsentscheids unterscheiden sind die Ensemblemethoden Bagging und Boosting. Ähnlich wie bei der Manipulation der Kategoriedeskriptoren, summiert das Bagging-Verfahren die Stimmen der Basisklassifizierer und wählt die Kategorie mit den meisten Stimmen. Das Boosting-Verfahren arbeitet mit Gewichtungen. Jeder Basisklassifizierer erhält durch den Boosting-Algorithmus eine Gewichtung mit der sein Klassifikationsergebnis im Mehrheitsentscheid berücksichtigt wird. Der Boosting-Algorithmus *AdaBoost* erstellt darüber hinaus Gewichtungen für die Elemente der Trainingsmenge [Sch01]. Schwierig zu klassifizierende Trainingsdokumente können durch eine hohen Gewichtung stärker berücksichtigt werden. Im Folgenden wird das AdaBoost-Verfahren genauer vorgestellt.

AdaBoost

Das AdaBoost-Verfahren ist iterativ und erzeugt in jedem Durchlauf einen Basisklassifizierer.

Alle Dokumente der ursprünglichen Trainingsmenge der Größe N haben initial die Gewichtung $w_j = 1/N$. Jeder Durchlauf wählt mit der Wahrscheinlichkeit w_j das Dokument d_j in die Trainingsmenge eines Basisklassifizierers.

Ein trainierter Basisklassifizierer hat eine Gewichtung α_i mit der sein Klassifikationsergebnis im Mehrheitsentscheid berücksichtigt wird. Die Berechnung der Gewichtung α_i beruht auf der

Fehlerrate ϵ_i des Klassifizierers B_i. Die Fehlerrate wird mit **Gleichung 3.18** bestimmt.

$$\epsilon_i = 1/N \sum_{j=0}^{N} w_j \cdot \begin{cases} 1, & \text{wenn } B_i(d_j) \neq c(d_j) \\ 0 & \text{(sonst)} \end{cases} \tag{3.18}$$

Sei $c(d_j)$ die Kategorie in der sich Dokument d_j befindet und sei $B_i(d_j)$ die Kategorie, die der Klassifizierer B_i dem Dokument d_j zugeordnet hat.

Die Gewichtung α_i des Klassifizierers B_i wird mit der **Gleichung 3.19** berechnet.

$$\alpha_i = \frac{1}{2} ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right) \tag{3.19}$$

Die **Abb. 3.4** zeigt, dass die Gewichtung eines Klassifizierers sehr hoch ist, wenn die Fehlerrate gering ausfällt. Die Entscheidung des Klassifizierers wird nicht berücksichtigt, wenn die Fehlerrate bei 50% liegt, da die Gewichtung dann gleich Null ist. Bei einer Fehlerrate zwischen 50% und 100% wird die Entscheidung des Klassifizierers als Gegenstimme im Ensemble gewertet, weil die Gewichtung einen negativen Wert annimmt.

Die Aktualisierung der Gewichtungen für die Trainingsdokumente wird im Durchlauf j mit **Gleichung 3.20** ermittelt.

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \cdot \begin{cases} exp^{-\alpha_j} & \text{wenn } B_i(d_j) = c(d_j) \\ exp^{\alpha_j} & \text{(sonst)} \end{cases} \tag{3.20}$$

Wobei Z_j der Normierung dient.

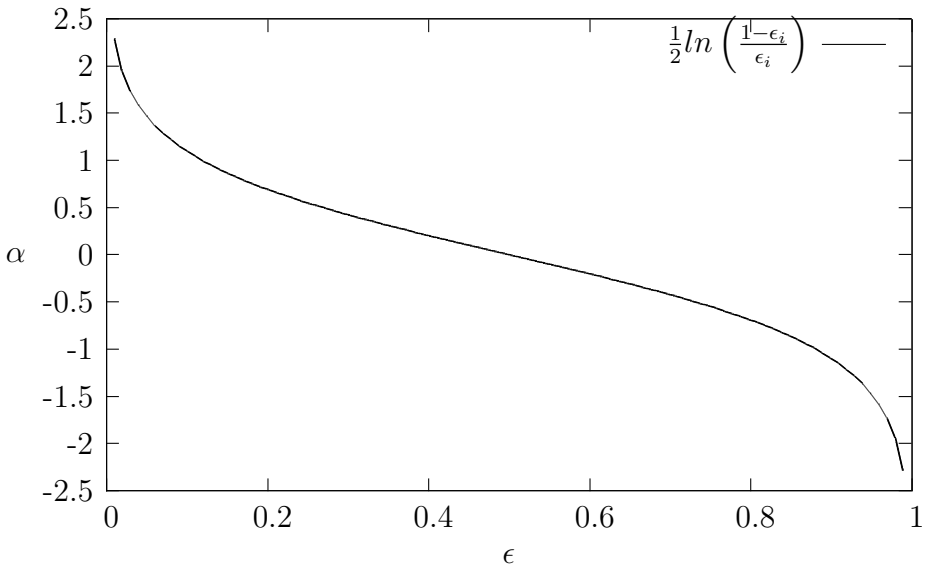


Abb. 3.4.: *AdaBoost Gewichtung α in Abhängigkeit der Fehlerrate ϵ*

4. Hierarchische Textklassifikation

Schwerpunkt dieser Arbeit ist die hierarchische Textklassifikation. Es werden in diesem Kapitel die besten Klassifikationstechniken vorgestellt. Hierbei wird sich am aktuellen Stand der Technik orientiert. Zum Vergleich wurde eine Übersicht früherer Arbeiten erstellt.

4.1. Methodik

In einem Verzeichnis mit einem hierarchischen Aufbau sind jedem Knoten der Hierarchie Dokumente zugeordnet. Die Knoten stellen Kategorien dar. Hierarchische Textklassifikation nutzt Struktur dieser Hierarchie, um daraus ein Klassifkationsmodell abzuleiten. Dies wird mit den folgenden zwei Definitionen erläutert.

Definition (Hierarchie). *Sei eine Hierarchie $H = (V, E)$ ein zusammenhängender, gerichteter azyklischer Graph mit der Kantenmenge $E \subseteq (V \times V)$ und der Knotenmenge V aus den Kategorien $c_0, \ldots, c_n$. Die Wurzel des Graphen ist ein Knoten, der keine eingehende Kante besitzt. Eine Kategorie c_s ist c_t untergeordnet, wenn c_t näher an der Wurzel liegt und c_s und c_t auf einem Pfad liegen; umgekehrt ist c_t der Kategorie c_s übergeordnet.*

Definition (Klassifikationsmodell). *Sei ein Klassifikationsmodell eine Struktur $KM = (K, R)$, wobei*

- *K eine Menge von Klassifizierern und*
- *$R \subseteq (K \times K)$ eine Menge von Relationen zwischen den Klassifizierern ist.*

Sei H eine Hierarchie mit den Kategorien $c_0, \ldots, c_n$ als Knotenmenge V und einer Kantenmenge E. Das hierarchisches Klassifikationsmodell hat für jede Kategorie c_i der Hierarchie einen Klassifizierer k_i, wenn gilt $(c_i, c_j) \in E$. Für die Relationen R im Klassifikationsmodell gilt $(k_i, k_j) \in R$, wenn $(c_i, c_j) \in E$.

*Der Klassifizierer k_i klassifiziert über die Kategorien c_j mit $(c_i, c_j) \in E$, wobei Klassifizierer k_i an Stelle von c_i ist. Siehe **Abb. 4.1**.*

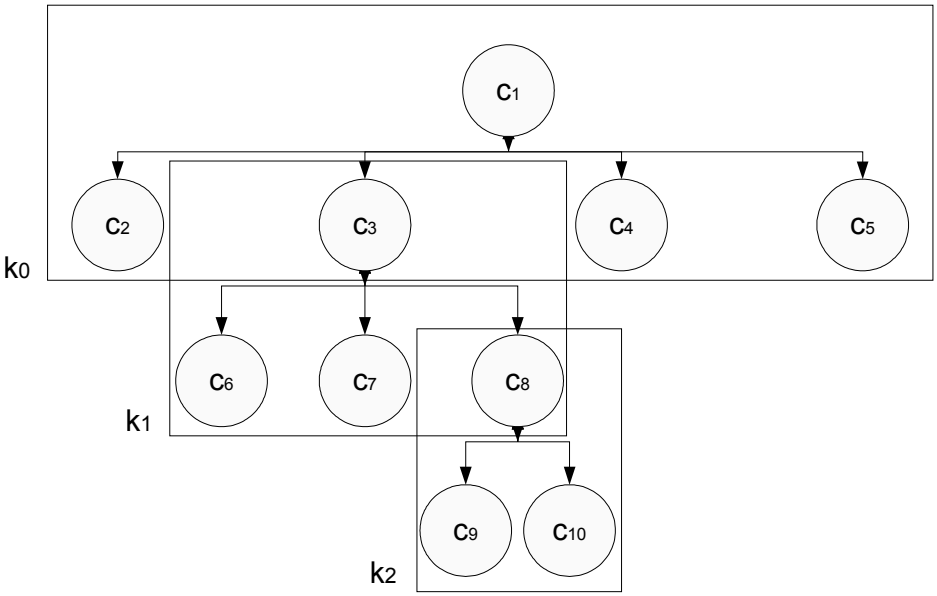


Abb. 4.1.: *Hierarchisches Klassifikationsmodell. Für jeden inneren Knoten der Hierarchie wird jeweils ein Klassifizierer erstellt. Ein Klassifizierer klassifiziert über seine Nachfolger in der Hierarchie. Ein Klassifizierer wird in der Abbildung mit k_n bezeichnet; eine Kategorie mit c_n. Die Relationen sind als Verbindungen dargestellt.*

Beim Klassifizieren wird das hierarchische Klassifikationsmodell rekursiv durchlaufen. Beginnend mit der Wurzel bestimmt ein lokaler Klassifizerer k_i zu welcher Kategorie c_j ein Dokument gehört. Der Klassifizierer k_j der Kategorie c_j klassifiziert im nächsten Schritt. So wird weiter vorgegangen bis die richtige Klasse gefunden wird. Dies wird als top-down-Ansatz bezeichnet.

Wird im hierarchischen Klassifizierungsprozess ein Fehler durch einen ungeeigneten Klassifizierer des Klassifikationsmodells gemacht, so wird der Fehler zum nächsten Klassifizierer propagiert. Der Fehler kann nicht mehr behoben werden, sodass die Fehlerrate erhöht wird.

Flache Textklassifikation basiert auf einer vereinfachten Form der Hierarchie, die nur aus der Wurzel und den Blättern besteht. Eine Hierarchie lässt sich in diese Form bringen, indem die inneren Knoten entfernt werden und jede Blattkategorie mit der Wurzel verbunden wird. Das Klassifikationsmodell entspricht dann einem einzelnen Klassifizierer in der Wurzel, der über alle Blätter klassifiziert.

Klassifizierer lassen sich in exklusive und nicht-exklusive Verfahren unterteilen. Ein exklusives Verfahren entscheidet mit „ja“ oder „nein“, ob ein Dokument zu einer Kategorie gehört. Nicht-exklusive Verfahren zeigen, mit welcher Wahrscheinlichkeit Dokumente Kategorien zugeordnet werden. In einer Hierarchie können nicht-exklusive Verfahren verwendet werden, um mehrere Pfade beim top-down-Vorgehen zu verfolgen. Eine Schranke bestimmt, bei welcher

Wahrscheinlichkeit ein Pfad weiterverfolgt wird. Als Ergebnis erhält man eine Labelmenge. In dieser Arbeit wird dies Multi-Pfad-Klassifikation genannt.

4.2. Frühere Arbeiten

In diesem Abschnitt werden frühere Arbeiten, die sich mit hierarchischer Textklassifikation beschäftigen, vorgestellt. Eine Übersicht der Arbeiten ist in **Tab. 4.1** zu sehen. Im Folgenden werden die Arbeiten und die erzielten Ergebnissen genauer beschrieben.

Autor	Klassifizierer	Datensatz	Größenordnung		
			Tiefe	# Kat.	# Dok.
Koller und Sahami [KS97]	Naïve Bayes	Reuters-22173 (modifiziert)	2	10	285
Dhillon et.al. [DMK02]	Naïve Bayes, Wort-Clustering	DMoz	3	49	≈1 700
McCallum et.al. [MRMN98]	Naïve Bayes, Shrinkage	Internetseiten, Newsgroups und Yahoo	2 2 -	71 20 264	6 440 20 000 14 831
Mladenic [Mla98]	Naïve Bayes	Yahoo	-	8 081	79 011
Dumais [DC00]	SVM	LookSmart-Dokumenten-sammlung	2	164	10 024
Sun und Lim [SL01]	SVM	Reuters-21578	2	7	978
Ruiz und Srinivasan [RS02]	Neuronale Netze (HME)	Ohsumed	4	103	50 216
Granitzer [Gra03]	BoosTexter, CenroidBooster	Reuters-Korpus, Band 1	4	102	18 252

Tab. 4.1.: *Übersicht früherer Arbeiten zur hierarchischen Textklassifikation. Aufgeführt sind die Autoren, die Klassifikationsverfahren, die Datensätze mit denen die Experimente durchgeführt werden und die Größenordnung der Testhierarchien. Die Größenordnung ist angegeben durch die Tiefe der Testhierarchien, die Anzahl der Kategorien und die Anzahl der Testdokumente. „-“ bedeutet, es wird ein kompletter Teilbaum benutzt. Die Tiefe des Teilbaumes ist nicht bekannt.*

Koller und Sahami verwenden in ihren Experimenten die Reuters-22173-Dokumentensammlung[1] [KS97]. Diese Dokumentensammlung besitzt keine vordefinierte hierarchische

[1] http://www.daviddlewis.com/resources/

Struktur. Daher konstruieren sie ein hierarchisches Modell durch das Auffinden von Labels, die andere Labels beinhalten, z. B. beinhaltet das Label „Sport“ die Labels „Fussball“ und „Tennis“. Die größte so erstellte Hierarchie besitzt Tiefe 2. Die Hierarchie besteht aus der Wurzel, aus drei Kategorien in Tiefe 1 und sechs Kategorien in Tiefe 2. Alle Dokumente befinden sich in den Blattkategorien der Hierarchie. Die Anzahl der Dokumente beträgt 939. Die Dokumente sind als boolesche Wortvektoren repräsentiert. Eine Merkmalauswahl findet durch die Zipf-Mandelbrot-Regel statt. Zur Klassifikation wird ein Naïve-Bayes-Verfahren benutzt. Die Klassifizierer werden mit 70% der Dokumente trainiert (654 Dokumente) und mit 30% der Dokumente gestestet (285 Dokumente). Bei der Klassifikation wird die Hierarchie top-down durchlaufen, wobei an jedem Knoten immer nur die beste Nachfolgekategorie gewählt wird. In den Experimenten können so etwa 80% bis 90% der Testdokumente korrekt klassifiziert werden. Zum Vergleich testen Koller und Sahami das Verfahren auf einem flachen Modell. Hierzu klassifizieren sie nur über die Blattkategorien der Hierarchie und ignorieren somit die hierarchische Struktur. Es wird festgestellt, dass nur bei einer geringen Anzahl von Merkmalen, etwa 10 bis 20, das hierarchische Verfahren ein besseres Klassifizierungsergebnis gegenüber dem flachen Verfahren liefert.

Dhillon et.al. benutzen Wort-Clustering zur hierarchischen Textklassifikation [DMK02]. Das Hauptaugenmerk ihrer Arbeit liegt auf der Merkmalauswahl. Sie verwenden ein Wort-Clustering-Verfahren, um die Menge der Merkmale zu reduzieren. Als hierarchischer Klassifizierer wird Naïve Bayes benutzt. Die Experimente führen sie an der DMoz-Open-Directory durch. Als Testhierarchie benutzen sie einen Teil des Science-Zweiges. Die Testhierarchie besitzt Tiefe 3 und besteht aus 49 Kategorien. Die Anzahl der Dokumente beträgt über 5 000. Alle Dokumente befinden sich in den Blattkategorien der Hierarchie. Es wird ein 1/3-2/3-Split verwendet. Das heißt 1/3 der Dokumente wird zum Testen verwendet und 2/3 der Dokumente zum Training. Für jede innere Kategorie erstellen sie einen Naïve-Bayes-Klassifizierer und trainieren ihn mit den Dokumenten der darunter liegenden Kategorien. Die Klassifikation erfolgte top-down und endete, wenn ein Blatt erreicht wird. Es stellt sich heraus, dass die Anwendung des Wort-Clustering-Verfahrens zur Merkmalauswahl gute Klassifikationsergebnisse liefert. Auch bei einer geringen Anzahl von Merkmalen. Es wird eine Erfolgsrate von ca. 93% erreicht. Bei einem Vergleich des Wort-Clustering-Verfahrens mit Merkmalauswahlverfahren, wie mutual information und χ^2 kann gezeigt werden, dass das Wort-Clustering-Verfahren den anderen Ansätzen überlegen ist, vor allem bei einer geringen Anzahl von Trainingsdokumenten. Des Weiteren vergleichen sie ihren hierarchischen Ansatz mit einem flachen Verfahren. Hierbei liegt die Erfolgsrate des hierarchischen Ansatzes 6% über der des flachen Verfahrens.

McCallum et.al. benutzen in ihrer Arbeit ein hierarchisches Trainingsverfahren, um einen fla-

chen Naïve-Bayes-Klassifizierer zu trainieren [MRMN98]. Das Trainingsverfahren basiert auf der statistischen Methode Shrinkage, siehe Abschnitt 4.3. Shrinkage wird beim Training eingesetzt, um die Schätzung der A-Posteriori-Wahrscheinlichkeiten für die Blattkategorien zu verbessern. Die Idee ist, die Schätzung für eine Kategorie durch lineare Interpolation der bedingten Wahrscheinlichkeiten aller Vorgängerkategorien zu glätten. Dies ist vor allem für Kategorien sinnvoll, die wenige Trainigsdokumente enthalten. Die Experimente führen sie an drei verschiedenen Datensätzen durch. Der erste Datensatz besteht aus hierarchisch geordneten Webseiten. Die Testhierarchie besitzt Tiefe 2 und beinhaltet 71 Kategorien sowie insgesamt 6 440 Dokumente. Der zweite Datensatz setzte sich aus 20 Newsgroups zusammen, die in eine hierarchische Struktur gebracht werden. Die so erzeugte Hierarchie besitzt Tiefe 2, beinhaltet 20 Kategorien und 20 000 Dokumente. Der dritte Datensatz ist die Yahoo-Ontologie[2]. Als Testhierarchie wird der komplette Science-Teilbaum benutzt. Die Testhierarchie besteht aus 264 Kategorien und beinhaltet 14 831 Dokumente. Zur Merkmalauswahl wird Mutual Information benutzt. Der Klassifizierer wird mit der Hälfte der Dokumente trainiert und mit der anderen Hälfte getestet. Es kann gezeigt werden, dass Shrinkage die Klassifikationsgenauigkeit um bis zu 29% verbessert. Vor allem bei einer geringen Anzahl von Trainingsdokumenten ist Shrinkage einem flachen Trainingsverfahren überlegen.

Auch Mladenić benutzt in ihrer Arbeit Naïve Bayes zur hierarchischen Textklassifikation [Mla98]. Für jeden Knoten in der Hierarchie wird ein Naïve-Bayes-Klassifizierer trainiert. Trainingsdokumente einer Kategorie gelten hier als positive Trainingsbeispiele für die Kategorie selbst und für alle Nachfolgekategorien. Gewichtet werden die Trainingsbeispiele auf Grund ihrer Position in der Hierarchie. Die Trainingsbeispiele werden benutzt, um die Parameter für den Klassifizierer zu bestimmen. Ein Klassifizierer berechnet die Postiori-Wahrscheinlichkeit für die Nachfolgekategorien. Die Nachfolgekategorien, für die der Klassifizierer eine Wahrscheinlichkeit größer als 0,95 bestimmt, werden ausgewählt. Mladenić führt ihre Experimente an Teilbäumen der Yahoo-Ontologie durch. Die größte Testhierarchie besteht aus 8 081 Kategorien und beinhaltet 79 011 Dokumente. Als Methode zur Merkmalauswahl wird Odds Ratio verwendet. Sie erzielt mit ihrer Methode gute Klassifikationsergebnisse. Auch hier wird berichtet, dass die Klassifikationsgenauigkeit stark von der Qualität und der Anzahl der Merkmale abhängt. Die besten Resultate werden mit einer geringen Anzahl von Merkmalen erzielt, mit etwa 50 bis 100.

Dumais trainiert für jede Kategorie der Hierarchie eine lineare SVM [DC00]. Als Testhierarchie wird ein Teilbaum der LookSmart-Dokumentensammlung[3] mit Tiefe 2 benutzt. In Tiefe

[2] http://www.yahoo.com
[3] http://www.looksmart.com

1 befinden sich 13 Kategorien und in Tiefe 2 150 Kategorien. Insgesamt umfasst die Testhierarchie 60 102 Dokumente. Von diesen Dokumenten werden 50 087 für das Training und 10 024 für den Test verwendet. Die Dokumente sind als boolesche Wortvektoren repräsentiert. Eine Merkmalauswahl findet durch Mutual Information statt. Für ein Testdokument werden die Wahrscheinlichkeiten für die Kategorien in Tiefe 1 berechnet sowie die Wahrscheinlichkeiten für die Kategorien in Tiefe 2. Dumais benutzt zwei verschiedene Verfahren, um die Wahrscheinlichkeiten der Kategorien in Tiefe 1 mit den Wahrscheinlichkeiten der Kategorien in Tiefe 2 zu kombinieren. Das erste Verfahren ist eine boolesche Bewertungsfunktion. Ein Testdokument wird einer Kategorie zugeordnet, wenn die Wahrscheinlichkeiten aus Tiefe 1 und die aus Tiefe 2 einen gegebenen Grenzwert überschreiten. In dem zweiten Verfahren wird die Wahrscheinlichkeiten aus Tiefe 1 und die aus Tiefe 2 multipliziert. Ein Testdokument wird einer Kategorie zugeordnet, falls das Produkt einen gegebenen Grenzwert überschreitet. Hierbei ist es möglich, ein Dokument in mehrere Kategorien zu klassifizieren. In den Experimenten wird ein Dokument im Durchschnitt in 1,07 Kategorien der Tiefe 1 klassifiziert und in 1,2 Kategorien der Tiefe 2. Die Evaluierung erfolgt nach dem F-Maß. Der gesamte F_1-Wert über alle Kategorien in Tiefe 1 ist 0,572 und für die Kategorien in Tiefe 2 0,495. Bei einem Vergleich des hierarchischen Verfahrens mit einem flachen Verfahren konnten leichte Vorteile des hierarchischen Ansatzes festgestellt werden.

Auch Sun und Lim verwenden wie Dumais zur Klassifikation eine SVM [SL01]. Sie trainieren für jede Kategorie der Hierarchie zwei binäre Klassifizierer, einen lokalen Klassifizierer und einen Klassifizierer für den darunter liegenden Teilbaum. Der lokale Klassifizierer entscheidet, ob ein Testdokument in die Kategorie gehört oder nicht. Falls das Testdokument in die Kategorie gehört, wird es ihr zugeordnet. Falls nicht, entscheidet der zweite Klassifizierer, zu welcher Nachfolgekategorie das Dokument weiter propagiert wird. Mit diesem Verfahren lassen sich Dokumente auch inneren Kategorien zuordnen. Sun und Lim führen ihre Experimente auf der Reuters-21578-Dokumentensammlung durch. Die größte Testhierarchie besitzt Tiefe 1 und besteht aus sieben Kategorien. Die Dokumente sind als boolesche Wortvektoren repräsentiert. Eine Merkmalauswahl findet nicht statt. Die Klassifizierer werden mit ca. 7 000 Dokumenten trainiert und mit ca. 1 000 Dokumenten getestet. Eine gute Erfolgsrate kann vor allem dann erzielt werden, wenn die Klassifizierer mit einer großen Anzahl von Dokumenten trainiert werden. Ein Vergleich des Verfahrens mit einem flachen Ansatz findet nicht statt.

Ruiz und Srinivasan benutzen zur hierarchischen Textklassifikation ein so genanntes Hierarchical-Mixture-of-Experts-Modell (HME) [RS02]. Hierbei handelt es sich um ein neuronales Netzwerk bestehend aus so genannten Gating-Netzwerken und Expert-Netzwerken. Ähnlich wie bei dem Ansatz von Sun und Lim wird hier nach einem divide-and-conquer-

Prinzip vorgegangen. Jeder inneren Kategorie der Hierarchie wird ein Gating-Netzwerk und ein Expert-Netzwerk zugewiesen. Für Blattkategorien ist ein Expert-Netzwerk ausreichend. Ein Testdokument wird nun top-down in die Hierarchie einsortiert. Dabei entscheidet an jeder Kategorie das Expert-Netzwerk, ob das Dokument in die aktuelle Kategorie gehört. Zusätzlich entscheidet an jeder inneren Kategorie das Gating-Netzwerk, ob das Dokument in eine Nachfolgekategorie gehört. Wenn dies der Falls ist, wird das Dokument entsprechend nach unten propagiert. Das Experiment wird auf einem Teil der Ohsumed-Dokumentensammlung durchgeführt. Die Testhierarchie besitzt Tiefe 4 und besteht aus 103 Kategorien. Die Kategorien beinhalten 233 455 Dokumente, wovon 183 229 zum Training und 50 216 für den Test benutzt wurden. Auch Ruiz und Srinivasan testen ihr Verfahren zum Vergleich auf einem flachen Modell. Hierbei erzielt das hierarchische Verfahren eine um bis zu 8% höhere Erfolgsrate. Des Weiteren vergleichen sie ihren Ansatz mit einem standard Rocchio-Klassifizierer. Hierbei kann kein Vorteil festgestellt werden, beide Verfahren liefern etwa die gleichen Klassifikationsergebnisse.

Granitzer benutzt ein Boosting-Verfahren zur hierarchischen Textklassifikation, den so genannten CentroidBooster-Algorithmus [Gra03]. Zu Vergleichszwecken wird zusätzlich eine SVM und ein flacher Rocchio-Klassifizierer benutzt. Die größte Hierarchie, an der er seine Experimente durchführt, ist ein Teilbaum des Reuters-Korpus[4], Band 1. Die Testhierarchie besitzt Tiefe 4 und besteht aus 102 Kategorien. Die Klassifizierer werden mit 19 798 Dokumenten trainiert und mit 18 252 Dokumenten getestet. Zur Merkmalauswahl wird die Zipf-Mandelbrot-Regel verwendet. Alle Merkmale, die nicht in mindestens drei Dokumenten vorkommen, werden entfernt. Gewichtet werden die Merkmale nach dem tfidf-Verfahren. In den Experimenten erzielt der CentroidBooster-Algorithmus im Durchschnitt leicht bessere Ergebnisse als die SVM. Im Vergleich der hierarchischen Ansätze mit dem flachen Rocchio-Klassifizierer zeigt sich, dass sowohl der CentroidBooster-Algorithmus als auch die SVM dem flachen Rocchio Ansatz überlegen sind.

4.3. Shrinkage

Das statistische Verfahren *Shrinkage* kann genutzt werden, um das Training eines Naïve-Bayes-Klassifizierers, siehe Abschnitt 3.3.2, zu optimieren. McCullum et.al. haben gezeigt, dass Shrinkage die Klassifikationsgenauigkeit verbessert [MRMN98]. Vor allem bei großen Hierarchien mit wenigen Trainingsdokumenten pro Kategorie.

[4] http://about.reuters.com/researchandstandards/corpus

Basierend auf Shrinkage wird die bedingte Wahrscheinlichkeit $\widehat{P}(f_k|c_i)$ berechnet. $\widehat{P}(f_k|c_i)$ ist die mit Shrinkage berechnete Wahrscheinlichkeit für ein Merkmal $f_k \in Dic$ bei gegebener Blattkategorie $c_i \in C$. Die Berechnung der bedingten Wahrscheinlichkeit erfolgt durch lineare Interpolation der bedingten Wahrscheinlichkeiten aller Kategorien auf dem Pfad von Kategorie c_i bis zur Wurzel der Hierarchie. Die Gewichte für die Interpolation werden durch eine Form der Erwartungswertmaximierung gelernt.

Ein Beispiel für die Berechnung der bedingten Wahrscheinlichkeit $\widehat{P}(f_k|c_i)$ ist in **Abb. 4.2** dargestellt. Zunächst wird die Hierarchie erweitert, indem eine zusätzliche, virtuelle Kategorie über der Wurzel hinzugefügt wird. Die bedingte Wahrscheinlichkeit der virtuellen Kategorie ist über alle Merkmale gleichverteilt und dient als Glättung. Für jedes Merkmal $f_k \in Dic$ und jede Kategorie $c_i \in C$ werden nun die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$ wie folgt berechnet. Die Berechnung erfolgt wie in Abschnitt 3.3.2, **Gleichung 3.14** nur ohne Laplace-Glättung:

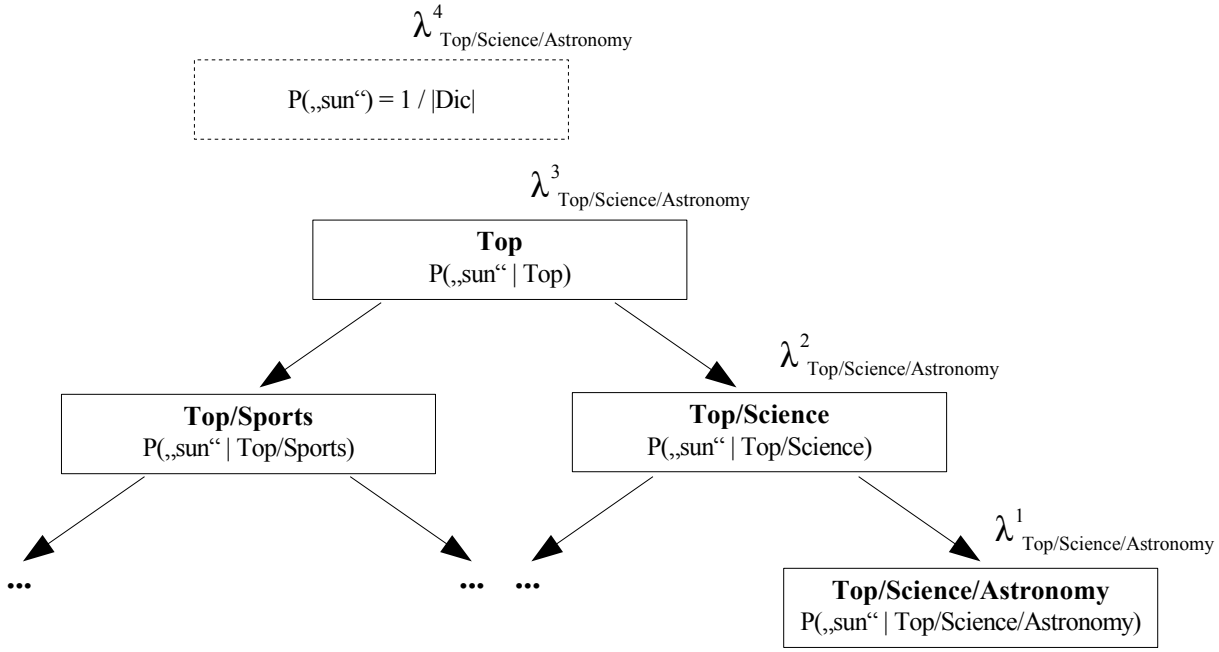
$$P(f_k|c_i) = \frac{FF(f_k, c_i)}{\sum_{j=1}^{|Dic|} FF(f_j, c_i)} \tag{4.1}$$

Nun wird Shrinkage angewendet, um die Genauigkeit der bedingten Wahrscheinlichkeiten zu verbessern. Für jedes Merkmal $f_k \in Dic$ und für jede Blattkategorie der Hierarchie c_i werden die bedingten Wahrscheinlichkeiten wie folgt neu berechnet:

$$\widehat{P}(f_k|c_i) = \lambda_i^1 P(f_k|c_1)_i + \lambda_i^2 P(f_k|c_2)_i + \cdots + \lambda_i^j P(f_k|c_j)_i$$

Wobei $P(f_k|c_1)_i = P(f_k|c_i)$ die bedingte Wahrscheinlichkeit der Blattkategorie c_i ist. $P(f_k|c_j)_i$ ist die gleichverteilte bedingte Wahrscheinlichkeit der virtuellen Kategorie über der Wurzel. Es gilt: $P(f_k|c_j)_i = 1/|Dic|$, für alle $f_k \in Dic$. Die bedingten Wahrscheinlichkeiten $P(f_k|c_x)_i$, mit $x = 2, \ldots, j-1$, werden wie in **Gleichung 4.1** berechnet. Es muss jedoch sichergestellt sein, dass die bedingten Wahrscheinlichkeiten entlang des gegebenen Pfades unabhängig sind. Aus diesem Grund werden vor der Berechnung der bedingten Wahrscheinlichkeit für eine Elternkategorie die Trainingsdaten der Kindkategorie von denen der Elternkategorie abgezogen. Der Faktor λ_i^x, mit $x = 1, \ldots, j$, ist das Gewicht von Kategorie c_x. Es gilt: $\sum_x \lambda_i^x = 1$. Die λ_i's werden nach **Algorithmus 3** bestimmt[5]. Hierzu wird ein Teilmenge der Trainingsdokumente aus der Kategorie c_x entfernt. Diese Dokumente bilden die so genannte held-out-Menge H_i.

[5]Quelle: [MRMN98]



$$\hat{P}(\text{„sun“} \mid \text{Top/Science/Astronomy}) = \lambda^{1}_{\text{Top/Science/Astronomy}} P(\text{„sun“} \mid \text{Top/Science/Astronomy}) + \lambda^{2}_{\text{Top/Science/Astronomy}} P(\text{„sun“} \mid \text{Top/Science}) + \lambda^{3}_{\text{Top/Science/Astronomy}} P(\text{„sun“} \mid \text{Top}) + \lambda^{4}_{\text{Top/Science/Astronomy}} P(\text{„sun“})$$

Abb. 4.2.: *Dargestellt ist ein Ausschnitt aus der DMoz-Ontologie, mit einer zusätzlichen, virtuellen Kategorie über der Wurzel. Darunter ist die Shrinkage-basierte bedingte Wahrscheinlichkeit des Wortes „sun“, bei gegebener DMoz-Kategorie Top/Science/Astronomy dargestellt. Die bedingte Wahrscheinlichkeit wird ausgedrückt, als gewichtete Summe der bedingten Wahrscheinlichkeiten aller Kategorien auf dem Pfad von der Blattkategorie Top/Science/Astronomy zu der Wurzel und weiter zu der virtuellen Kategorie. λ^i ist das Gewicht von Kategorie i. Die Quelle: [MRMN98]*

Algorithmus 3 : Bestimmung der Gewichte λ_i

Initialisierung: Setze die λ_i's auf einen beliebigen Wert, z. B. $\lambda_i^x = \frac{1}{j}$.
Iteriere:
(1) Bestimme den Grad dafür, wie gut die bedingte Wahrscheinlichkeit $P(f_k|c_i)$ voraussagt, dass f_k in der held-out-Menge H_i von c_i ist:

$$\beta_i^x = \sum_{f_k \in H_i} \frac{\lambda_i^x P(f_k|c_x)_i}{\sum_{m=1}^{j} \lambda_i^m P(f_k|c_m)_i}$$

(2) Berechne die neuen Gewichte, indem die β's normiert werden:

$$\lambda_i^x = \frac{\beta_i^x}{\sum_{m=1}^{j} \beta_i^m}$$

Terminierung: Bei Konvergenz der λ_i's. Dieses wird in den meisten Fällen nach etwa einem Dutzend Iterationen erreicht.

4.4. Subsumption

Vor dem Hintergrund, die Fehlerrate eines Klassifikationsergebnisses zu senken, sind Algorithmen entstanden, die verschiedene Klassifizierer kombinieren. Die unterschiedlichen Methoden, wie Klassifizierer kombiniert werden, führen dazu, dass die Ergebnisse statistischer Untersuchungen sehr unterschiedlich ausfallen. Das resultierende Verhalten bei einer Änderung eines Klassifikationsverfahrens ist nicht bzw. schwer vorhersagbar.

Der Gedanke bei den Klassifiziererensembles ist, das Klassifikationsergebnis zu optimieren, indem ein Mehrheitsentscheid getroffen wird, vgl. Abschnitt 3.3.3. Selbige Idee findet sich bei dem Subsumptionsverfahren. Die Voraussetztung ist eine hierarchische Anordnung von Kategorien, zu denen eine Menge von Klassifizierern zuordnet werden. Das Verfahren kombiniert die Ergebnisse von Klassifizierern, indem die gemeinsamen Vorgängerkategorien der Ergebnisse ausgewählt werden. Dieses Verfahren wird im Folgenden motiviert und beschrieben.

4.4.1. Idee

Die Klassifikationsaufgabe wird mit Zunahme der Zielkategorien schwieriger. Das hat zur Folge, dass auch die Fehlerrate mit steigender Anzahl der Kategorien wächst. Die Fehlerrate steigt auch, je mehr Klassifikationen in einer Hierarchie gemacht werden, also je tiefer und damit spezieller klassifiziert wird.

Ausgangspunkt sei ein hierarchisches Klassifikationsmodell, in dem jeder Knoten der Hierarchie aus einem Klassifizierer besteht. Bei jedem Schritt in einem top-down-Verfahren ist die Möglichkeit gegeben, falsch zu klassifizieren. Wenn ein Dokument in den ersten drei Tiefen einer Hierarchie richtig und dann falsch klassifiziert wird, ist das Gesamtergebnis nicht korrekt, obwohl durch die vorherigen Zuordnungen bereits der richtige Teilbaum ausgewählt wird.

Die Strategie, möglichst speziell zu klassifizierern, um so einen hohen Informationsgehalt zu gewinnen, führt zu einer höheren Fehlerrate. Umgekehrt verhält es sich so, dass bei einer weniger tiefen Zuordnung, die Genauigkeit nicht durch weitere Klassifikationen beeinflusst wird. Der Wissensgewinn fällt kleiner aus, unter der Annahme, dass alle weiteren Schritte zu einem korrekten Ergebnis geführt hätten.

4.4.2. Vorgehen

Subsumption ist der Prozess, eine Kategorie mit einer allgemeineren Kategorie zu umfassen. Die Kategorien, die einem Dokument durch einen oder mehreren Klassifizierern zugeordnet werden, lassen sich subsumieren.

In einer Hierarchie können die abstrakteren Vorgängerkategorien für eine Kategorie c_i ermittelt werden. Die Sortierung der Vorgängerkategorien nach ihrem Abstraktionsgrad ist durch ihre Anordnung in der Hierarchie gegeben. Sei die allgemeinste Kategorie die Wurzel in der Hierarchie. Seien die Vorgängerkategorien $PAR(c_i)$ die Kategorien, die auf dem Pfad von der Wurzel bis zu der Kategorie c_i liegen. Für zwei Kategorien c_i und c_j ist die Schnittmenge $SUB = PAR(c_i) \cup PAR(c_j)$ die Menge der gemeinsamen, abstrakten Vorgängerkategorien und damit eine Menge von Subsumptionen.

Die Ergebnisse von Klassifizierern lassen sich subsumieren. Dabei legt eine Subsumptionsgrenze fest, wie abstrakt die Kategorien einer Subsumption sein dürfen. In einer Hierarchie ist diese Grenze eine festgelegte Tiefe, die die speziellste Kategorie der Subsumption überschreiten muss. Ansonsten sei in diesem Verfahren das Ergebnis der Subsumption die Kategorie eines einzelnen Klassifizierers, die auf der Subsumptionsgrenze liegt. Die Subsumption ist als korrekt zu bewerten, wenn die speziellste Kategorie der Subsumption ein Vorgänger der tatsächlichen Kategorie ist.

Die **Abb. 4.3** gibt Beispiele für korrekte und nicht korrekte Subsumptionen mit einer Subsumptionsgrenze (subsumption level). Im den oberen Teilbäumen der Abbildung ist die Subsumptionsgrenze in der Tiefe 1. Im der oberen, linken Zeichnung ist die speziellste Kategorie in

der Subsumption in den Vorgängerknoten der Zielkategorie enthalten. Die Subsumptionsgrenze wird dabei überschritten. Das Subsumptionsergebnis ist korrekt. Im oberen, rechten Bild wird die Subsumptionsgrenze nicht überschritten. Die Kategorie des ersten Klassifizierers in der Tiefe 1 wird bewertet, wobei das Subsumptionsergebnis korrekt ist. In der unteren, linken Zeichnung ist die Subsumption nicht korrekt. Um die Subsumptionsgrenze zu überschreiten, wählt das Subsumptionsverfahren das Ergebnis des ersten Klassifizierers bis zur Tiefe 2. Die speziellste Kategorie des Ergebnisses ist somit nicht mehr unter den Vorgängern der Zielkategorie. In der letzten Zeichnung fallen die Ergebnisse beider Klassifizierer zusammen, jedoch ist das Ergebnis nicht korrekt.

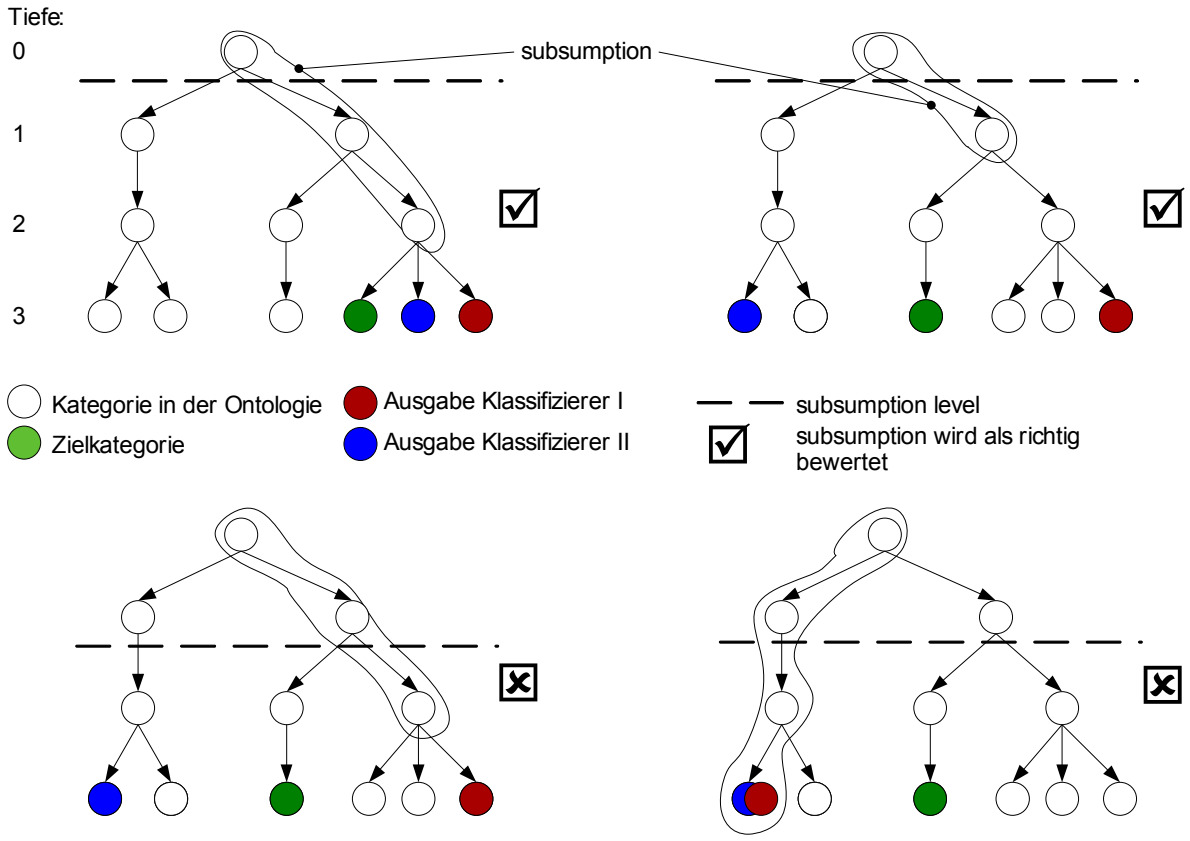


Abb. 4.3.: *Subsumption: Übersicht und Beispiele. In den oberen Zeichnungen ist die Subsumptionsgrenze eins und in den unteren Zeichnungen ist sie zwei. Das vorgestellte Verfahren bewertet die oberen Subsumptionen als korrekt. Die unteren Subsumptionen sind nicht korrekt, da die speziellste (also tiefste) Kategorie der Subsumption nicht zu den Vorgängern der Zielkategorie gehört.*

4.5. Multi-Pfad-Klassifikation

Der Ausgangspunkt einer Multi-Pfad-Klassifikation ist eine Hierarchie mit einem Klassifizierer für jede Kategorie. Ein Dokument wird beginnend bei der Wurzel top-down in die Hierarchie einsortiert. Jeder Klassifizierer klassifiziert über seine direkten Nachfolgekategorien. Allerdings wird das Dokument nicht nur zur besten Nachfolgekategorie weiter propagiert (Single-Pfad), sondern zu allen Nachfolgekategorien, deren Klassifikationsergebnis über einem bestimmten Grenzwert liegt. Es ist also möglich, mehrere Pfade zu verfolgen. Das Ergebnis der Klassifikation ist eine Menge von Kategorien. Ein Beispiel einer Multi-Pfad-Klassifikation ist in **Abb. 4.4** dargestellt.

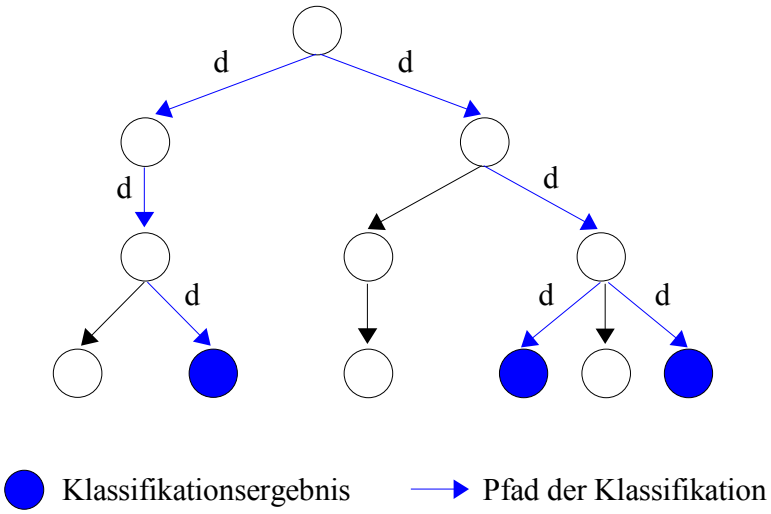


Abb. 4.4.: *Beispiel einer Multi-Pfad-Klassifikation. Ein Dokument d wird top-down in die Hierarchie einsortiert, dabei werden mehrere Pfade verfolgt. Das Ergebnis ist eine Menge von Kategorien.*

Der Grenzwert legt fest, ab welcher Wahrscheinlichkeit ein Pfad verfolgt wird. Ist der Grenzwert beispielsweise null, würden alle Pfade verfolgt. Die Ergebnismenge würde somit alle Blattkategorien enthalten. Folglich läge die Erfolgsrate bei 100%. Wird der Grenzwert jedoch zu hoch gewählt, kann es passieren, dass kein Pfad verfolgt wird. In diesem Fall kann das Dokument nicht klassifiziert werden. Die Wahl des Grenzwertes ist also von entscheidender Bedeutung, um ein gutes Klassifikationergbnis zu erzielen.

5. Software zur Evaluierung

Im Rahmen dieser Arbeit wurden verschiedene Softwarewerkzeuge und ein Framework entwickelt. In diesem Kapitel wird zunächst die Architektur und die Funktionsweise des Frameworkes erläutert. Das Framework beinhaltet drei Klassifikationsmodelle. Diese Klassifikationsmodelle werden in den darauf folgenden Abschnitten im Detail beschrieben. Des Weiteren stellt das Framework ein Evaluierungsmodul bereit. Dieses Modul bietet die Möglichkeit, auf den Klassifikationsmodellen Experimente auszuführen und diese auszuwerten. Das Evaluierungsmodul wird in Abschnitt 5.5 beschrieben. Zum Indexieren der Dokumente wurde ein weiteres Werkzeug entwickelt. Die Funktionsweise des Indexierers wird in Abschnitt 5.6 erläutert. Um die Benutzung der Softwarewerkzeuge zu vereinfachen wurde eine grafische Oberfläche erstellt. Die Funktionalität der Benutzungsschnittstelle wird in Abschnitt 5.7 beschrieben. Außerdem wird eine Anleitung zur Verwendung der Benutzungsschnittstelle gegeben.

5.1. Framework

Das Framework ermöglicht eine einheitliche Evaluierung von verschiedenen Klassifikationsmodellen. Außerdem stellt es verschiedene Typen von Klassifizierern und Trainingsverfahren bereit. In **Abb. 5.1** ist das Framework als UML-Klassendiagramm dargestellt.

Das Interface `ClassificationModel` dient als Schnittstelle zwischen der Evaluierung und den Klassifikationsmodellen. Aus der Klasse `Evaluation` wird ein Experiment mit einem bestimmten Klassifikationsmodell gestartet und die Ergebnisse der Klassifikation ausgewertet. Über das Interface `ClassificationModel` können verschiedene Klassifikationsmodelle angebunden werden. Jedes Klassifikationsmodell muss die folgenden Methoden des Interfaces implementieren:

- `generateClassificationModel`:
  Ein Aufruf dieser Methode bewirkt, dass das entsprechende Klassifikationsmodell generiert wird. Die Generierung beinhaltet das Erzeugen der Klassifizierer sowie deren

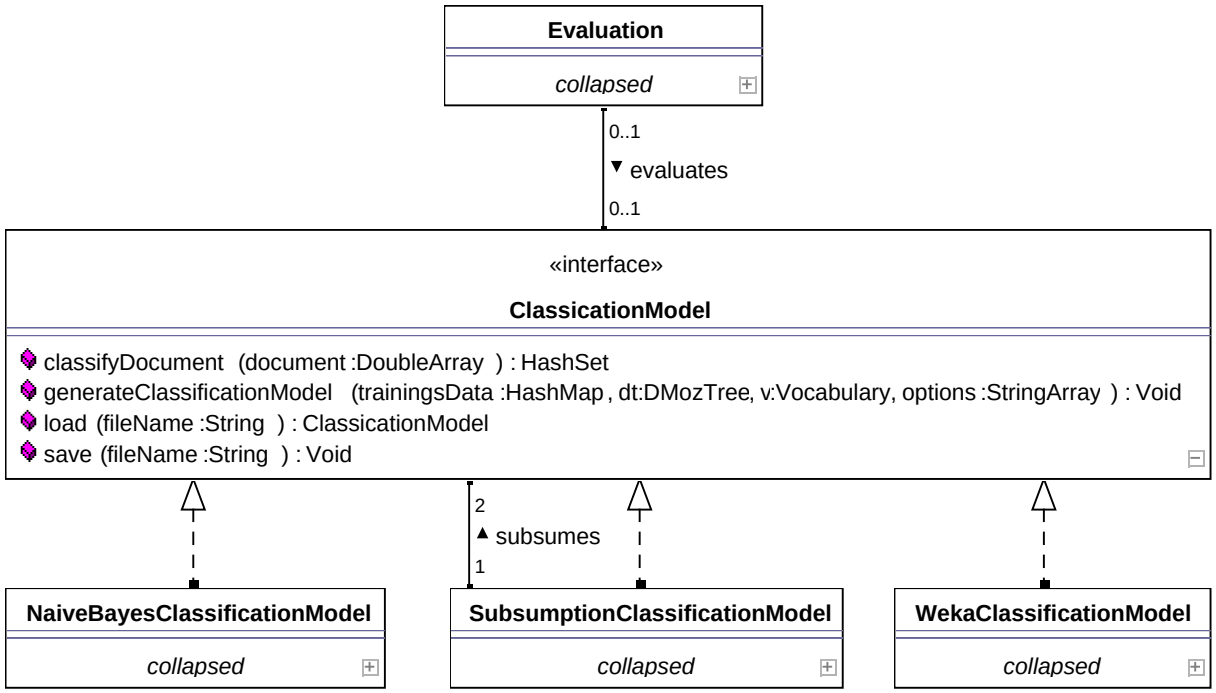


Abb. 5.1.: *UML-Klassendiagramm des Frameworkes. Dargestellt sind die wichtigsten Klassen bzw. Interfaces und die wichtigsten Methoden. Die Klasse* `Evaluation` *besitzt ein Interface* `ClassificationModel`*. Darüber sind drei Klassifikationsmodelle angebunden, die das Interface implementieren:* `NaiveBayesClassificationModel`*,* `SubsumptionClassificationModel` *und* `WekaClassificationModel`*.*

Training. Als erster Parameter werden die Trainingsdaten übergeben. Hierbei handelt es sich um eine Menge von Dokumenten, denen ein Label zugeordnet ist. Der zweite Parameter ist die Repräsentation der Hierarchie als `DMozTree`. Als dritter Parameter wird das Vokabular übergeben. Der letzte Parameter bietet die Möglichkeit verschiedene Optionen zu setzen, beispielsweise den konkreten Typ der Klassifizierer oder die Trainingsmethode.

- `classifyDocument`:
  Diese Methode startet die Klassifikation eines Dokumentes auf dem Klassifikationsmodell. Als Parameter wird das zu klassifizierende Dokument übergeben. Der Rückgabewert ist das Ergebnis der Klassifikation. Hierbei handelt es sich um eine Menge von Labels.

- `save`:
  Die Methode speichert das komplette Klassifikationsmodell im Dateisystem ab. Als Parameter wird der Dateiname übergeben.

- `load`:
  Die statische Methode `load` lädt ein zuvor gespeichertes Klassifikationsmodell in den Speicher. Als Parameter wird der Dateiname des gespeicherten Klassifikationsmodelles übergeben. Der Rückgabewert der Methode ist eine Referenz auf das geladene Klassifikationsmodell.

Im Rahmen dieser Arbeit wurden drei Klassifikationsmodelle implementiert: Das Naïve-Bayes-Klassifikationsmodell, das Subsumption-Klassifikationsmodell und das Weka-Klassifikationsmodell. Alle sind über das Interface `ClassificationModel` angebunden und stellen verschiedene Typen von Klassifizierern bzw. Trainingsmethoden bereit. In den folgenden Abschnitten wird auf die konkrete Implementierung der Klassifikationsmodelle eingegangen sowie auf die Funktionsweise der Evaluierung und die Verwendung der grafischen Benutzungsschnittstelle.

5.2. Naïve-Bayes-Klassifikationsmodell

Das Klassifikationsmodell stellt verschiedene, selbst implementierte Naïve-Bayes-Klassifizierer sowie entsprechende Trainingsverfahren zur Verfügung. In **Abb. 5.2** ist ein UML-Klassendiagramm des Klassifikationsmodelles dargestellt. Die zentrale Klasse `NaiveBayesClassification` implementiert das Framework-Interface `ClassificationModel`. Des Weiteren stellt die Klasse zwei Interfaces bereit: Das `NBCTrainingInterface` und das `NaiveBayesClassifierInterface`. Über diese Schnittstellen können verschiedene Trainingsverfahren bzw. Klassifizierer angebunden werden.

Im Folgenden werden die beiden Interfaces beschrieben. Danach wird die Implementierung der Methoden des Framework-Interfaces `ClassificationModel` erläutert. Anschließend wird die allgemeine Funktionsweise des Klassifikationsmodelles beschrieben. In Abschnitt 5.2.1 wird auf die Trainingsverfahren eingegangen, die das Klassifikationsmodell bereitstellt. Die von diesem Klassifikationsmodell bereitgestellten Naïve-Bayes-Klassifizierer werden in Abschnitt 5.2.2 beschrieben.

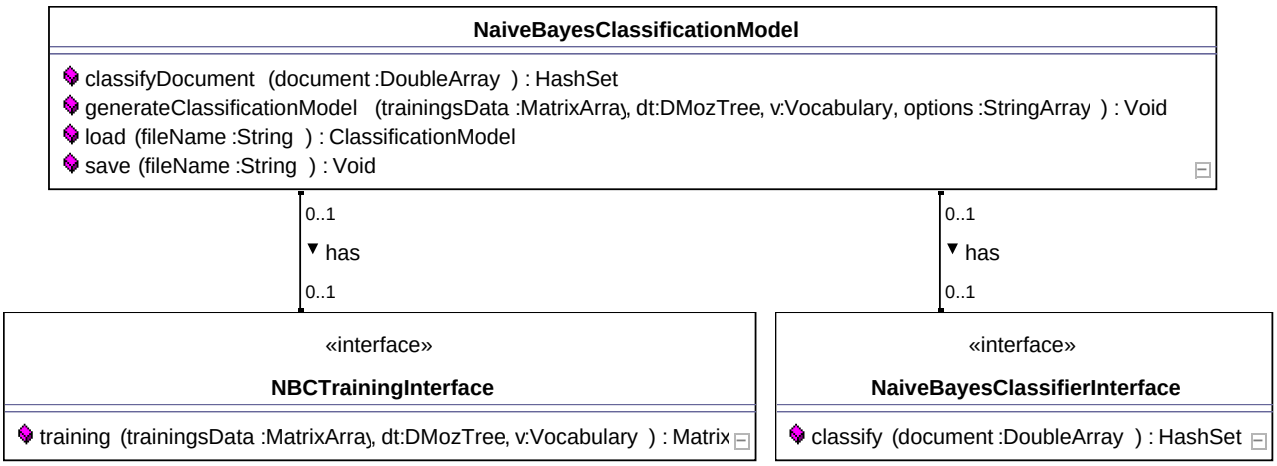


Abb. 5.2.: *UML-Klassendiagramm des Naïve-Bayes-Klassifikationsmodelles. Dargestellt sind die wichtigsten Klassen bzw. Interfaces und die wichtigsten Methoden. Die Klasse* `NaiveBayesClassificationModel` *besitzt zwei Interfaces: Das* `NBCTrainingInterface` *und das* `NaiveBayesClassifierInterface`*. Über die Interfaces können verschiedene Trainingsverfahren bzw. Klassifizierer angebunden werden.*

Interfaces

- `NBCTrainingInterface`:
  Das Interface `NBCTrainingInterface` dient als Schnittstelle für verschiedene Implementierungen von Trainingsverfahren für Naïve-Bayes-Klassifizierer.

  Das Interface definiert die Methode `training`. Diese Methode initiiert das Training der Klassifizierer. Als erster Parameter werden die Trainingsdokumente übergeben. Der zweite Parameter ist das Vokabular und der dritte die Repräsentation der Hierarchie als `DMozTree`. Der Rückgabewert der Methode ist eine Matrix, die das Trainingsergebnis enthält. Hier wird eine $|Dic| \times |C|$-Matrix verlangt, die die entsprechenden bedingten Wahrscheinlichkeiten enthält.

- `NaiveBayesClassifierInterface`:
  Das zweite Interface `NaiveBayesClassifierInterface` stellt eine Schnittstelle für Implementierungen von verschiedenen Varianten von Naïve-Bayes-Klassifizierern zur Verfügung.

  Das Interface definiert die Methode `classify`. Diese Methode startet die Klassifikation eines Dokumentes. Der einzige Parameter ist das zu klassifizierende Dokument. Der Rückgabewert ist das Ergebnis der Klassifikation. Hierbei handelt es sich um eine Menge mit keinem, einem oder mehreren Labels.

Methoden

Die Klasse `NaiveBayesClassificationModel` implementiert das Framework-Interface `ClassificationModel`, vgl. Abschnitt 5.1. Die konkreten Implementierungen der Methoden werden nun erläutert.

- `generateClassificationModel`:
  Die Methode `generateClassificationModel` ist als Aktivitätendiagramm in **Abb. 5.3** dargestellt. Zunächst wird ein Trainingsverfahren instanziiert. Über das Interface `NBCTrainingInterface` wird auf dem Objekt `trainer` die Methode `training` aufgerufen. Dies liefert eine Matrix zurück, die die Trainingsergebnisse für die gesamte Hierarchie enthält. Nun wird mit dem Trainingsergebnis ein Klassifizierer für die Wurzel der Hierarchie instanziiert. Damit ist die Generierung des Modelles abgeschlossen.

- `classifyDocument`:
  In **Abb. 5.4** ist die Methode `classifyDocument` als Aktivitätendiagramm dargestellt. Das Objekt `rootClassifier` ist der bei der Generierung des Modelles instanziierten Klassifizierer für die Wurzel der Hierarchie. Über das Interface `NaiveBayesClassifierInterface` wird auf dem Objekt `rootClassifier` die Methode `classify` aufgerufen. Als Parameter wird das zu klassifizierende Dokument übergeben. Die Methode liefert das Klassifikationsergebnis zurück. Das Klassifikationsergebnis ist auch der Rückgabewert der Methode `classifyDocument`.

- `save`:
  Die Methode `save` speichert die Klasse `NaiveBayesClassificationModel` mit den Inhalten aller Klassenvariablen in die als Parameter übergebene Datei.

- `load`:
  Die Methode `load` lädt das Klassifikationsmodell aus einer als Parameter übergebenen Datei in den Speicher. Der Rückgabewert der Methode ist ein Objekt vom Typ `NaiveBayesClassificationModel`.

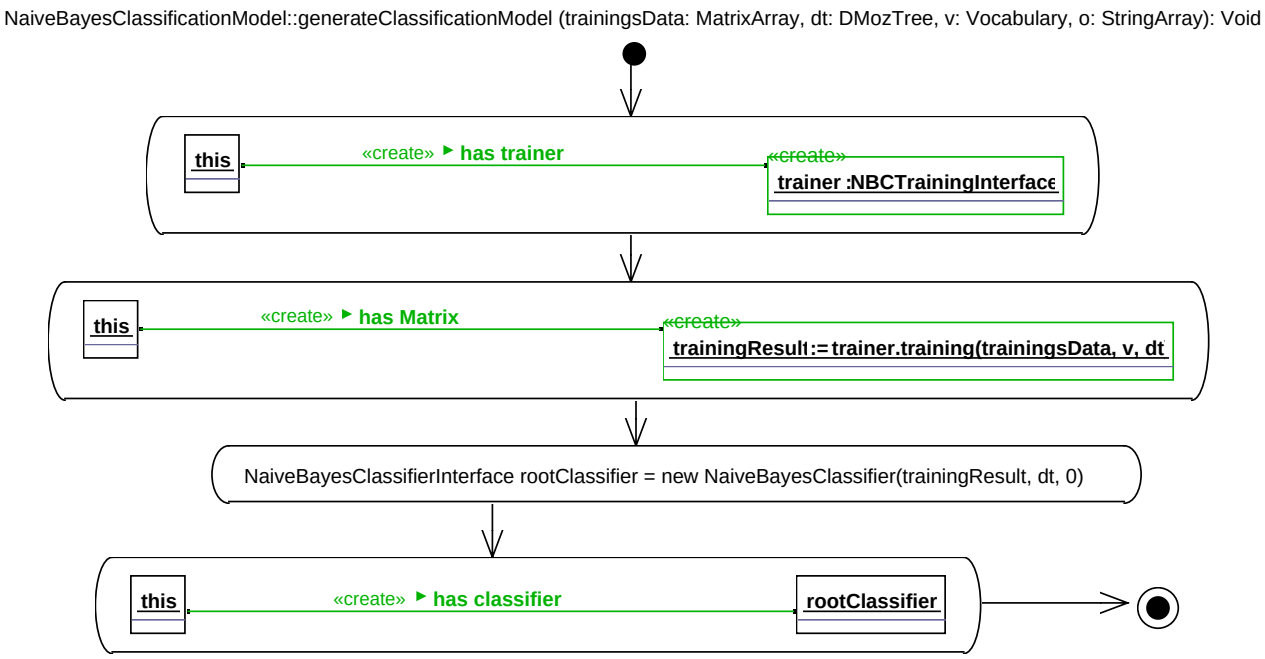


Abb. 5.3.: *Aktivitätendiagramm der Methode* `generateClassificationModel` *der Klasse* `NaiveBayesClassificationModel`*. Auf dem Objekt* `trainer` *wird über das Interface* `NBCTrainingInterface` *die Methode* `training` *aufgerufen. Diese liefert die Matrix mit den bedingten Wahrscheinlichkeiten zurück. Danach wird ein Klassifizierer für die Wurzel der Hierarchie erzeugt.*

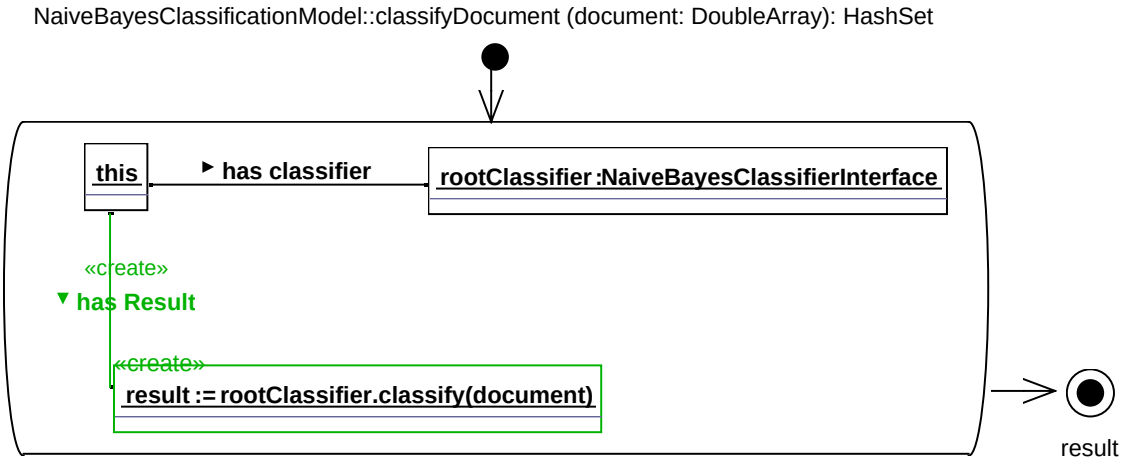


Abb. 5.4.: *Aktivitätendiagramm für die Methode* `classifyDocument` *der Klasse* `NaiveBayesClassificationModel`*. Auf dem Klassifizierer für die Wurzel wird über das Interface* `NaiveBayesClassifierInterface` *die Methode* `classify` *aufgerufen. Als Parameter wird das zu klassifizierende Dokument übergeben. Der Rückgabewert ist das Klassifikationsergebnis.*

Funktionsweise

Das Naïve-Bayes-Klassifikationsmodell ist ein hierarchisches Klassifikationsmodell. Das heißt für jede Kategorie wird ein eigener Klassifizierer trainiert, der über seine Nachfolgekategorien klassifiziert, vgl. Abschnitt 4.1.

Trainiert wird ein Naïve-Bayes-Klassifizierer, indem die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$ berechnet werden, für alle Merkmale $f_k \in Dic$ und für alle Kategorien $c_i \in C$, vgl. Abschnitt 3.3.2. Die Methode `generateClassificationModel` der Klasse `NaiveBayesClassificationModel` startet ein Trainingsverfahren. Das Training liefert eine $|Dic| \times |C|$-Matrix zurück. Die Matrix enthält die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$, für alle Merkmale $f_k \in Dic$ und für alle Kategorien $c_i \in C$. In dieser Matrix sind alle benötigten Informationen vorhanden, um an jedem inneren Knoten der Hierarchie einen Naïve-Bayes-Klassifizierer zu erstellen. Ein Beispiel hierfür ist in **Abb. 5.5** dargestellt.

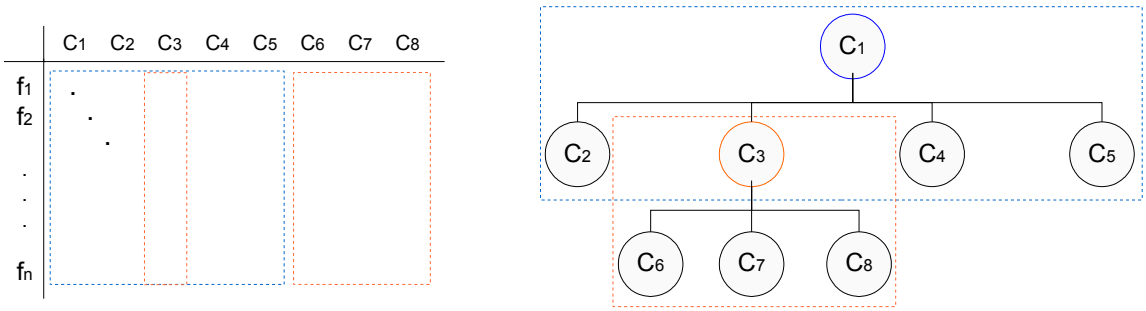


Abb. 5.5.: *Dargestellt ist die Zuordnung der Trainingsdaten zu den Klassifizierern. Links ist die $|Dic| \times |C|$-Matrix dargestellt. Die Matrix wird durch das Training erstellt und enthält die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$. Auf der rechten Seite ist die entsprechende Hierarchie zu sehen, mit zwei Klassifizieren an den inneren Kategorien c_1 und c_3.*

Nun wird für die Wurzel der Hierarchie ein Klassifizierer mit den entsprechenden Trainingsdaten instanziiert. Das Klassifikationsmodell besteht also aus der Matrix mit den bedingten Wahrscheinlichkeiten und einem Klassifizierer an der Wurzel der Hierarchie.

Ein Aufruf der Methode `classifyDocument` startet nun die Klassifikation. Auf dem Klassifizierer an der Wurzel wird über das Interface `NaiveBayesClassifierInterface` die Methode `classify` aufgerufen. Als Parameter wird das zu klassifizierende Dokument übergeben. Dieser Klassifizierer klassifiziert nun über seine Nachfolgekategorien. Für die Nachfolgekategorie mit dem besten Ergebnis wird rekursiv ein weiterer Klassifizierer erzeugt und gestartet bzw. mehrere Klassifizierer im Falle einer Multi-Pfad-Klassifikation. Die Rekursion setzt sich fort, bis ein Blatt erreicht ist. Dann wird das Label des Blattes als Ergebnis zurückgegeben bzw. mehrere Labels.

Das Naïve-Bayes-Klassifikationsmodell stellt neben den hierarchischen Klassifizierern noch einen flachen Klassifizierer zur Verfügung. Der flache Klassifizierer stellt jedoch einen Spezialfall dar. Die Wurzel ist hierbei eine virtuelle Kategorie, deren Nachfolgekategorien alle Blattkategorien der Hierarchie sind. Der Klassifizierer, der für die Wurzel erzeugt wird, ist somit der Einzige im gesamten Klassifikationsmodell.

5.2.1. Trainingsmodul

Es wurden drei verschiedene Verfahren zum Trainieren eines Naïve-Bayes-Klassifizierers implementiert. Die entsprechenden Klassen sind in dem UML-Klassendiagramm in **Abb. 5.6** dargestellt. Die abstrakte Klasse `NBCTraining` implementiert das Interface `NBCTrainingInterface` des Naïve-Bayes-Klassifikationsmodelles. Die Klasse dient als Oberklasse für die Trainingsverfahren und stellt allgemeine Methoden bereit. Die Klassen `NBCTrainingHierarchical`, `NBCTrainingShrinkage` und `NBCTrainingFlat` erweitern die Klasse `NBCTraining` und implementieren die Methode `training` des Interface `NBCTrainingInterface`. Im Folgenden werden die drei Trainingsverfahren genauer beschrieben.

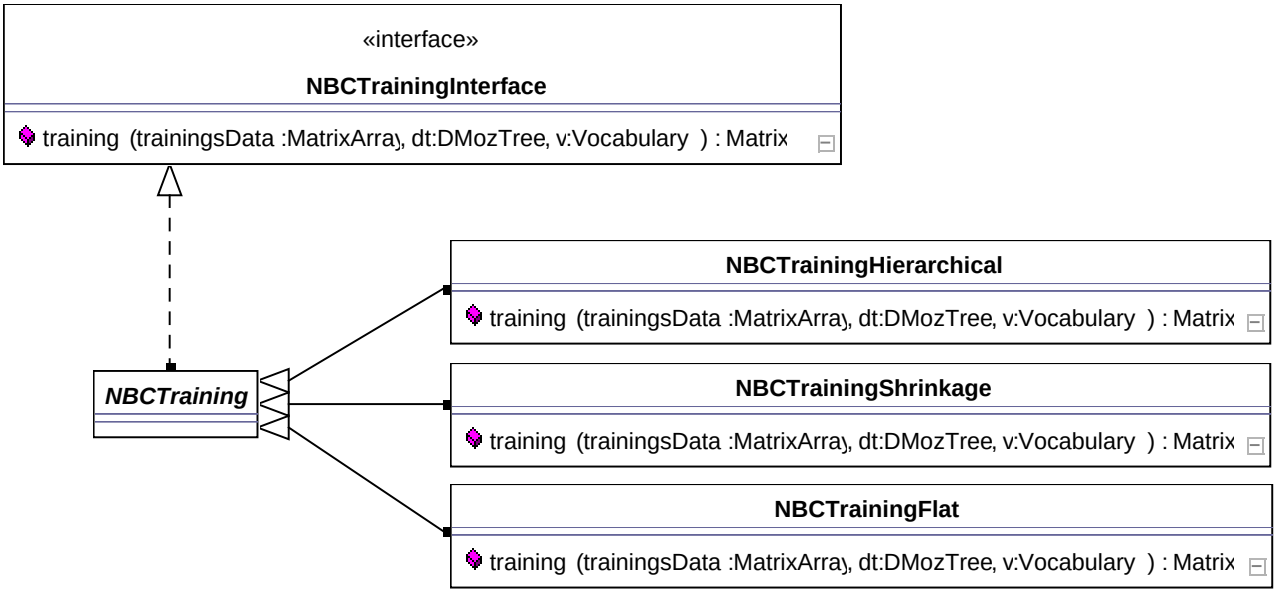


Abb. 5.6.: *Trainingsmodul des Naïve-Bayes-Klassifikationsmodelles. Die abstrakte Klasse* `NBCTraining` *implementiert das Interface* `NBCTrainingInterface`*. Die Klassen* `NBCTrainingHierarchical`*,* `NBCTrainingShrinkage` *und* `NBCTrainingFlat` *erweitern die Klasse* `NBCTraining` *und implementieren die Methode* `training` *des Interfaces* `NBCTrainingInterface`*.*

NBCTrainingHierarchical

In der Klasse `NBCTrainingHierarchical` wurde ein hierarchisches Trainingsverfahren implementiert. Für jede innere Kategorie der Hierarchie wird ein Naïve-Bayes-Klassifizierer erstellt. Trainiert wird der Klassifizierer mit allen in der Hierarchie darunter liegenden Dokumenten.

Das Ergebnis des Trainings ist eine $|Dic| \times |C|$-Matrix. Die Matrix enthält die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$, für alle Merkmale $f_k \in Dic$ und alle Kategorien $c_i \in C$. Die bedingten Wahrscheinlichkeiten werden über die Häufigkeit der Merkmale berechnet und mit der Laplace-Glättung geglättet, siehe Abschnitt 3.3.2, Gleichung 3.14. Die Häufigkeiten werden wie folgt bestimmt:

$$FF(f_k, c_i) = \sum_{c' \in succ(c_i)} FF(f_k, c')$$

Die Häufigkeit eines Merkmales f_k in einer Kategorie c_i ist die Summe der Häufigkeiten von f_k in allen Unterkategorien von c_i.

NBCTrainingShrinkage

In der Klasse `NBCTrainingShrinkage` ist ein weiteres hierarchisches Trainingsverfahren implementiert. Dieses Trainingsverfahren ist eine Erweiterung des zuvor erläuterten Verfahrens.

Zunächst werden die bedingten Wahrscheinlichkeiten $P(f_k|c_i)$, für alle Merkmale $f_k \in Dic$ und alle Kategorie $c_i \in C$ wie zuvor berechnet. Zusätzlich wird jedoch Shrinkage eingesetzt, siehe Abschnitt 4.3, um die bedingten Wahrscheinlichkeiten der Blattkategorien zu glätten. Das Ergebnis des Trainings ist eine $|Dic| \times |C|$-Matrix. Für alle inneren Kategorien c_l enthält die Matrix die bedingten Wahrscheinlichkeiten $P(f_k|c_l)$. Für alle Blattkategorien c_j enthält die Matrix die mit Shrinkage geglätteten bedingten Wahrscheinlichkeiten $\widehat{P}(f_k|c_j)$.

NBCTrainingFlat

Die Klasse `NBCTrainingFlat` implementiert ein flaches Trainingsverfahren. Es wird nur ein Klassifizierer trainiert. Der Klassifizierer klassifiziert über die Blattkategorien der Hierarchie. Die hierarchische Struktur wird ignoriert.

$C' \subset C$ sei die Menge der Blattkategorien. Für alle Merkmale $f_k \in Dic$ und alle Blattkategorien $c_j \in C'$ werden die bedingten Wahrscheinlichkeiten $P(f_k|c_j)$ berechnet. Dies geschieht wie bei

dem hierarchischen Verfahren über die Häufigkeiten der Merkmale und die Laplace-Glättung. Das Ergebnis des Trainings ist eine $|Dic| \times |C'|$-Matrix, die die bedingten Wahrscheinlichkeiten enthält.

5.2.2. Klassifikationsmodul

Das Klassifikationsmodul des Naïve-Bayes-Klassifikationsmodelles stellt drei verschiedene Typen von selbst implementierten Naïve-Bayes-Klassifizierern zur Verfügung. Die entsprechenden Klassen sind in dem UML-Klassendiagramm in **Abb. 5.7** dargestellt. Die abstrakte Klasse `NBC` implementiert das Interface `NaiveBayesClassifierInterface` des Naïve-Bayes-Klassifikationsmodelles. Die Klasse stellt verschiedene allgemeine Methoden bereit und dient als Oberklasse für die konkreten Klassifizierer. Die drei Klassen `NaiveBayesClassifier`, `NaiveBayesClassifierMultiPath` und `NaiveBayesClassifierFlat` erweitern die abstrakte Klasse `NBC` und implementieren die Methode `classify` des Interfaces `NaiveBayesClassifierInterface`.

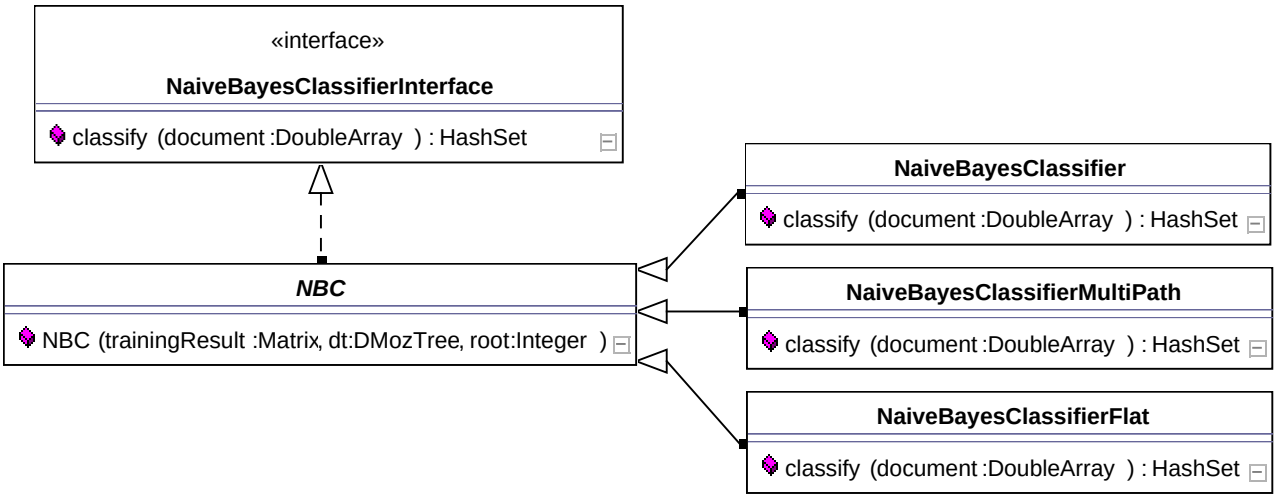


Abb. 5.7.: *Klassifikationsmodul des Naïve-Bayes-Klassifikationsmodelles. Die abstrakte Klasse* `NBC` *implementiert das Interface* `NaiveBayesClassifierInterface`*. Die Klassen* `NaiveBayesClassifier`*,* `NaiveBayesClassifierMultiPath` *und* `NaiveBayesClassifierFlat` *erweitern die Klasse* `NBC` *und implementieren die Methode* `classify` *des Interfaces* `NaiveBayesClassifierInterface`*.*

Jede der drei Klassifiziererklassen überschreibt den Konstruktor der Oberklasse `NBC`. Somit bekommen die Klassen bei der Instanziierung die folgenden Parameter übergeben:

1. Die Matrix mit den Trainingsergebnissen.
2. Die Repräsentation der Hierarchie als `DMozTree`.
3. Die Stammkategorie. Das ist die Kategorie, an der sich der Klassifizierer befindet.

Des Weiteren wird in den Konstruktoren der Klassifiziererklassen eine Liste mit den Kategorien erstellt, über die klassifiziert werden soll. Im Falle der hierarchischen Klassifizierer handelt es sich hierbei um die direkten Nachfolgekategorien der Stammkategorie. Damit stehen dem Klassifizierer alle benötigten Informationen zu Verfügung, um ein Dokument zu klassifizieren.

Angenommen ein Dokument d soll in eine Menge von Kategorien C' klassifiziert werden. Ein Naïve-Bayes-Klassifizierer berechnet hierzu die bedingten Wahrscheinlichkeiten $P(c_i|d)$, für alle Kategorien $c_i \in C'$, siehe Abschnitt 3.3.2. Das Dokument d wird dann der Kategorie mit der höchsten Wahrscheinlichkeit zugeordnet. Die Berechnung der bedingten Wahrscheinlichkeiten ist für alle, in diesem Klassifikationsmodell implementierten Typen von Naïve-Bayes-Klassifizierern gleich. Aus diesem Grund stellt die gemeinsame Oberklasse `NBC` eine Methode bereit, die diese Berechnung durchführt. Die Methode `naiveBayes` der Klasse `NBC` berechnet die bedingten Wahrscheinlichkeiten $P(c_i|d)$ für alle Kategorien $c_i \in C'$. Als Ergebnis wird ein Array mit den bedingten Wahrscheinlichkeiten für jede Kategorie c_i zurückgegeben.

Im Folgenden wird die Implementierung der Klassifizierer und deren Funktionsweise genauer beschrieben.

NaiveBayesClassifier

Die Klasse `NaiveBayesClassifier` implementiert einen hierarchischen Single-Pfad-Naïve-Bayes-Klassifizierer. Ein Beispiel einer Klassifikation mit diesem Typ von Klassifizierer ist in **Abb. 5.8** dargestellt. Ein Dokument wird top-down in die Hierarchie einsortiert. Hierbei wird immer nur ein Pfad verfolgt. Jeder Klassifizierer erzeugt und startet rekursiv einen weiteren Klassifizierer für die Kategorie, in die er das Dokument klassifiziert hat. Die Rekursion setzt sich fort bis eine Blattkategorie erreicht ist. Dann wird das Label der Blattkategorie als Klassifikationsergebnis zurückgegeben.

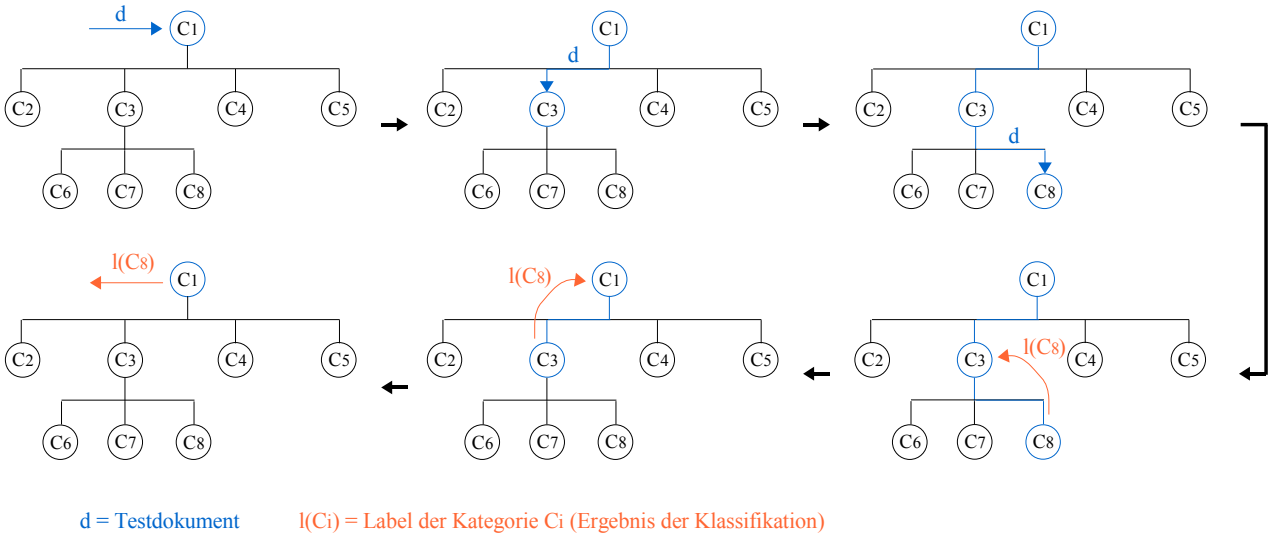


Abb. 5.8.: *Beispiel einer Klassifikation eines Dokumentes d mit Single-Pfad-Naïve-Bayes-Klassifizierern (`NaiveBayesClassifier`). Das Dokument wird beginnend bei der Wurzel top-down weiter propagiert. Hierbei wird immer nur der beste Pfad verfolgt. Wenn eine Blattkategorie erreicht ist, wird das Label der Blattekategorie als Klassifikationsergebnis zurückgegeben.*

Die Liste der Kategorien, über die klassifiziert werden soll, enthält immer die direkten Nachfolgekategorien der Stammkategorie, beispielsweise c_2, c_3, c_4, c_5 für den Klassifizierer mit Stammkategorie c_1 in Abb. 5.8. In der Methode `classify` der Klasse `NaiveBayesClassifier` werden zunächst für alle diese Kategorien die bedingten Wahrscheinlichkeiten berechnet. Dies geschieht durch die Methode `naiveBayes` der Oberklasse `NBC`. Als nächstes wird geprüft, welche Kategorie c_j die höchste Wahrscheinlichkeit besitzt. Beispielsweise ist $c_j = c_3$ für den Klassifizierer mit Stammkategorie c_1 in Abb. 5.8. Falls es sich bei der Kategorie c_j um eine Blattkategorie handelt, wird das Label von c_j als Klassifikationsergebnis zurückgegeben. Ansonsten, c_j ist eine innere Kategorie, wird ein neuer Klassifizierer mit Stammkategorie c_j instanziiert und gestartet.

NaiveBayesClassifierFlat

Die Klasse `NaiveBayesClassifierFlat` implementiert einen flachen Naïve-Bayes-Klassifizierer. Der Klassifizierer klassifiziert nur über die Blattkategorien der Hierarchie. Ein Beispiel einer flachen Klassifikation ist in **Abb. 5.9** zu sehen.

Im Konstruktor wird die Liste der Kategorien erstellt, über die klassifiziert werden soll. In diese Liste werden alle Blattkategorien der Hierarchie eingetragen, beispielsweise $c_2, c_6, c_7, c_8, c_4, c_5$

für den Klassifizierer mit Stammkategorie c_1 in Abb. 5.9. In der Methode `classify` der Klasse `NaiveBayesClassifierFlat` wird zunächst die Methode `naiveBayes` der Oberklasse `NBC` aufgerufen. Die Methode `naiveBayes` berechnet die bedingten Wahrscheinlichkeiten für alle Blattkategorien. Nun wird die Blattkategorie mit der höchsten Wahrscheinlichkeit bestimmt, beispielsweise c_8 in Abb. 5.9. Das Label dieser Blattkategorie wird dann als Klassifikationsergebnis zurückgegeben.

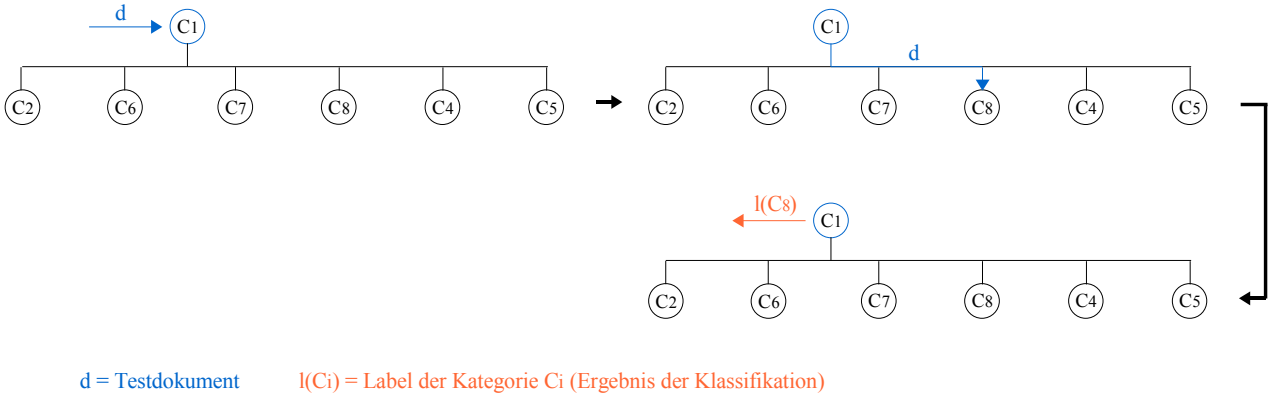


Abb. 5.9.: *Beispiel einer Klassifikation eines Dokumentes d mit einem flachen Naïve-Bayes-Klassifizierer (`NaiveBayesClassifierFlat`). Der Klassifizierer an der Wurzel klassifiziert über alle Blattkategorien der Hierarchie. Das Label der Blattkategorie mit der höchsten Wahrscheinlichkeit wird als Klassifikationsergebnis zurückgegeben.*

NaiveBayesClassifierMultiPath

Bei dem in der Klasse `NaiveBayesClassifierMultiPath` implementierten Klassifizierer handelt es sich um einen hierarchischen Multi-Pfad-Naïve-Bayes-Klassifizierer. **Abb. 5.10** zeigt ein Beispiel einer Klassifikation mit diesem Typ von Klassifizierer. Wie bei dem Single-Pfad-Naïve-Bayes-Klassifizierer wird die Hierarchie beginnend bei der Wurzel top-down durchlaufen. Allerdings werden bei der Multi-Pfad-Klassifikation mehrere Pfade verfolgt. Jeder Klassifizierer erzeugt und startet für die besten Nachfolgekategorien rekursiv jeweils einen Klassifizierer. Die Rekursion setzt sich auf einem Pfad fort, bis eine Blattkategorie erreicht wurde. Dann wird das Label der Blattkategorie zurückgegeben. Das Ergenis der Klassifikation ist eine Menge von Labels.

Die Liste der Kategorien, über die klassifiziert wird, setzt sich wie bei dem Single-Pfad-Klassifizierer aus den direkten Nachfolgekategorien der Stammkategorie zusammen. In der Methode `classify` der Klasse `NaiveBayesClassifierMultiPath` werden zunächst für alle diese Kategorien die bedingten Wahrscheinlichkeiten berechnet. Die Berechnung erfolgt durch

die Methode `naiveBayes` der Oberklasse `NBC`. Nun werden alle Kategorien c_j bestimmt, deren Wahrscheinlichkeiten einen gewissen Grenzwert überschreiten, beispielsweise c_6 und c_8 für den Klassifizierer mit Stammkategorie c_3 in Abb. 5.10. Als nächstes wird für jede Kategorie c_j geprüft, ob c_j eine Blattkategorie ist. Falls dies der Fall ist, wird das Label der Kategorie c_j zurückgegeben. Falls die Kategorie c_j keine Blattkategorie ist, wird ein neuer Klassifizierer mit c_j als Stammkategorie instanziiert und gestartet.

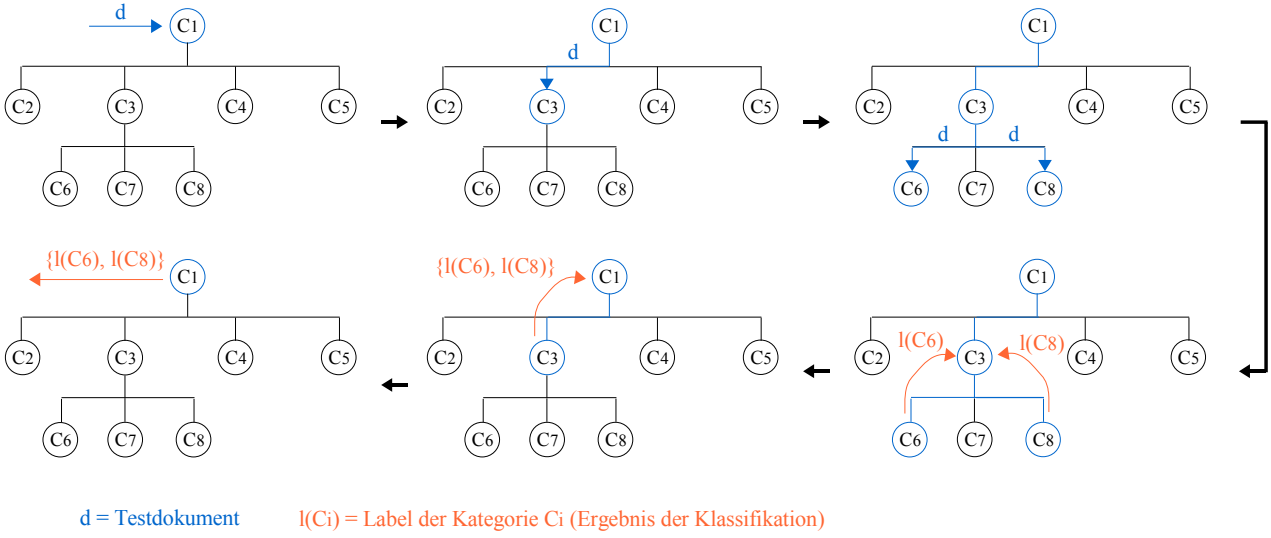


Abb. 5.10.: *Beispiel einer Klassifikation eines Dokumentes d mit einem Multi-Pfad-Naïve-Bayes-Klassifizierer (`NaiveBayesClassifierMultiPath`). Das Dokument wird beginnend bei der Wurzel top-down weiter propagiert. Allerdings wird nicht nur der beste Pfad verfolgt, sondern mehrere, vielversprechende Pfade. Das Klassifikationsergebnis ist eine Menge mit den Labels der erreichten Blattkategorien.*

5.3. Weka-Klassifikationsmodell

Die Software *Waikato Environment for Knowledge Analysis*[1](Weka) bietet eine Reihe von in Java implementierten Klassifizierern und eine Fülle von Algorithmen zum maschinellen Lernen an [WF05]. Weka ist unter der GNU General Public License veröffentlichte Open Source Software. Die Experimente in dem folgenden Kapitel benutzen die Development Version 3.5.3.

Hierarchische Textklassifikationsverfahren können mit Weka nicht evaluiert werden. Deshalb wurde in dieser Arbeit ein hierarchisches Klassifikationsmodell erstellt, das es ermöglicht Weka-

[1]http://www.cs.waikato.ac.nz/ ml/index.html

Klassifizierer für hierarchische Klassifizierungen zu verwenden und zu evaluieren. Für die Experimente der Arbeit wurden eine Reihe von Naïve-Bayes-Varianten —vgl. Abschnitt 3.3.2— ausgewählt, die Weka anbietet.

- Naïve Bayes
- Naïve Bayes Multinomial
- Complement Naïve Bayes

Darüber hinaus bietet Weka das Meta-Klassifikationsverfahren AdaBoostM1 —das Verfahren ist in Abschnitt 3.3.3 beschrieben— und eine Schnittstelle zur Einbindung einer SVM-Library [CL01]. Der SVM-Algorithmus wird in den Experimenten nicht verwendet, ist aber dennoch im Framework integriert.

Das hierarchische Klassifikationsmodell, um die Weka-Klassifizierer einzubinden, ist in **Abb. 4.1** dargestellt. Das Klassifikationsmodell soll die Kategorien einer Hierarchie repräsentieren. Jede Kategorie der Hierarchie, entspricht einem Klassifizier, der mit den Dokumenten der Nachfolger trainiert wird. Ausgangspunkt sind die Trainingsdokumente der Blattkategorien in der Hierarchie, die in ein Weka kompatibles Format umgewandelt und zum Erstellen der Klassifizierer verwendet werden. Für die inneren Knoten der Hierarchie werden die Trainingsdaten aus den Dokumenten der Nachfolger generiert. Zwei unterschiedliche Vorgehensweisen sind hierfür implementiert. Zum einen besteht die Möglichkeit, jedem Nachfolger einen festgelegten Anteil an Traingsdokumenten zu entnehmen. Zum anderen lässt sich dieser Anteil dynamisch bestimmen, sodass alle Knoten die gleiche Anzahl zum Training besitzen. Die Auswahl der Dokumente geschieht zufällig.

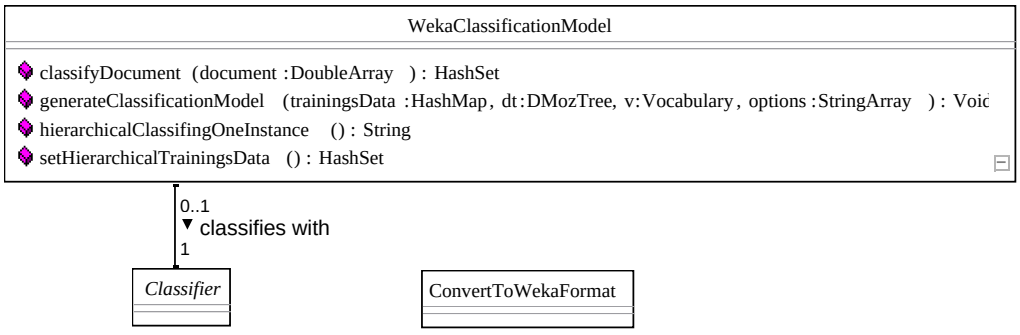


Abb. 5.11.: *UML-Klassendiagramm des Weka-Klassifikationsmodells. Weka-Klassifizierer können für hierarchische Klassifikation genutzt werden. Die Klasse* `WekaClassificationModel` *implementiert das Interface* `ClassificationModel`*. Weka-Klassifizierer nutzen die abstrakte Klasse* `Classifier` *und sind dadurch im Modell eingebunden. Die Klasse* `ConvertToWekaFormat` *konvertiert die im Framework verwendeten Daten.*

Das Weka-Klassifikationmodell in **Abb. 5.11** implementiert das Interface `ClassificationModel` in **Abb. 5.1** und kann so im Evaluierungsframework eingebunden werden. Die Klasse `WekaClassificationModel` kann Weka-Klassifizier nutzen, die von der abstrakten Klasse `weka.classifiers.Classifier` erben. Die für diese Arbeit entwickelte Benutzungsschnittstelle stellt eine Auswahl an Weka-Klassifiziern zur Verfügung.

Die Klasse `ConvertToWekaFormat` konvertiert die vorliegenden Daten in ein Weka-Format, welches zum Training und zur Evaluierung von Weka-Klassifiziern genutzt werden können. Wie man in dem Aktivitätendiagramm zu `generateClassificationModel` in **Abb. 5.12** sehen kann, wird im ersten Schritt die Konvertierung der Daten in `Instances` vollzogen. Sobald dies für jede Kategorie der Eingabe geschehen ist, wird wie oben beschrieben die DMoz-Hierarchie nachgebildet. Anschließend werden die Weka-Klassifizierer erstellt und trainiert. Die Klassifizierer werden in einer Datenstruktur gespeichert, die als Schlüssel den Namen der Kategorie verwendet, die sich an gleicher Position wie der Klassifizierer in der Hierarchie befindet.

Das Klassifizieren eines Dokumentes mit `classifyDocument` ist in **Abb. 5.13** dargestellt. Bevor das Dokument klassifiziert wird, wird es konvertiert und anschließend durch den obersten Klassifizierer in der Hierarchie klassifiziert. Dieser liefert als Ergebnis das Label einer Kategorie. Es wird solange der nächste zum aktuellen Label gehörende Klassifizierer mit dem Eingabedokument aufgerufen, bis es keinen Nachfolger mehr gibt. Dieses Vorgehen entspricht dem **Algorithmus 2** in Kapitel 2. Die konvertierten Trainings- und Testdaten lassen sich im Weka-eigenen Arff-Format speichern und laden. Um eine Vorstellung davon zu geben, welche Informationen in Weka erforderlich sind, wird an dieser Stelle ein Beispiel für eine Arff-Datei gegeben.

```
@RELATION category
@ATTRIBUTE music REAL
@ATTRIBUTE page REAL
@ATTRIBUTE perform REAL
@ATTRIBUTE movi REAL
@ATTRIBUTE record REAL
@ATTRIBUTE class {Top.Arts.Literature, Top.Arts.Movies, Top.Arts.Music}
@DATA
1.0, 3.0, 0.0, 7.0, 7.0, Top.Arts.Literature
5.0, 6.0, 0.0, 1.0, 0.0, Top.Arts.Literature
1.0, 0.0, 0.0, 4.0, 1.0, Top.Arts.Movies
0.0, 0.0, 0.0, 1.0, 7.0, Top.Arts.Movies
5.0, 0.0, 1.0, 2.0, 0.0, Top.Arts.Music
```

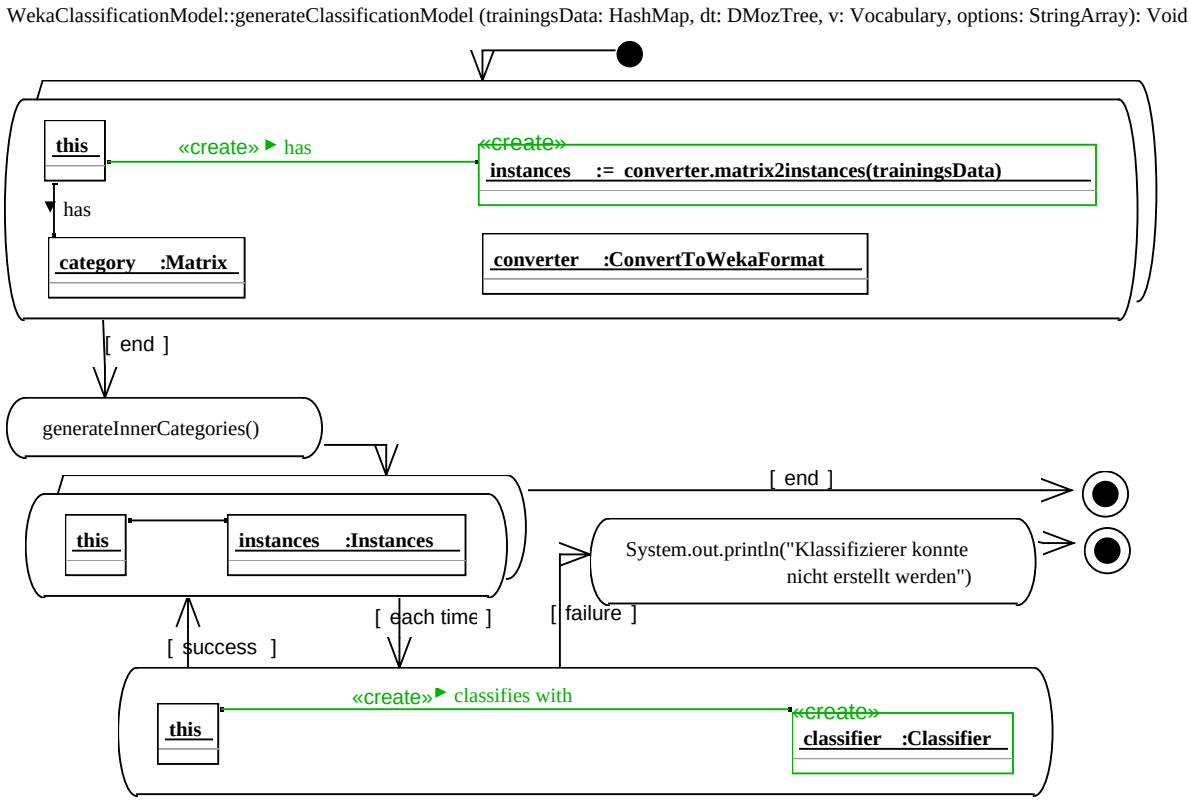


Abb. 5.12.: *Aktivitätendiagramm der Methode* `generateClassificationModel` *in Klasse* `WekaClassificationModel`*. Die Dokumente der Katgorien werden zunächst konvertiert. Die konvertierten Daten entsprechen den so genannten* `Instances`*-Objekten. Anschließend wird eine Hierarchie aus den Blattkategorien generiert. Mit der erzeugten Trainingshierarchie wird für jeden inneren Knoten ein Weka-Klassifizierer erstellt und trainiert.*

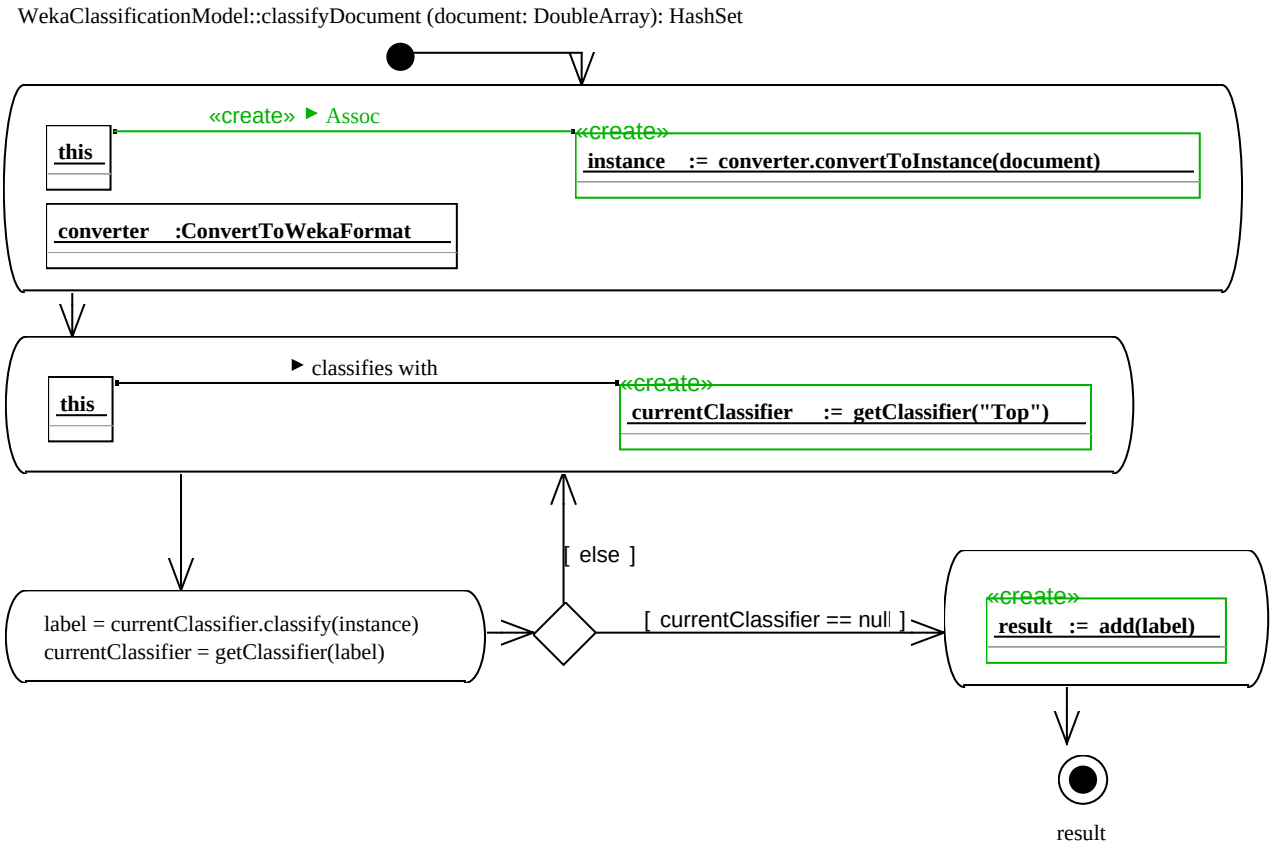


Abb. 5.13.: *Aktivitätendiagramm der Methode* `classifyDocument` *in Klasse* `WekaClassificationModel`*. Nach dem Konvertieren des Eingabedokumentes wird der oberste Klassifizier in der Hierarchie aufgerufen. In einem top-down-Verfahren wird klassifiziert und abschließend das erzeugte Label zurückgegeben.*

Zunächst werden die Merkmale definiert, die Reelle Zahlen sind. In diesem Fall sind das „music“, „page“, „perform“ usw. . Zusätzlich werden die Namen der Kategorien angelegt. Diese gehören zu dem Attribut „class“. Anschließend folgen die Daten. Jede Zeile repräsentiert ein Dokument. In der Reihenfolge, wie auch die Attribute definiert sind, sind die Werte der Merkmale des Dokumentes anzugeben.

Das Framework bietet mit der Klasse `ArffConverter` die Möglichkeit, aus den Blattkategorien Arff-Dateien zu erzeugen, wobei für jede Kategorie in der Hierarchie eine eigene Datei angelegt wird. Die Inhalte der inneren Kategorien werden aus den Blattkategorien erstellt, indem zufällig Dokumente den Nachfolgern einer Kategorie entnommen und zu dieser hinzugefügt werden. Die generierten Dateien kann man mit dem Weka-Tool laden und dort Experimente durchführen, die sich auf einen einzelnen Knoten in der Hierarchie beziehen.

Anmerkung zur Performanz:
Beim Aufruf der Methode `generateClassificationModel` werden etwa 21% der Rechenzeit vom Weka-Framework zur Erstellung des Klassifizierers benötigt und 79% um die Trainingsdaten bereit zu stellen. Dazu gehört das Konvertieren und das Nachbilden der Hierarchie. Beim Klassifizierern eines Dokumentes mit `classifyDocument` werden 55% der Gesamtdauer des Aufrufs für das Konvertieren des Dokumentes benötigt und 45% für das Klassifizieren. Das Erstellen des Klassifikationsmodells wird durch den Aufbau der Hierarchie und durch das Konvertieren der Trainingsdaten verlangsamt. Für die Verwendung zur Topic Identification ist dies jedoch nicht weiter interessant, da man hier annimmt, dass das Modell a priori erstellt wird. Zu beachten ist der Zeitaufwand einer Klassifikation. Dieser verdoppelt sich in etwa, wenn ein Dokument noch konvertiert wird.

5.4. Subsumption-Klassifikationsmodell

Das Subsumptionsverfahren aus Abschnitt 4.4, ist mit der Klasse `SubsumptionClassificationModel` umgesetzt worden. Das Modell ist ein Meta-Klassifikationsmodell, da es sich aus zwei Klassifikationsmodellen zusammensetzt, die beliebig ausgewählt werden können. Die Methode `generateClassificationModel` in **Abb. 5.14** erzeugt die Instanzen der ausgewählten Klassifikationsmodelle und ruft deren `generateClassificationModel`-Methoden auf. Beim Klassifizieren eines Dokumentes —vgl. **Abb. 5.15**— wird bei jedem Modell die Methode `classifyDocument` ausgeführt. Die Rückgabewerte werden mit der Methode `subsume` zusammengefügt.

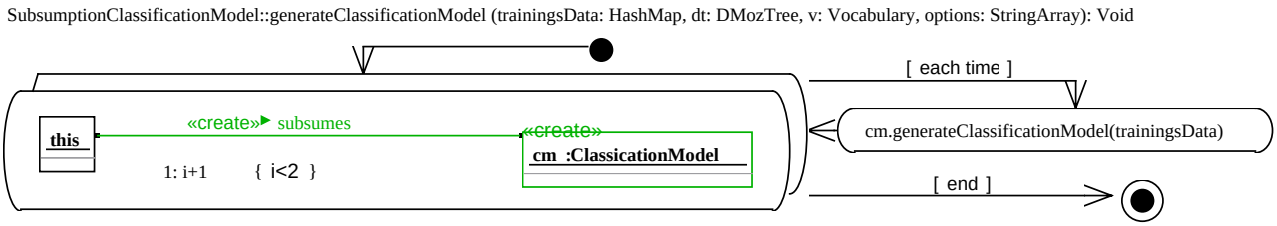


Abb. 5.14.: *Aktivitätendiagramm der Methode* `generateClassificationModel` *in Klasse* `SubsumptionClassificationModel`*. Zwei Klassifikationsmodelle werden instanziert.*

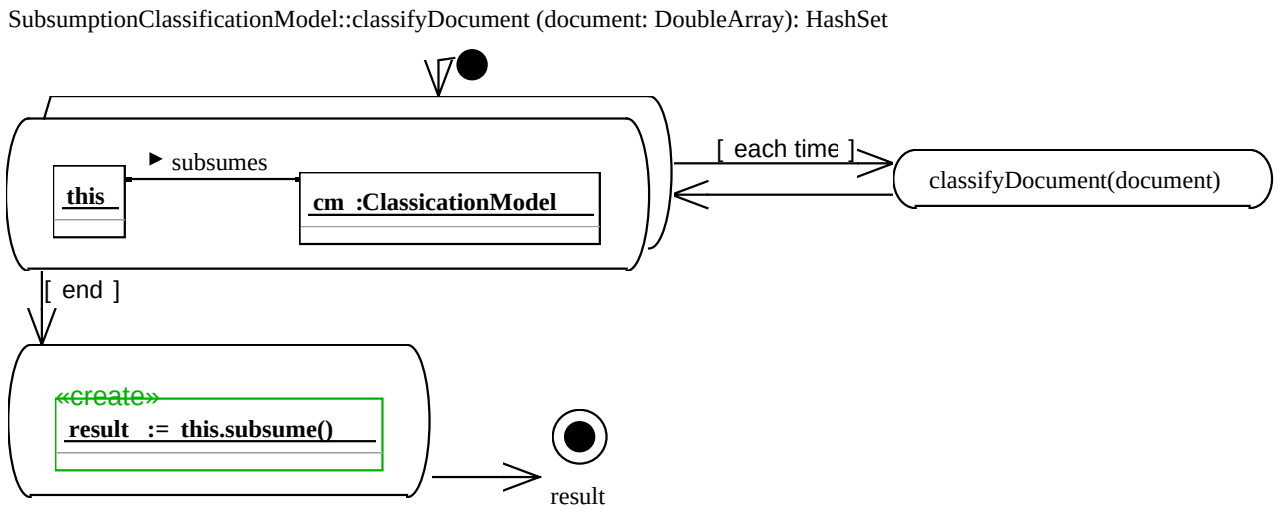


Abb. 5.15.: *Aktivitätendiagramm der Methode* `classifyDocument` *in Klasse* `SubsumptionClassificationModel`*. Die Methode* `classifyDokument` *wird bei jedem instanzierten Klassifikationsmodell aufgerufen und die Ergebnisse werden subsumiert.*

5.5. Evaluierung

Teil des Frameworkes ist ein Evaluierungsmodul, dessen Aufbau in **Abb. 5.16** dargestellt ist. Es besteht im wesentlichen aus der Hauptklasse `Evaluation`, einem Splitting-Verfahren (`EasySplit`) und einer Menge von Evaluierungsfunktionen. `EasySplit` teilt die vorhandnen Dokumente mit einem Zufallsverfahren in eine Trainings- und eine Testmenge. Dabei wird der Klasse ein Splitting-Verhältnis übergeben. `Evaluation` arbeitet in mehreren Durchläufen, wobei in jedem Durchlauf die Dokumente erneut aufgeteilt werden. Die Ergebisse werden nach Ablauf der Evaluierung gemittelt.

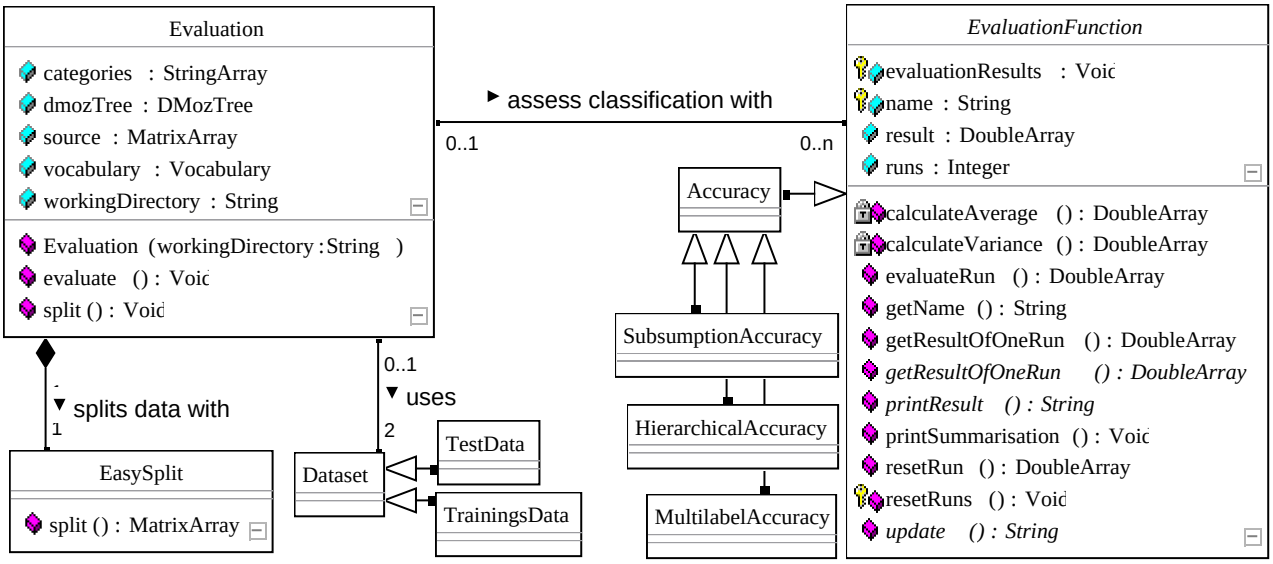


Abb. 5.16.: *UML-Klassendiagramm der Evaluierung. In* `Evaluation` *ist die Funktionalität zum Durchführen der Experimente mit mehreren Durchläufen implementiert. Die vorliegenden Daten werden mit* `EasySplit` *in Trainings- und Testdaten aufgeteilt. In* `Evaluation` *werden verschiedene Evaluierungsfunktionen aufgerufen.* `EvaluationFunction` *ist eine abstrakte Klasse, die Grundfunktionalitäten für spezielle Maße bietet.*

Das Aktivitätendiagramm in **Abb. 5.17** zeigt einen einzelnen Durchlauf in der Evaluierung. Nach dem Teilen in Trainings- und Testdokumente, wird ein ausgewähltes Klassifikationsmodell erstellt und trainiert. Anschließend wird `classifyDocument` mit einem Dokument der Testmenge auf dem ausgewählten Modell ausgeführt. Bei jeder instanzierten Evaluierungsfunktion wird die Methode `update` mit der Rückgabe der Klassifikation aufgerufen. Die Ergebnisse werden über alle Testdokumente gemittelt und ausgegeben.

Das Framework kann mit Evaluierungsfunktionen ergänzt werden, indem die neue Evaluierungsfunktion von der abstrakten Klasse `EvaluationFunction` erbt. Diese Klasse übernimmt

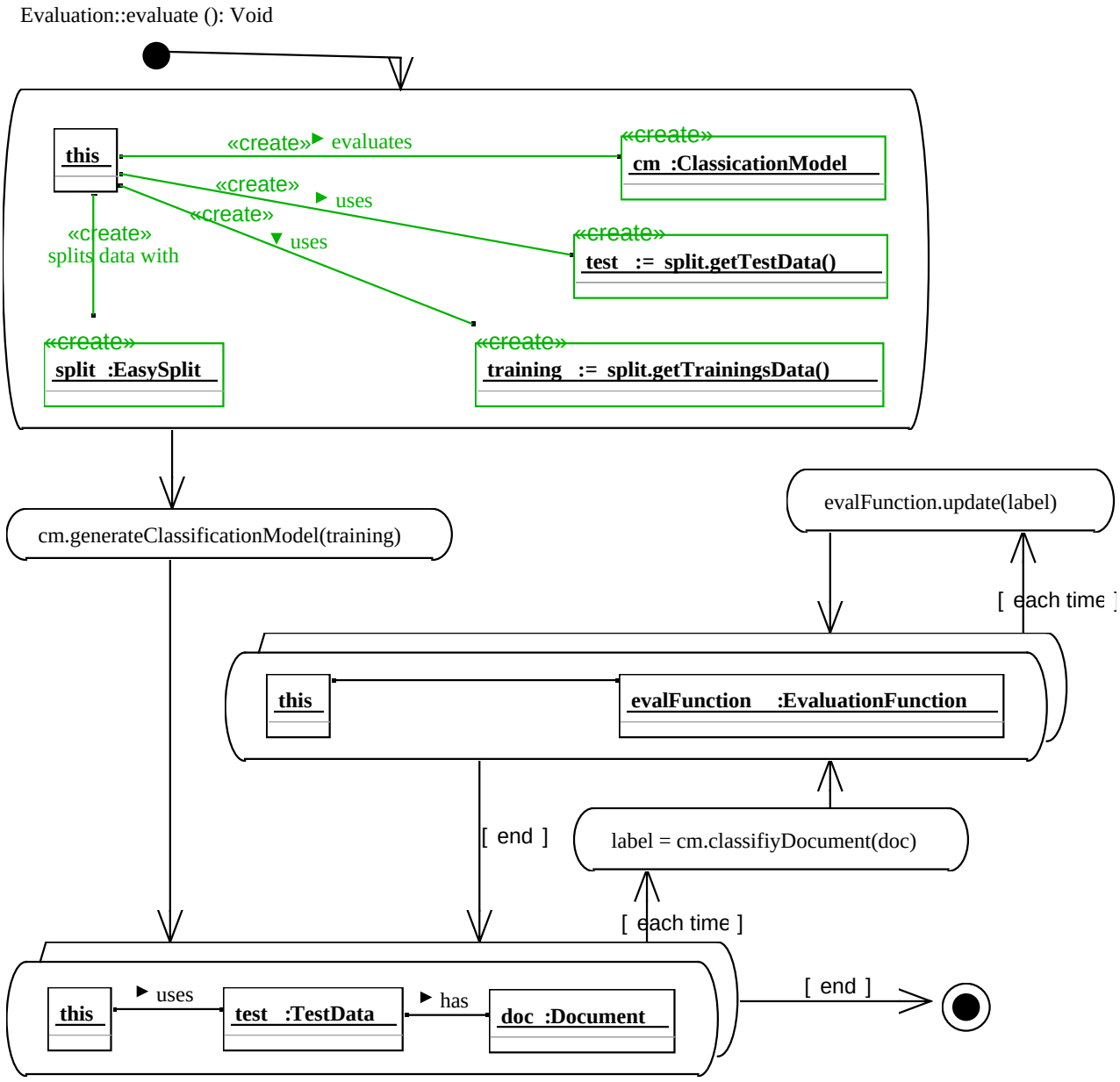


Abb. 5.17.: *Aktivitätendiagramm der Methode* `evaluate` *in Klasse* `Evaluation`*. Vorliegende Daten werden in Trainings- und Testdokumente aufgeteilt. Mit den Trainingsdokumenten wird das Klassifikationsmodell generiert und die Testdokumente klassifiziert, wobei mit dem Ergebnis die Evaluierungsfunktionen aktualisiert werden. Zum Schluss erfolgt die Ausgabe der Werte der Funktionen.*

verschiedene Grundfunktionalitäten, wie beispielsweise das Mitteln bei mehreren Evaluierungsläufen und erzeugt für jede Funktion ein Ausgabefenster. Bisweilen sind verschiedene Genauigkeitsfunktionen implementiert. Das Maß für die Fehler- bzw. Erfolgsrate eines Klassifizierers ist mit der Klasse `Accuracy` umgesetzt.

Ein spezielleres Maß ist die hierarchische Genauigkeit. Die hierarchische Genauigkeit zeigt die Erfolgsrate eines hierarchischen Klassifikationsverfahren für jede Tiefe der Hierarchie. Wenn beispielweise die korrekte Kategorie Top/Arts/Movies wäre, aber die vorgeschlagene Kategorie Top/Arts/Music lautet, dann ist in den Tiefen 0 und 1 korrekt klassifiziert worden, jedoch nicht in der Tiefe 2. Die hierarchische Genauigkeit ist mit der Klasse `HierarchicalAccuracy`), die von `Accuracy` erbt, implementiert.

Die Klasse `SubsumptionAccuracy` implementiert das in Abschnitt 4.4 beschriebene Evaluierungsmaß.

5.6. Indexierung

Die Indexierung ist in dieser Arbeit eine besondere Herausforderung, da große Datenmengen verarbeitet werden. Es muss darauf geachtet werden, dass möglichst wenig Speicher gebraucht wird und dass der Indexierungsprozess mit einer geringen Laufzeit ausgeführt werden kann.

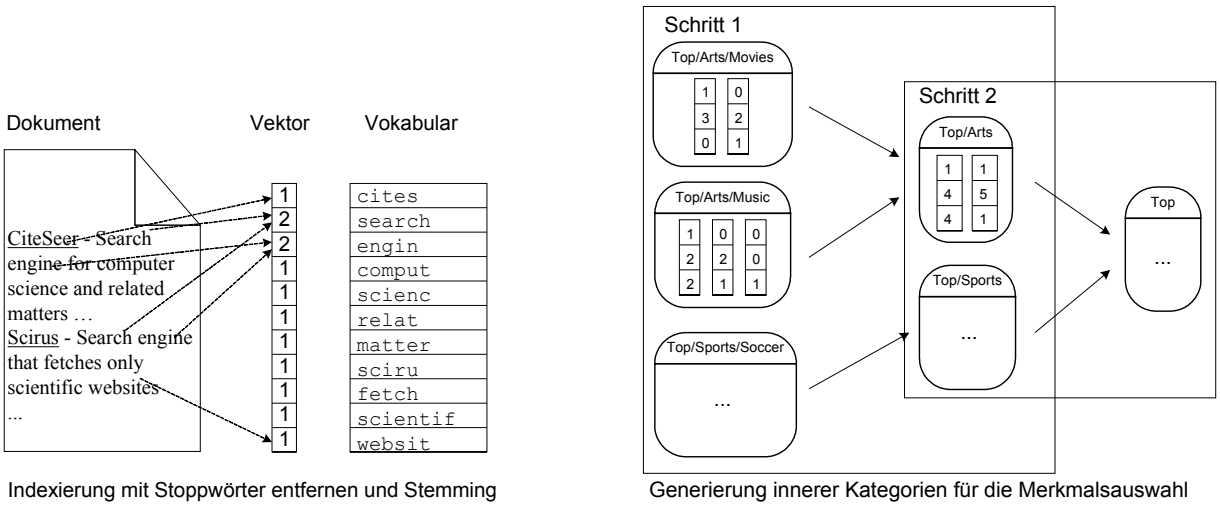


Abb. 5.18.: *(links) Indexierung: Mit jedem neuen Wort wird ein Eintrag im Vokabular erzeugt und die Worthäufigkeit wird in einem Vektor gespeichert, der das Dokument repräsentiert. (rechts) Generierung innerer Kategorien aus Blattkategorien in Tiefe 2: Die Vektoren einer Kategorie werden summiert und bilden zusammen einen neuen Vektor in der Vorgängerkategorie.*

Die Indexierung erstellt Vektoren, die den Inhalt eines oder mehrerer Dokumente repräsentieren. Dazu werden die Häufigkeiten der vorkommenden Merkmale ermittelt und als Eintrag im Vektor gespeichert, vgl. **Abb. 5.18** (links). Um die Größe der Vektoren möglichst klein zu halten, ohne das wichtige Informationen verloren gehen, wird eine Merkmalsauswahl angewendet, vgl. Abschnitt 3.1. Das Verfahren Odds Ratio eignet sich am besten, vgl. Abschnitt 3.1.2. Die Merkmale müssen für jeden Klassifizierer in dem hierarchischen Modell geeignet sein müssen. Die zu indexierenden Dokumente entstammen aus den Blattkategorien und die innerern Kategorien müssen generiert werden. Hierfür werden Vektoren erzeugt, die die Nachfolger der jeweiligen Kategorie repräsentieren, vgl. **Abb. 5.18** (rechts). Die Merkmalsauswahl berechnet für die gesamte Hierarchie, wie gut sich die Merkmale in einem Knoten zum Klassifizieren eignen und wählt die Besten aus.

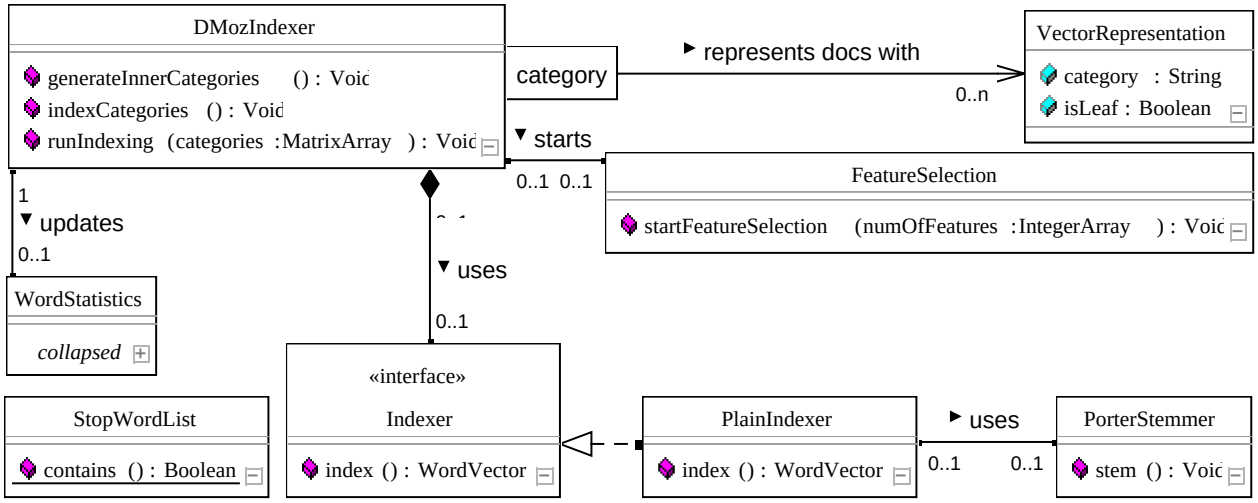


Abb. 5.19.: *UML-Klassendiagramm der Indexierung.* `DMozIndexer` *startet den Indexierungsprozess. Über das Interface* `Indexer` *können verschiedene Indexierer verwendet werden. Der Indexierungsprozess speichert die Dokumente der Kategorien zur Weiterverarbeitung in komprimierten Vekoren (*`VectorRepresentation`*). Die Klasse* `FeatureSelection` *wählt die besten Merkmale aus.*

Die Indexierungskomponente in **Abb. 5.19** setzt sich aus dem `DMozIndexer`, der den Indexierungsprozess startet, der Klasse `FeatureSelection` für die Merkmalsauswahl und einem `Indexer` zusammen.

`Indexer` ist ein Interface aus dem aitools-Package[2], das dieser Arbeit teilweise zur Verfügung stand. Über dieses Interface sind verschiedene Index-Algorithmen verwendbar. Der zum Einsatz kommende Indexierer verwendet eine Stoppwortliste, zum Entfernen ungeeigneter Wörter und einen Porter-Stemmer-Algorithmus, um die Wörter in ihrer Grundform zu bringen. Mit

[2]Aus den Sourcen der Meta-Suchmaschine AiSearch

der Klasse `WordStatistics` werden während der Indexierung Informationen, wie die Gesamthäufigkeit eines Wortes und die Anzahl der Kategorien in denen ein Wort auftaucht, festgehalten.

Die Methode `runIndexing` der Klasse `DMozIndexer` startet den vollständigen Indexierungsprozess. Das Vorgehen der Methode ist in **Abb. 5.20** zu sehen. Die Blattkategorien der Eingabe werden indexiert. Das Ergebnis der Indexierung ist ein Vektor pro Dokument, dessen Einträge die Worthäufigkeiten im Dokument sind. Jeder Eintrag steht für genau ein Wort. Die indexierten Wörter bilden das Vokabular. Dies ist in **Abb. 5.18** (links) zu sehen. Das Vokabular wird mit dem Hinzufügen eines neuen Wortes erweitert. Die Anzahl der unterschiedlichen Merkmale pro Dokument etwa 200.

Jedes Dokument wird als indexierter Vektor in der zugehörigen Kategorie gespeichert. Nachdem die Blattkategorien vollständig indexiert sind, werden die inneren Kategorien generiert. In **Abb. 5.18** (rechts) ist das Erstellen der inneren Kategorien verdeutlicht. Die Summe der Vektoren einer Kategorie bildet einen neuen Vektor in der Vorgängerkategorie. Rekursiv werden so die fehlenden Kategorien der Hierarchie erzeugt. Nach dieser Konstruktion wird der Merkmalsauswahlprozess gestartet.

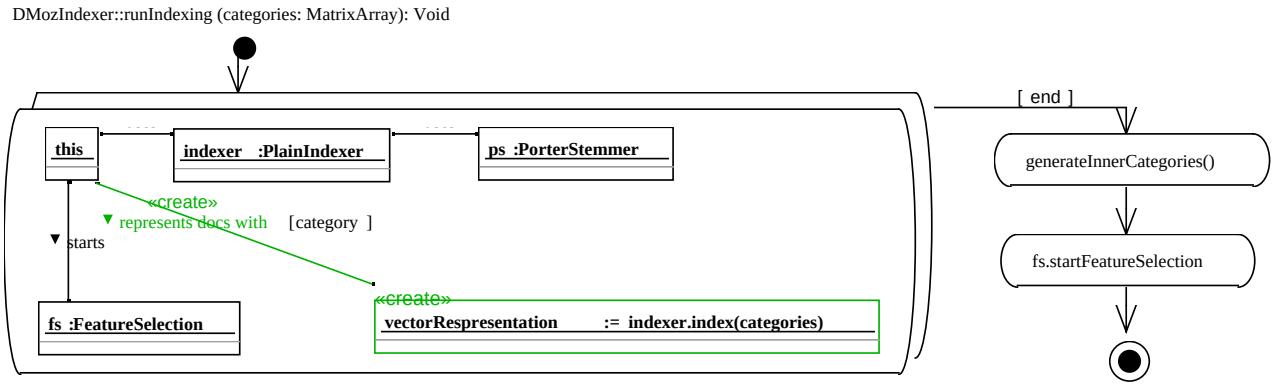


Abb. 5.20.: *Aktivitätendiagramm der Methode* `runIndexing` *in Klasse* `DMozIndexer`*. Jedes Dokument einer Blattkategorie wird indexiert und mit den Ergebnissen werden die inneren Kategorien erzeugt. Die Klasse* `FeatureSelection` *verwendet alle Kategorien zur Merkmalsauswahl.*

Die Klasse `FeatureSelection` hat die Möglichkeit, verschiedene Funktionen zur Auswahl von Merkmalen über ein Interface einzubinden. Der Aufbau ist in **Abb. 5.21**.

Die Methode `startFeatureSelection` wird in **Abb. 5.22** visualisiert. Diese Methode setzt die Merkmalsauswahl um und speichert die Kategorien mit den ausgwählten Merkmalen. Im ersten Schritt wird die Zipf-Mandelbrot-Regel gegebenenfalls ausgeführt. Der Odds-Ratio-Algorithmus in Abschnitt 3.1.2 berechnet die besten Merkmale. Nachdem für jedes Dokument

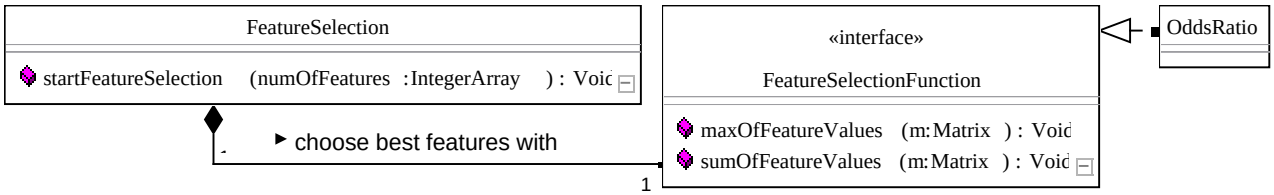


Abb. 5.21.: *UML-Klassendiagramm der Merkmalsauswahl. Die Klasse* `FeatureSelection` *kürzt die Anzahl der Merkmale und speichert die Dokumente mit der entsprechenden Merkmalsanzahl in Matrizen. Es können verschiedene Verfahren verwendet werden, die* `FeatureSelectionFunction` *implementieren, wie beispielsweise* `OddsRatio`*.*

der Kategorien das Merkmalsauswahlverfahren angewendet wurde, werden die neuen Blattkategorien mit ausgewählten Merkmalen in Matrizen transformiert und gespeichert. Für jede Kategorie gibt es eine Matrix, in der jedes Dokument durch ein Spalte repräsentiert wird und jedes Merkmal eine Zeile bildet.

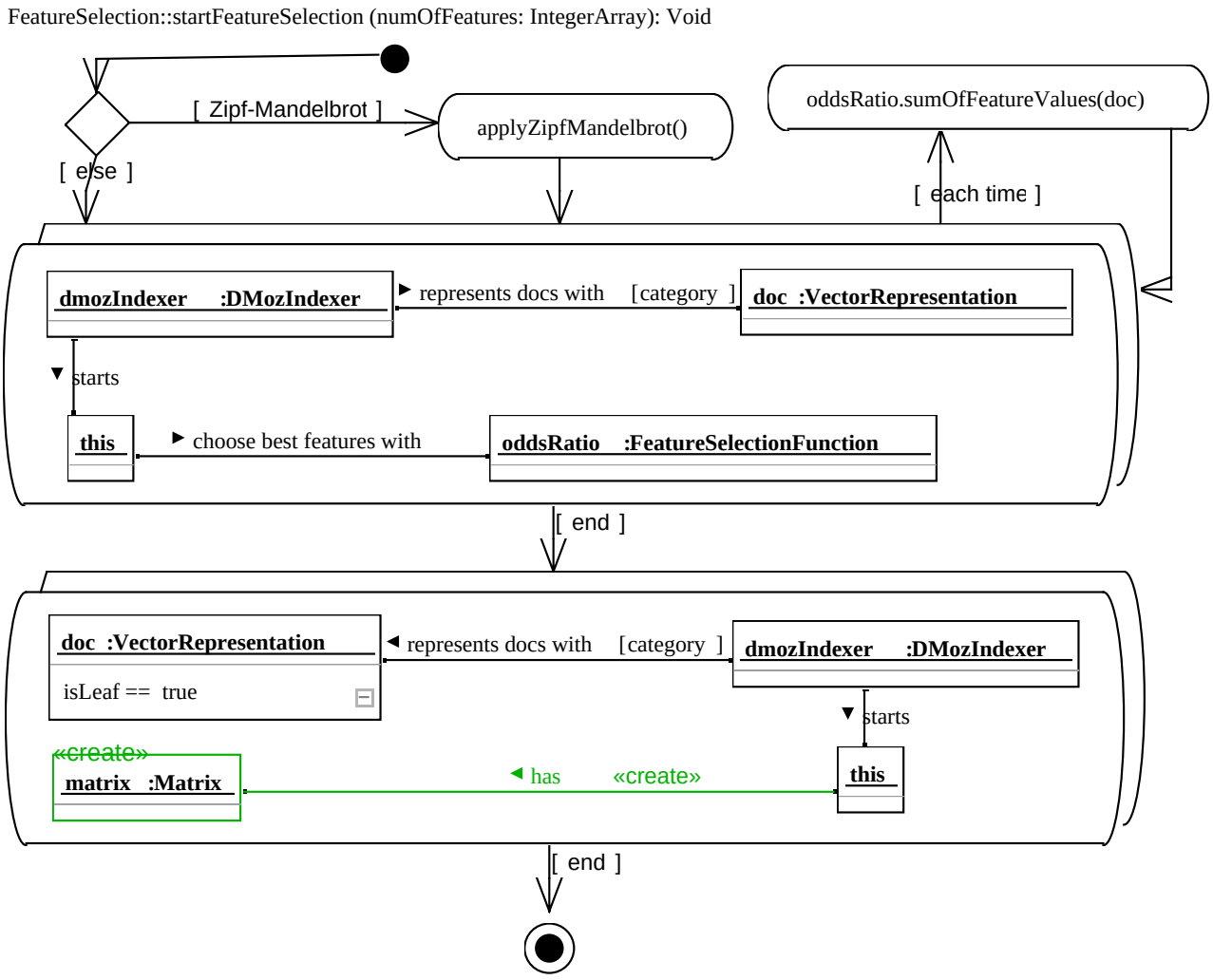


Abb. 5.22.: *Aktivitätendiagramm für die Methode* `startFeatureSelection` *der Klasse* `FeatureSelection`*. Die Merkmalsauswahl Odds Ratio berechnet für jedes Dokument aller Kategorien die besten Merkmale. Die Dokumente werden mit ihren Merkmalen in Matrizen.*

5.7. Grafische Benutzungsschnittstelle

Für das Durchführen der Experimente dieser Arbeit gibt es eine grafische Benutzeroberfläche, mit der sich grundlegende Parametrisierungen festlegen lassen. Da es eine Vielzahl unterschiedlicher Einstellungen gibt, ist der Benutzer auf eine effiziente Unterstützung durch eine grafische Benutzungsschnittstelle angewiesen. Die implementierte Benutzungsschnittstelle hat zu diesem Zweck besondere Eigenschaften:

- Die nötigen Schritte zur Durchführung eines Experiments sind von der Benutzungsschnittstelle aus steuerbar. Dabei wird deren Reihenfolge durch ihre grafische Anordnung wiedergespiegelt.

- Alle Parameter werden zusätzlich durch Tooltips erläutert.

- Pfade können über einen Auswahldialog festgelegt werden. Weitere Pfade in anderen Feldern werden dann durch die Benutzungsschnittstelle mit sinnvollen Vorschlägen belegt.

- Erstellte Dateien werden mit einer eindeutigen Dateiendung ergänzt.

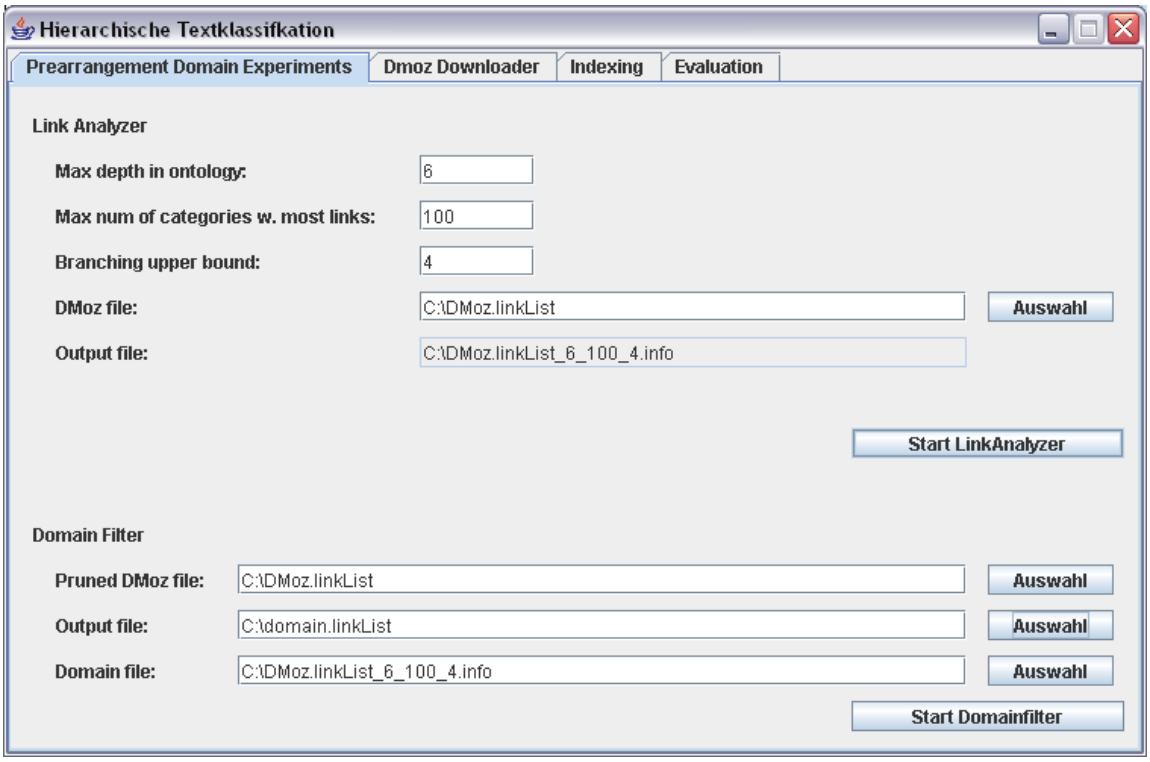
- Die Auswahl von Dateien filtert die in Frage kommenden Dateien.

- Parameterfelder sind mit Defaults belegt.

Mit der Benutzungsschnittstelle lassen sich nicht alle Einstellungen und Funktionen des Frameworkes nutzen. Der Benutzer sollte nicht überfordert sein, indem er aus einer unnötigen Vielzahl an Parametern wählen muss. Die Parameter in der Benutzungsschnittstelle sind etwa solche, die zum einen im Experimentaufbau der Arbeit verändert werden dürfen und zum anderen zu flexiblen Varianten führen. Es lässt sich beispielweise nicht in der Benutzungsschnittstelle festlegen, welche Stoppwortliste für den Indexierungsprozess oder welcher Reduktionsalgorithmus verwendet werden soll. Da es Parameter gibt, deren Untersuchung nicht im Fokus dieser Arbeit liegt, tauchen diese auch nicht in der Benutzungsschnittstelle auf und sind im Framework festgelegt. Das bietet den Vorteil, dass die Komplexität der Variationen in den Experimenten beschränkt wird.

In den nächsten Abschnitten werden die einzelnen Funktionen der grafischen Oberfläche erläutert.

5.7.1. Prearrangement Domain Experiments



In dieser Arbeit werden Experimente gemacht, die sich teilweise auf spezielle Fachgebiete beziehen. Hierfür wird ein einzelner Teilbaum der DMoz-Ontologie entnommen. Dieser Teilbaum soll bestimmten Anforderungen genügen. Er sollte möglichst viele Kategorien haben, die eine große Menge an Dokumenten beinhalten. Alle Blattkategorien müssen in einer vorgegeben Tiefe liegen und der Verzweigungsgrad der inneren Knoten darf einen Grenzwert nicht überschreiten.

Die Ansicht `Prearrangement Domain Experiments` ermöglicht es, einen solchen Teilbaum zu extrahieren. Der `Link Analyzer` untersucht die Linkstruktur der DMoz-Ontologie, mit den Parametern in **Tab. 5.1**. In **Abb. 5.23** werden diese Parameter erläutert. Zeichnung 1 zeigt eine DMoz-Hierarchie. Die Anzahl der Dokumente ist in den zugehörigen Kategorien angegeben. In Zeichnung 2 ist für `Max depth in ontology` = 2 dargestellt, dass die Kategorien der Tiefe zwei alle Dokumente der Nachfolger beinhalten. Mit `Branching upper bound` = 2 behält jede Kategorie maximal zwei Nachfolger (Zeichnung 3). Dabei werden die Nachfolger mit den meisten Dokumenten selektiert. Zeichnung 4 zeigt, dass nur die in `Max num of categories w. most links` bestimmte Anzahl an Kategorien übernommen werden, wobei die Kategorien mit den meisten Links bevorzugt werden.

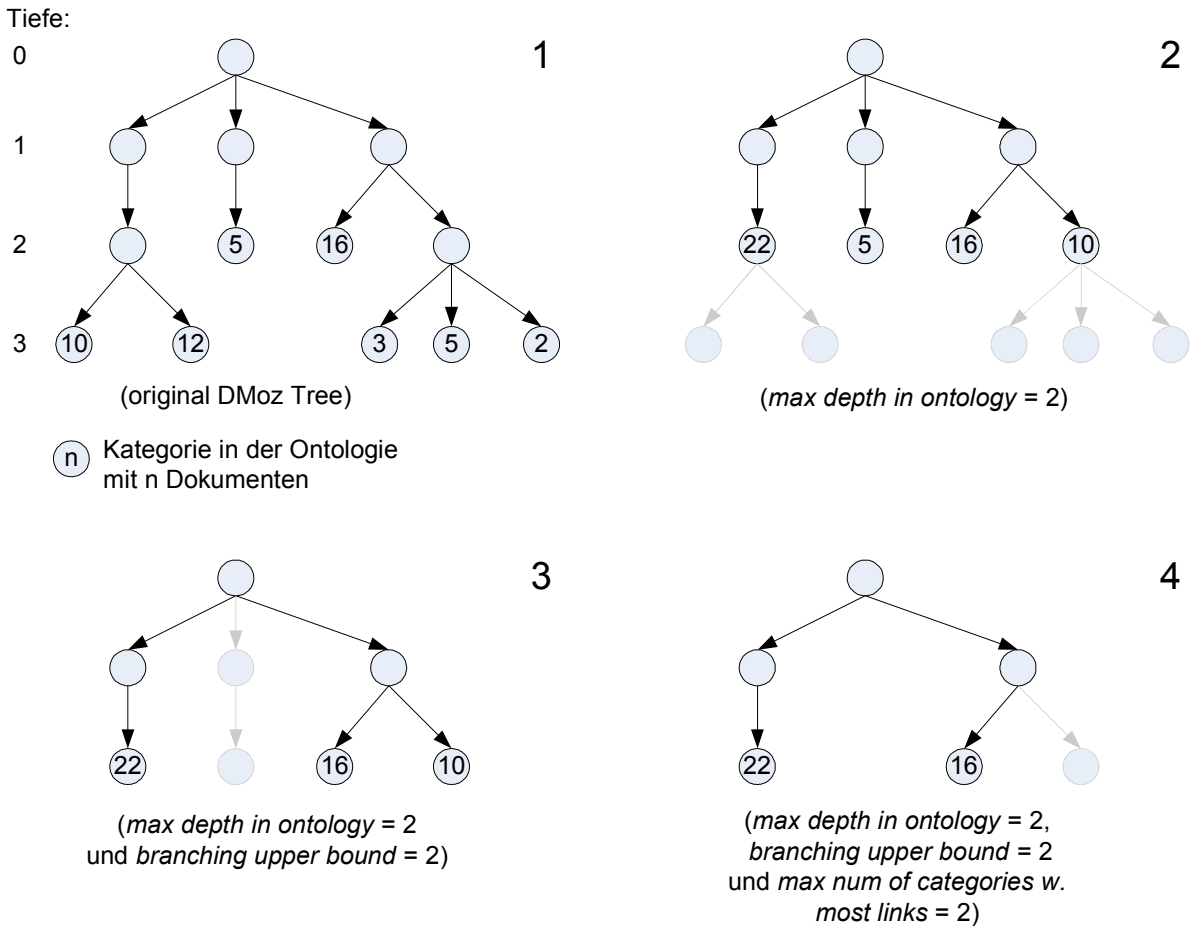


Abb. 5.23.: *Schrittweise wird die Hierarchie analysiert und den vorgegebenen Parametern angepasst: (1) Original Hierarchie, (2) Hierarchie mit Tiefenlimit, (3) Hierarchie mit Breitenlimit und (4) Auswahl eine festgelegten Anzahl von Kategorien.*

Parameter	Beschreibung	Default
`Max depth in ontology`	Tiefe, in der sich die Blattkategorien befinden sollen.	`6`
`Max num of categories w. most links`	Maximale Anzahl an Kategorien in der Ausgabe mit den meisten Links.	`100`
`Branching upper bound`	Obere Schranke für die Verzweigung innerer Knoten.	`4`
`DMoz file`	Pfad zur Ontologie, die untersucht werden soll.	
`Output file`	Datei, in der die Ausgabe gespeichert wird.	

Tab. 5.1.: *Parameter für* `Link Analyzer`.

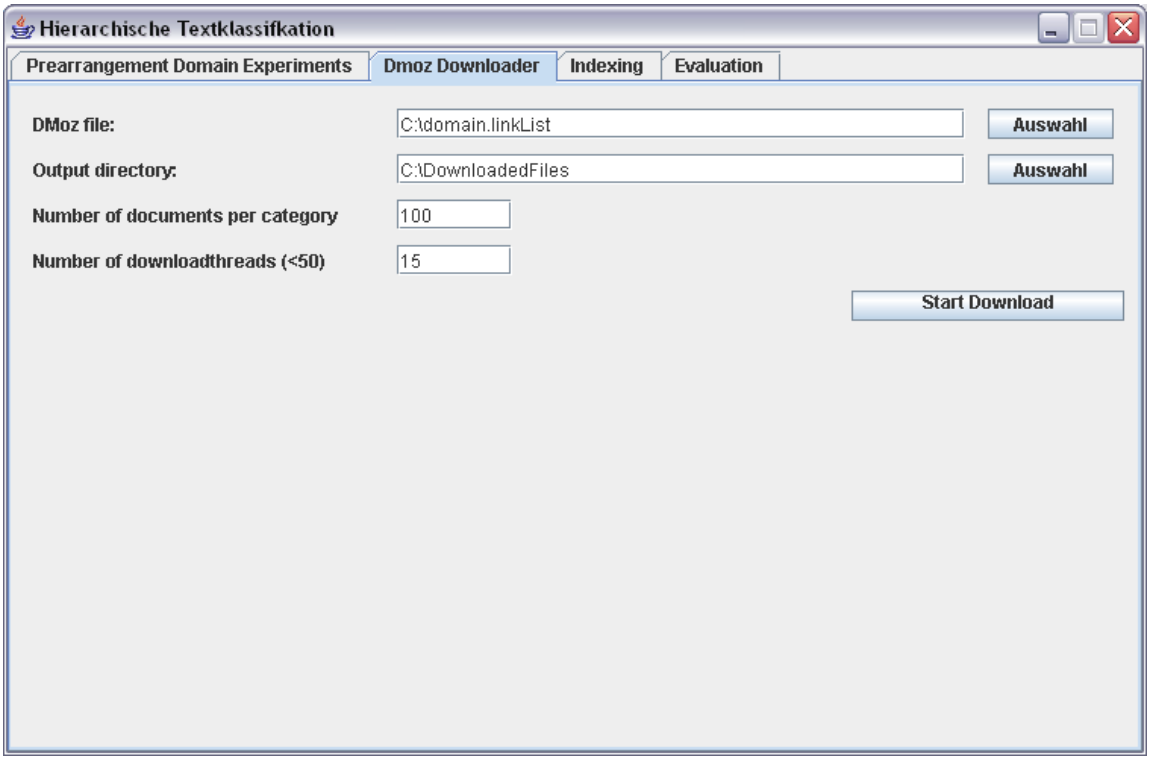
Die Ausgabe ist eine Datei, in der Kategorienamen aufgelistet sind, die den Anforderungen genügen. Als nächstes muss eine Liste mit den Links der Kategorien erstellt werden, die heruntergeladen werden sollen. Dies übernimmt der `Domain Filter`. Alle Links einer Kategorie werden zusammen mit ihren Namen in eine Datei geschrieben. Wichtig ist, dass die Ein-

Parameter	Beschreibung	Default
`Pruned DMoz file`	DMoz-Datei mit der Tiefe der ausgewählten Katgorien in `Domain file`.	
`Output file`	Datei in der die ausgewählten Kategorien mit ihren Links gespeichert werden.	
`Domain file`	Datei mit einer Auswahl von Kategorien.	

Tab. 5.2.: *Parameter für* `Domain Filter`.

gabedatei `Pruned DMoz file` genau die Tiefe der ausgewählten Kategorien in `Domain file` repräsentiert.

5.7.2. DMoz-Downloader

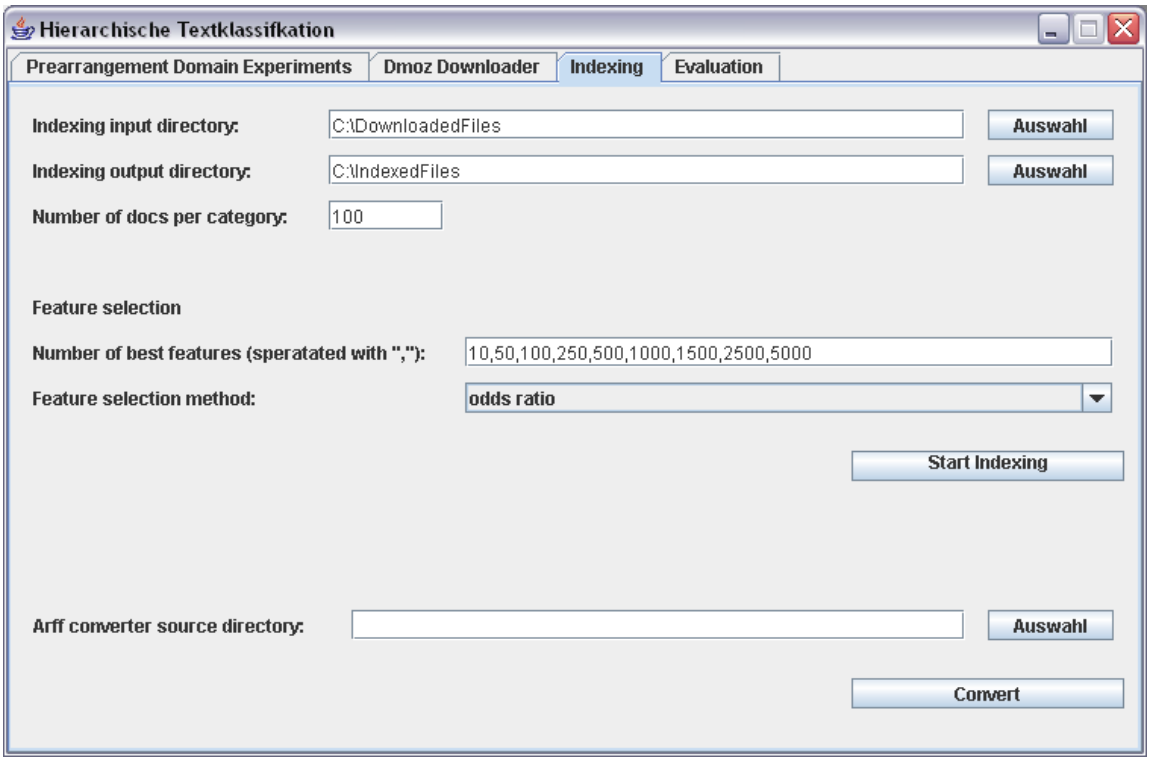


Der Downloader ermöglicht es, eine Liste von Links lokal zu speichern. Nachdem eine Auswahl von Kategorien mit Links getroffen wurde, können diese mit dem Downloader heruntergeladen werden. Dabei wird die Datei mit den Links übergeben und ein Zielordner festgelegt. Der Downloader legt für jede Kategorie eine Datei an, deren Zeilen jeweils ein Dokument darstellen. Wieviele Dokumente pro Kategorie heruntergeladen werden, wird durch `Number of documents per category` bestimmt. Sollte es nicht möglich sein für eine Kategorie die festgelegte Anzahl von Dokumenten herunterzuladen, werden die Kategorien im Zielordner mit „!!!*Kategoriename*“ gekennzeichnet. Dies kann vorkommen, wenn von vornherein nicht genügend Links vorhanden sind, ein Fehler beim Download auftritt oder wenn das Dokument nicht englisch ist. Um die Bandbreite der Internetverbindung besser auszunutzen bietet der Downloader die Möglichkeit mehrere Threads zu starten, die gleichzeitig verschiedene Dokumente herunterladen. Die Anzahl der Threads sollte dennoch nicht zu hoch gewählt werden.

Parameter	Beschreibung	Default
`DMoz file`	Datei vom Typ „.linkList“, die die heruntergeladenen Kategorien mit ihren Links beinhaltet	
`Output directory`	Verzeichnis, in dem die heruntergeladenen Katgorien gespeichert werden.	
`Number of documents per category`	Anzahl der Dokumente, die pro Kategorien heruntergeladen werden.	`100`
`Number of download threads`	Anzahl der Threads, die zum herunterladen verwendet werden.	`15`

Tab. 5.3.: *Parameter für* `Downloader`.

5.7.3. Indexing



Der Indexierer verarbeitet die Kategorien im angegebenen Ordner und erstellt als Ausgabe für jede Blattkategorie eine Matrix, in der jede Spalte ein Dokument darstellt. Die Zeilen repräsentieren die ausgewählten Merkmale. Der Indexierungsvorgang entfernt die Stoppwörter

und transformiert die Wörter in ihre Grundform, die dann in einem Index aufgenommen werden. Für jedes Dokument wird ermittelt, wie häufig ein Merkmal im Text vorkommt. Die Merkmale, die insgesamt in allen Dokumenten weniger als zweimal vorkommen, werden nicht weiter untersucht. Für alle anderen wird mit der ausgewählten Merkmalsauswahlmethode bestimmt, welche Merkmale sich am besten zum Klassifizieren eignen. Es lassen sich in einem Durchlauf mehrere Matrizen mit unterschiedlicher Anzahl von Merkmalen generieren, die jeweils in einem nach der Anzahl der Merkmale genannten Ordner gespeichert werden. `Number of best features` gibt eine Liste an, wieviele Merkmale übernommen werden sollen. Die Einträge müssen mit einem Komma und ohne Leerzeichen getrennt sein. Für den Eintrag `Number of best features` = 5, 10, 20 werden drei Ordner erstellt, deren Inhalt indexierte Kategorien sind.

Parameter	Beschreibung	Default
`Indexing input directory`	Verzeichnis mit den heruntergeladenen Kategorien, die indexiert werden sollen.	
`Indexing output directory`	Verzeichnis, in dem die indexierten Katgorien gespeichert werden.	
`Number of docs per category`	Anzahl der Dokumente, die pro Kategorien indexiert werden.	`100`
`Number of best features`	Anzahl der besten Merkmale, die zur Indexierung verwendet werden.	`100, 250, 500`
`Feature selection method`	Auswahlverfahren für Merkmale.	`odds ratio`

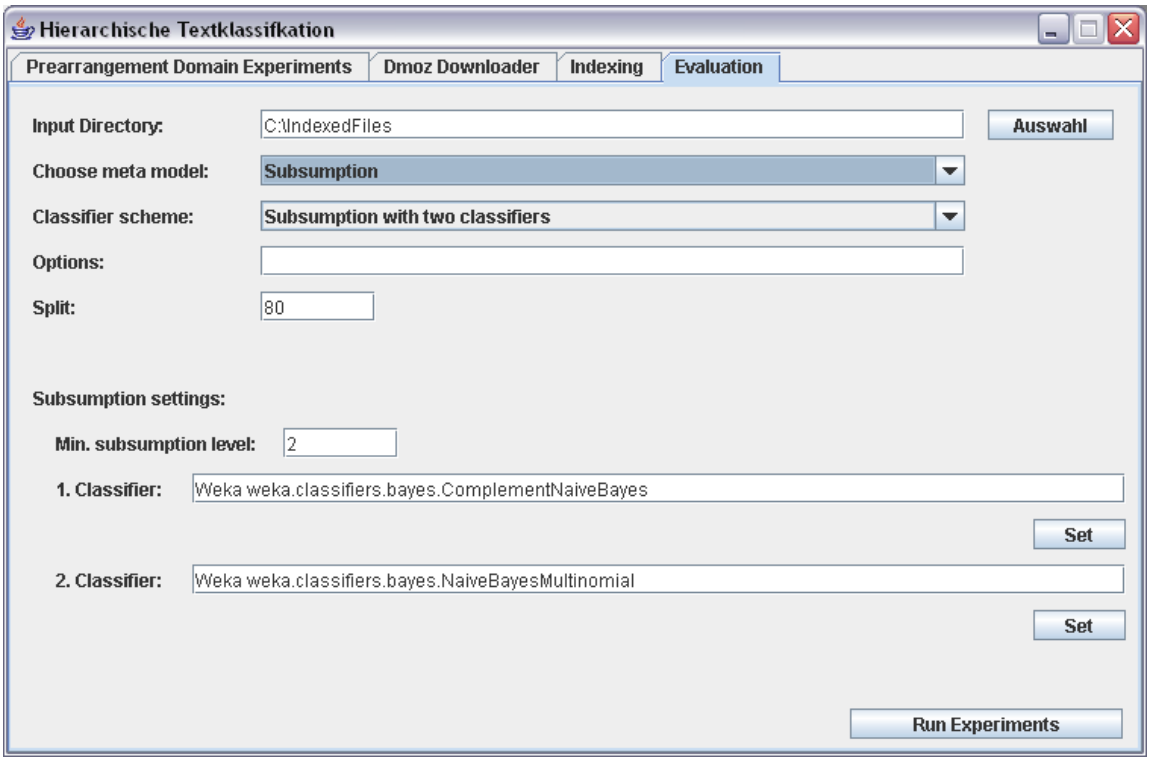
Tab. 5.4.: *Parameter für* `Indexer`.

Der Arff-Converter transformiert indexierte Kategorien in das Weka-kompatible Arff-Format. Für jede Kategorie in der Hierarchie wird eine Arff-Datei erzeugt. Weka hat eine Umgebung implementiert mit der Klassifizierer untersucht werden können, jedoch keine hierarchischen Verfahren. Die Umgebung kann Arff-Dateien lesen und aus den Daten Klassifizierer erzeugen und evaluieren. Damit können einzelne Klassifizierer in einem hierarchischen Verfahren beurteilt werden. Für die Evaluierung eines hierarchischen Klassifikationsmodells mit mehreren Klassifizierern in dem Tool dieser Arbeit ist keine Arff-Transformation notwendig.

Parameter	Beschreibung	Default
`Source directory`	Verzeichnis mit indexierten Kategorien, die in Arff-Dateien konvertiert werden.	

Tab. 5.5.: *Parameter für* `Arff-Converter`.

5.7.4. Evaluation



In der Evaluierungsansicht lassen sich indexierte Kategorien, die zu einer Hierarchie gehören, mit verschiedenen Klassifikationsmodellen und Klassifizierern untersuchen. Mit `Choose meta model` lassen sich die verschiedenen implementierten Klassifikationsmodelle auswählen, und mit `Classifier scheme` können die dazugehörigen Klassifikationsalgorithmen verwendet werden. Zu den Klassifikationsalgorithmen lassen sich weitere Optionen angeben. Beispielsweise muss bei Auswahl von AdaBoostM1 angegeben werden, welche Klassifizierungsalgorithmen

verwendetet werden sollen und einige weitere Parameter. Für AdaBoostM1 ist per Default Complement Naïve Bayes eingestellt. Bei allen Weka-Klassifizierern lassen sich zusätzliche Parameter bestimmen, die in dem Feld `Options` eingetragen werden können. Informationen über mögliche Parameter und weiters sind der Weka-Dokumentation und dem Buch [WF05] zu entnehmen.

Das Verhältnis mit dem die Dokumente der Eingabe in Traings- und Testdokumente zur Evaluierung geteilt werden, wird mit `Split` festgelegt. Für die Klassifikationsmodelle „Weka“ und „Naïve Bayes “ kann die Evaluierung nun gestartet werden. Möchte man „Subsumption“

Parameter	Beschreibung	Default
`Input directory`	Verzeichnis mit indexierten Kategorien.	
`Choose meta model`	Auswahl eines implementierten Klassifikationsmodells.	
`Classifier scheme`	Auswahl eines Klassifizierungsalgorithmus	
`Options`	Optionen für den Klassifizierungsalgorithmus.	
`Split`	Verhältnis der Anzahl Trainings- zu den Testdokumenten.	

Tab. 5.6.: *Parameter für* `Evaluation`.

verwenden, sind zusätzlich Parameter festzulegen. In diesem Tool ist ein Subsumptionprototyp umgesetzt worden, der mit zwei Klassifizierern arbeitet und dem Modell in Abschnitt 4.4 entspricht. Durch das Auswählen von `Choose meta model`, `Classifier scheme` und `Options` lässt sich ein Klassifizierer definieren und mit dem Button „Set“ dem ersten bzw. zweiten Subsumptionklassifizierer zuweisen. In dem Feld `Min. subsumption level` muss festgelegt werden, welche minimale Tiefe in der Hierarchie durch ein Label repräsentiert werden muss. Bevor die Evaluierung gestartet werden kann, muss in `Choose meta model` „Subsumption“ ausgewählt werden. Je nach Experiment, wählt die Benutzungsschnittstelle automatisch die relevanten Evaluierungsmaße aus. Für jedes Maß wird nach Beenden der Evaluierung ein Fenster mit den Ergebnissen geöffnet.

Parameter	Beschreibung	Default
`Min. subsumption level`	Minimale Tiefe einer Klassifikation, die erreicht werden muss.	`2`
`1. Classifier`	Auswahl des ersten Klassifizierers für Subsumption.	
`2. Classifier`	Auswahl des zweiten Klassifizierers für Subsumption.	

Tab. 5.7.: *Parameter für* `Subsumption`.

6. Experimente und Ergebnisse

In dieser Arbeit werden verschiedene Methoden und Algorithmen vorgestellt, die in der Literatur besonders gute Ergebnisse in der Textklassifikation erzielen, vgl. Abschnitt 4.2. Die Klassifikationsergebnisse sind von der Art der Dokumente und vom Wissensgebiet der Kategorien abhängig. Ist von einem speziellen Wissensgebiet die Rede, so wird dies als Domäne bezeichnet.

Die hierarchische Textklassifikation wird hier in zwei unterschiedlichen Experimentanordnungen untersucht:

- Domänenspezifische Kategorisierung
- Abstrakte Kategorisierung

Die Experimente zur domänenspezifischen Kategorisierung stellen heraus, wie sich die Klassifikationsgenauigkeit verhält, wenn Dokumente in Domänen eingeordnet werden. Das Ergebnis zeigt, ob sich hierarchische Textklassifikation zur externen Topic Indentification eignet. Externe Topic Identification ermöglicht es für Dokumente abstrakte Label zu generieren, vgl. Abschnitt 2.3. Die Experimente zur abstrakten Kategorisierung verdeutlichen, wie gut abstrakte Label für Dokumente gefunden werden können.

Die Vorbereitungen und Parameter, die für beide Experimentanordnungen gleich sind, werden im Folgenden erläutert.

6.1. Vorbereitung

DMoz-Anpassung Diese Arbeit verwendet die DMoz-Datei „content.rdf.u8.gz“[1], die im DMoz-Webseitenarchiv[2] zu finden ist. Die Datei repräsentiert die Struktur sowie die

[1] Die Datei wurde am 06. Juni 2006 erstellt.
[2] http://rdf.dmoz.org/rdf/archive/

Links zu den Dokumenten der Kategorien in DMoz. Zusätzlich gibt es zu jedem Link eine kurze Inhaltsangabe.

Da die Struktur der Ontologie und die Links der Kategorien für die Experimente relevant sind, lässt sich die Größe der DMoz-Datei von 2 GB auf 358 MB reduzieren, indem Informationen wie Titel und Inhaltsangaben nicht übernommen werden. Die Klasse `RDFMinimizer` besitzt die entsprechende Funktionalität.

DMoz-Beschränkung Ein hierarchischen Klassifikationsverfahren benötigt für jede Kategorie in der Ontologie einen eigenen Klassifizierer. Für die Experimente werden 100 bis 150 Dokumente zum Trainieren der Klassifizierer verwendet. Da die Kategorien jedoch selten eine so hohe Dichte an Dokumenten besitzen, wird die Hierarchie auf eine festgelegte Tiefe begrenzt. Die unter dieser Tiefe liegenden Dokumente zählen zu ihren Vorgängern in der festgelegten Tiefe. Die Klasse `DMozPruning` ermöglicht dies.

Download Für unsere Arbeit beschränken wir uns auf englische Dokumente. Während des Downloads eines Dokumentes, wird dessen Sprache analysiert. Wird festgestellt, dass es sich nicht um ein englischsprachiges Dokument handelt, wird das Dokument verworfen. Ein einfacher Test prüft, ob die Wörter „is“ und „the“ häufiger als dreimal im Dokument enthalten sind. Falls ja, wird angenommen, dass das Dokument englischsprachig ist.

6.2. Allgemeine Parameter

Zipf-Mandelbrot In vielen Arbeiten zur Textklassifikation wird die Zipf-Mandelbrot-Regel angewandt, um die Anzahl der Merkmale zu reduzieren. Diese Regel besagt, dass ein Merkmal in mindestens n Kategorien auftauchen muss. Für verschiedene n wurden Versuche gemacht, die herausstellen, dass sich die Klassifikationsergebnisse verschlechtern. Das Verfahren Zipf-Mandelbrot wird in den Experimenten nicht angewendet.

Merkmalsauswahl Als Verfahren für die Auswahl der Merkmale wird Odds Ratio aus Abschnitt 3.1.2 verwendet.

Splitting-Einstellung Das Splitting-Verhältnis ist auf 80% festgelegt; in den Experimenten werden 80% der Dokumente zum Trainieren der Klassifizierer verwendet und 20% zum Testen.

Durchläufe Für die Evaluierung werden die Durchschnittswerte aus 10 Durchläufen ermittelt, wobei vor jedem Durchlauf die Trainings- und Testdokumente zufällig zusammengestellt werden.

Subsumptionseinstellung Im Rahmen dieser Arbeit wurde das Subsumptionsverfahren in Abschnitt 4.4 auf zwei Klassifizierer beschränkt. Damit sich die Genauigkeit der einzelnen Klassifizierer nicht auf Grund einer zu geringen Anzahl von Trainingsdokumenten verändert, werden beide Klassifizierer mit den gleichen Trainingsdokumenten trainiert. Für die Subsumption ist es erforderlich, dass die Klassifizierer teilweise unterschiedliche Ausgaben liefern. Aus diesem Grund sind zwei verschiedene Klassifizierer auszuwählen.

AdaBoostM1-Optionen Für das AdaBoostM1-Verfahren werden Complement-Naïve-Bayes-Klassifizierer verwendet.

Grenzwert für Multi-Pfad Der Grenzwert wurde auf 0.02 festgelegt. Alle Klassifikationen, die ein Dokument mit einer Wahrscheinlichkeit größer als 0.02 zu einer Kategorie zuordnen, werden weiter verfolgt.

6.3. Domänenspezifische Kategorisierung

Mit domänenspezifischer Kategorisierung meint man die Zuordnung von Dokumenten innerhalb einer speziellen Domäne. Dabei soll die Klassifikation möglichst genaue Informationen über die Eingabe liefern. Die DMoz-Ontologie ist so aufgebaut, dass je tiefer eine Kategorie in der Ontologie liegt, desto spezialisierender ist sie. Um also viel Wissen über das Thema eines Dokumentes zu erlangen, muss es in eine tief liegende Kategorie klassifiziert werden.

Die DMoz-Struktur wird in **Abb. 2.2** gezeigt. Eine Domäne ist in diesem Fall eine Kategorie der Tiefe eins mit ihren Nachfolgern, beispielweise Top/Arts. Die **Abb. 6.1** zeigt ein Beispiel.

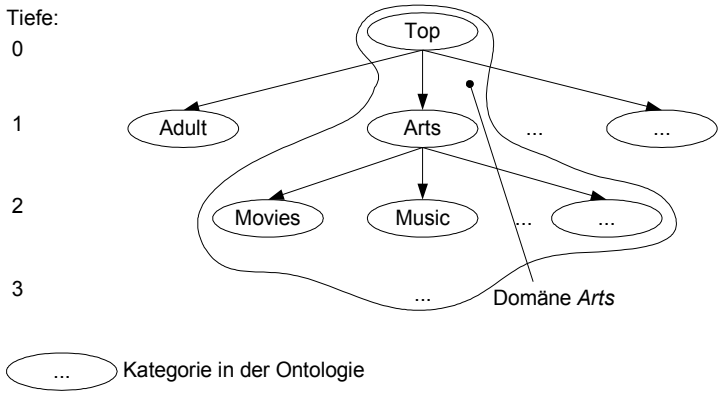


Abb. 6.1.: *Beispiel für eine ausgewählte Domäne in Tiefe 1. Die Domäne „Arts“ wird erstellt, indem alle Nachfolger der Kategorie „Top/Arts“.*

6.3.1. Auswahl der Domänen

Für die Versuche unterschiedlicher Klassifikationsverfahren auf Basis einzelner Domänen, wird eine spezielle Auswahl getroffen. Domänen mit besonders vielen Dokumenten sind Top/World > Top/Regional > Top/Arts > Top/Society und (>) Top/Science. Die Domänen Top/World und Top/Regional eignen sich nicht besonders für die Experimente, da hier viele verschiedene Sprachen vorkommen, aber nur englischsprachige Dokumente in den Experimenten verarbeitet werden. Deshalb ist die Auswahl auf drei Domänen gefallen:

- Top/Arts (im Folgenden nur noch Arts genannt) **Abb. 6.2**
- Top/Science (Science) **Abb. 6.3**
- Top/Society (Society) **Abb. 6.4**

6.3.2. Domänenstatistiken

Die ausgewählten Domänen werden in diesem Abschnitt genauer untersucht. Es ist festzustellen, dass in jeder Kategorie die Anzahl der kumulierten Dokumente und Kategorien in den Tiefen 4, 5 und 6 stark abfällt und in den Tiefen 0 bis 3 fast konstant bleibt, was bedeutet, dass die Kategorien erst ab Tiefe 3 Dokumente beinhalten (vgl. Abb. Dokumente pro Ebene).

Auffällig ist, dass die Kategorien Arts und Society die meisten Dokumente in relativ speziellen Kategorien haben (Tiefe 5, 6 und Tiefe 8), während der Großteil der Dokumente in Science in Tiefe 4 liegt. Daraus lässt sich folgern, dass sich die Dokumente mit dem Thema „Science“ relativ eindeutig zuordnen lassen. Die Dokumente der beiden anderen Domänen unterscheiden sich noch in großer Tiefe inhaltlich so stark, dass sie in verschiedene Kategorien müssen.

Die Anzahl der Kategorien und Dokumente ist in den drei Domänen sehr unterschiedlich: Arts besitzt die meisten Kategorien und Dokumente, gefolgt von Society. Science besitzt mit Abstand die wenigsten Dokumente und Kategorien. Die durchschnittliche Anzahl der kumlierten Dokumente pro Knoten hat weniger starke Abweichungen. Das Verhältnis zwischen der Anzahl der Kategorien und der Dokumente ist sehr ähnlich.

Für die Evaluierung der Klassifizierer mit den Dokumenten einer Domäne ist es wünschenswert, eine möglichst große Tiefe in der Domäne zu erlangen. Dennoch sollten genügend Dokumente vorhanden sein (etwa 100). Aus diesem Grund wird für alle Domänen Tiefe 5 ausgewählt, obwohl dort die durchschnittliche Anzahl kumulierter Dokumente bei ca.

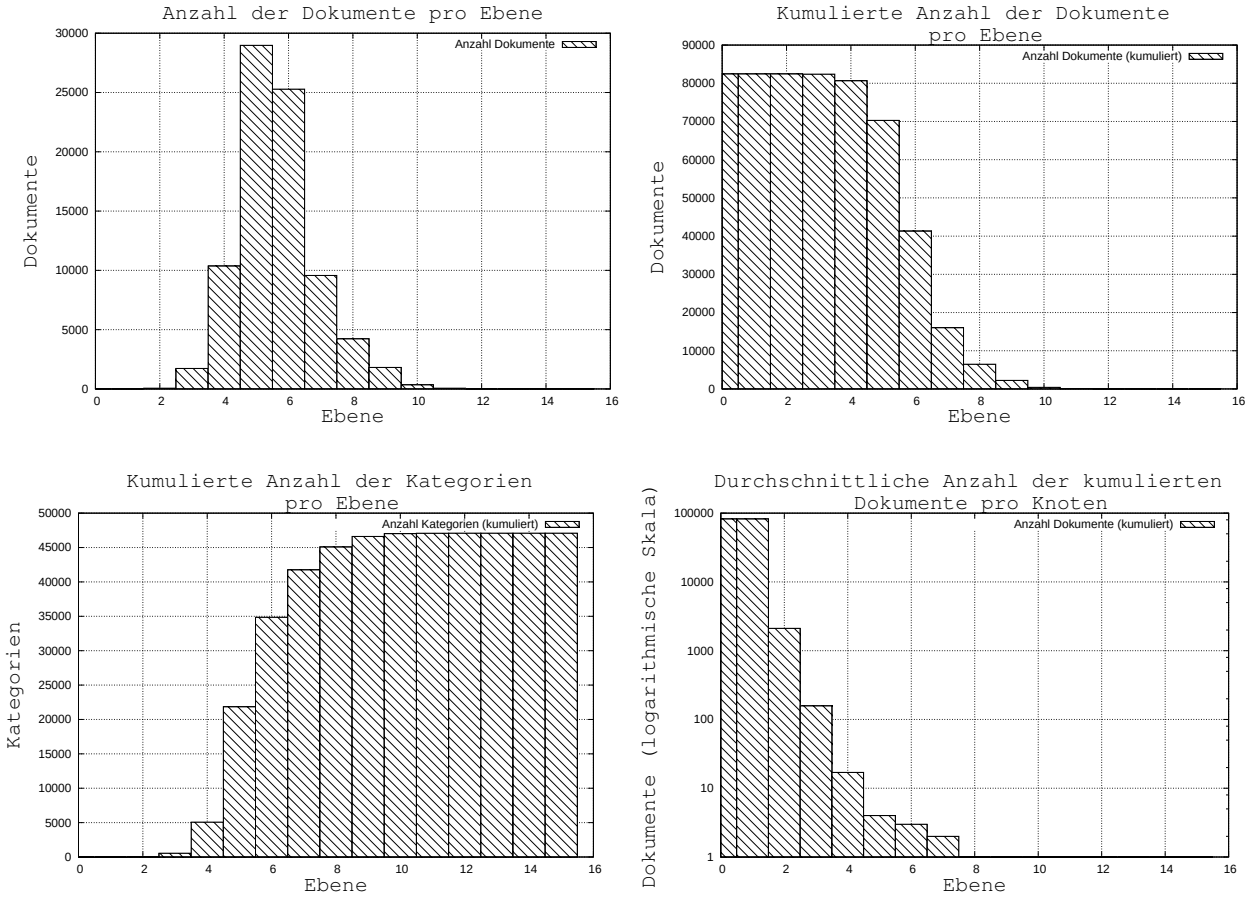


Abb. 6.2.: *Größenordnung und Beschaffenheit der Domäne Top/Arts. Dargestellt sind: Die Anzahl der Dokumente pro Ebene, die kumulierte Anzahl pro Ebene, die kumulierte Anzahl der Kategorien pro Ebene und die durchschnittliche Anzahl der kumulierten Dokumente pro Ebene. Die durchschnittliche Anzahl der kumulierten Dokumente pro Ebene ist entscheidend, wenn eine bestimmte Anzahl an Dokumenten pro Knoten verlangt wird.*

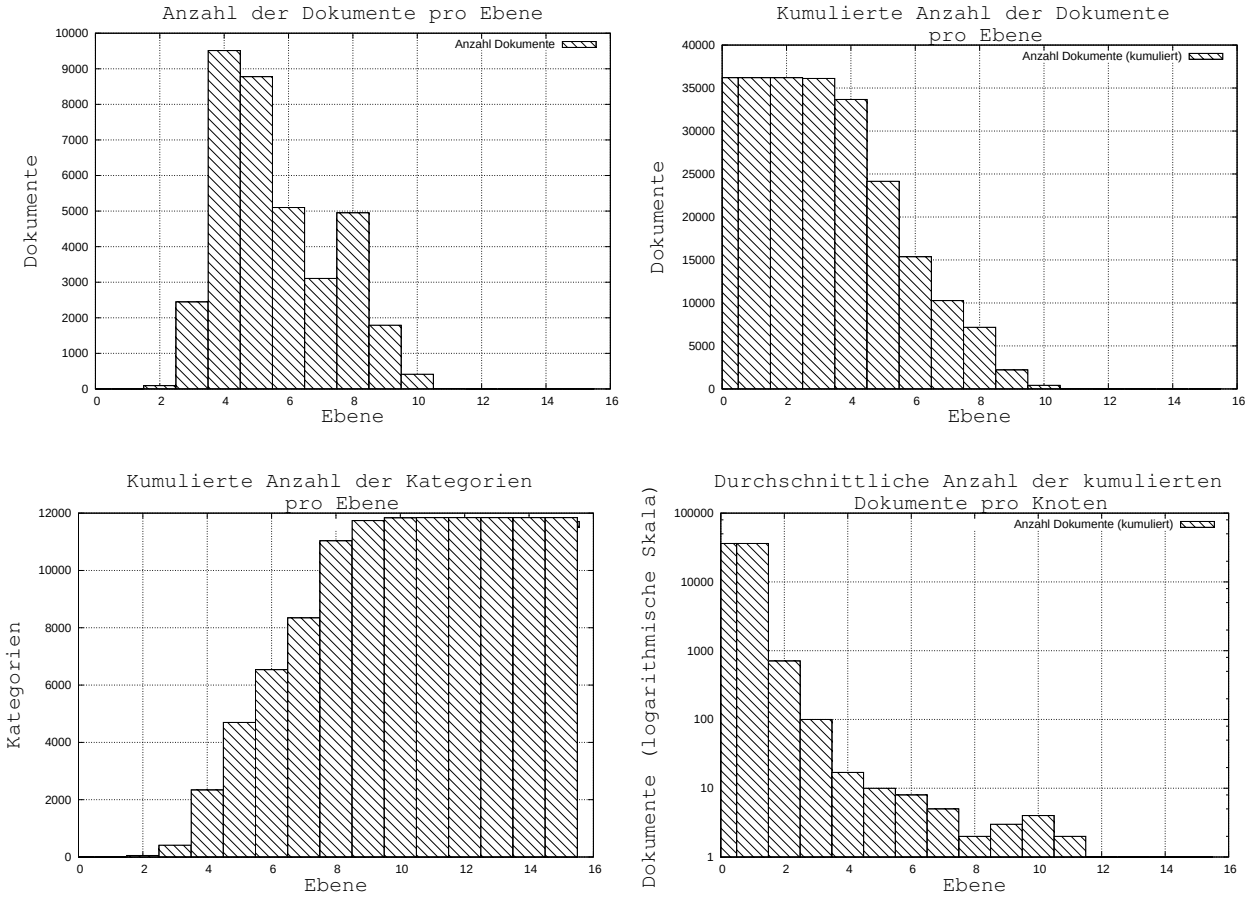


Abb. 6.3.: *Größenordnung und Beschaffenheit der Domäne Top/Science. Dargestellt sind: Die Anzahl der Dokumente pro Ebene, die kumulierte Anzahl pro Ebene, die kumulierte Anzahl der Kategorien pro Ebene und die durchschnittliche Anzahl der kumulierten Dokumente pro Ebene. Die durchschnittliche Anzahl der kumulierten Dokumente pro Ebene ist entscheidend, wenn eine bestimmte Anzahl an Dokumenten pro Knoten verlangt wird.*

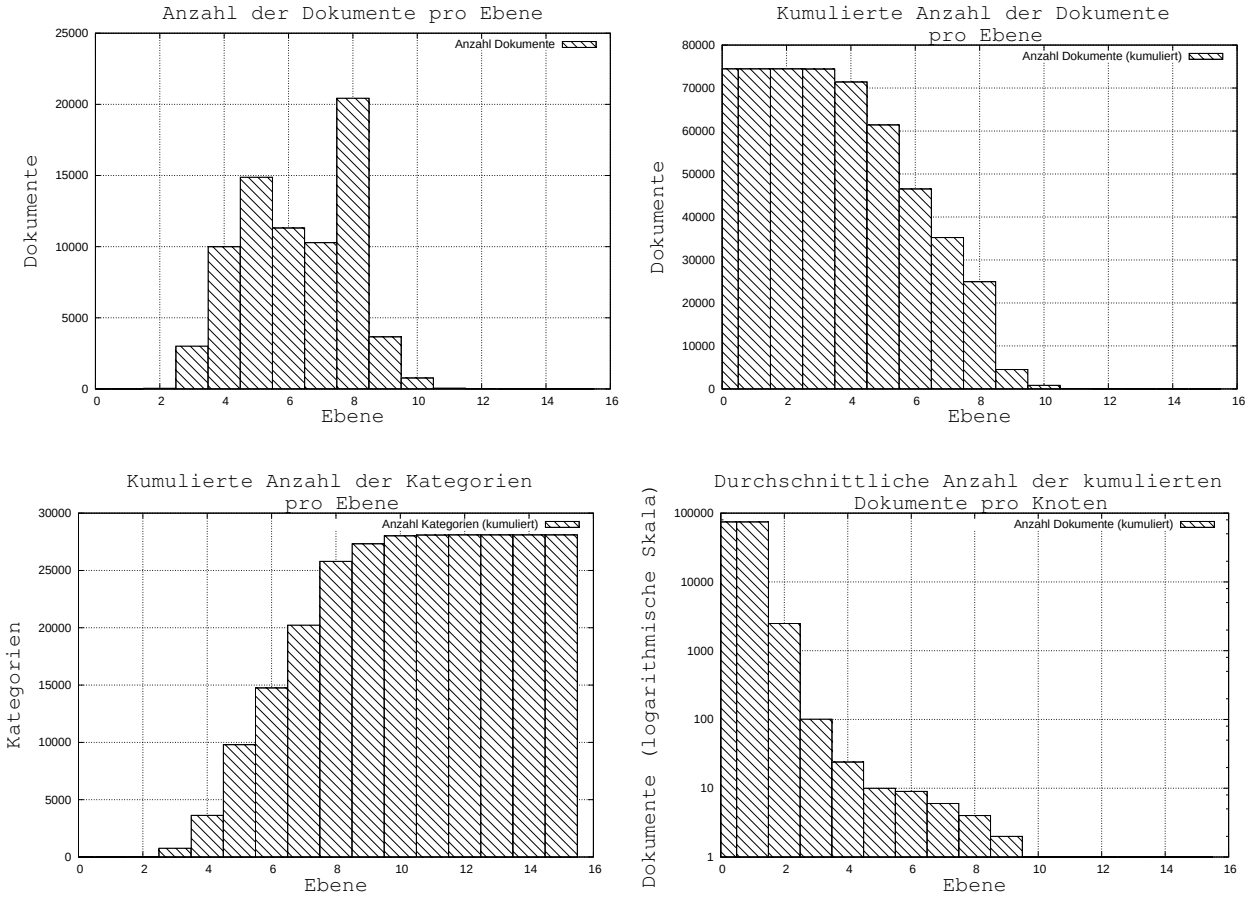


Abb. 6.4.: *Größenordnung und Beschaffenheit der Domäne Top/Society. Dargestellt sind: Die Anzahl der Dokumente pro Ebene, die kumulierte Anzahl pro Ebene, die kumulierte Anzahl der Kategorien pro Ebene und die durchschnittliche Anzahl der kumulierten Dokumente pro Ebene. Die durchschnittliche Anzahl der kumulierten Dokumente pro Ebene ist entscheidend, wenn eine bestimmte Anzahl an Dokumenten pro Knoten verlangt wird.*

10 liegt. Das hat zur Folge, dass sich nur wenig Kategorien mit 100 Dokumenten herunterladen lassen. Informationen zu den heruntergeladenen Kategorien finden sich in **Tab. 6.1**.

Domäne	Max. Verzweigung	# Blätterkategorien	Physikalische Größe	# Wörter
Arts	4	29	21,1MB	94 730
	5	36	25,5MB	93 895
Science	4	15	15,1MB	76 400
	5	15	15,8MB	77 520
Society	4	25	29,4MB	80 992
	5	33	39,4MB	97 102

Tab. 6.1.: *Informationen zum Download einer Auswahl von Domänen. Es wurden jeweils 100 Dokumente pro Kategorie heruntergeladen. Die maximale Tiefe der Domänen beträgt 5.*

6.3.3. Ergebnisse

Naïve-Bayes-Klassifizierer

Die für diese Arbeit entwickelten Klassifizierer werden hier mit unterschiedlichen Domänen getestet. Die Ergebnisse sind in **Abb. 6.5** dargestellt. Die besten Klassifikationsergebnisse werden in „Science“ mit einer Erfolgsrate von etwa 58% durch einen flachen Klassifizierer erzielt. Die hierarchischen Klassifikationsverfahren erreichen hier eine geringere Erfolgsrate. In den Domänen Society und Arts fallen die Ergebnisse schlechter aus, was offensichtlich mit der höheren Anzahl an Kategorien zusammenhängt. Auffällig ist, dass die Genauigkeit der Klassifizierer in Society stärker auseinander liegen als in anderen Domänen. Die Diagramme zeigen, dass das Shrinkageverfahren in allen Domänen zu schlechteren Ergebnissen kommt, als das hierarchische Naïve-Bayes-Verfahren.

Die besten Ergebnisse können mit einer Merkmalenanzahl zwischen 100 und 250 erzielt werden. Bei einer höheren Anzahl ist eine Verschlechterung zu erwarten.

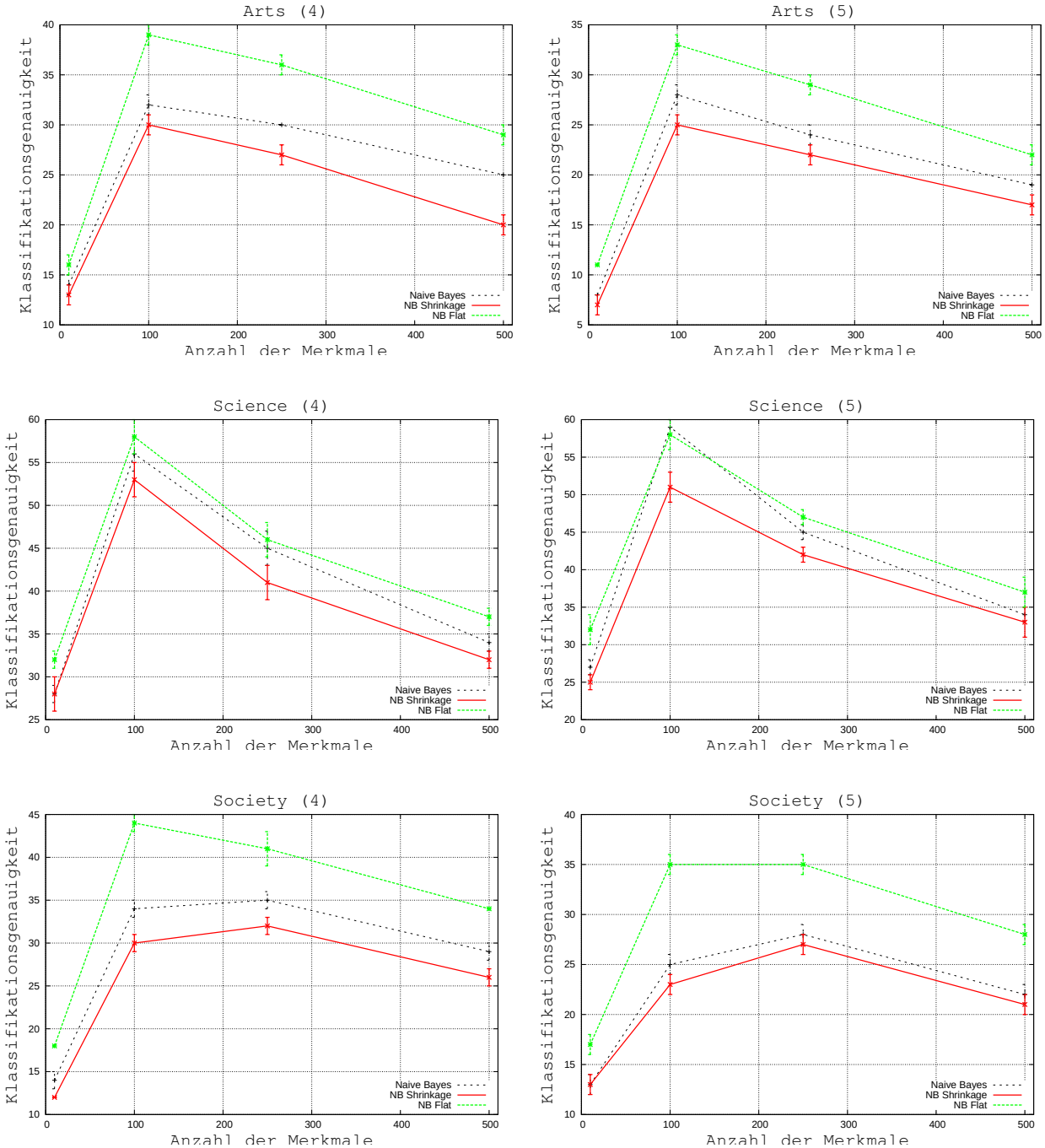


Abb. 6.5.: *Ergebnisse der Naïve-Bayes-Klassifizierer. Die Diagramme zeigen die Erfolgsraten (Klassifikationsgenauigkeit in Prozent) der Klassifizierer Naïve Bayes, Naïve Bayes Shrinkage und Naïve Bayes Flat in den drei Domänen Arts, Science und Society (jeweils mit den maximalen Verzweigungsgraden 4 und 5).*

Weka-Klassifizierer

Im Folgenden werden Weka-Klassifizierer in den drei Domänen untersucht. Die Ergebnisse sind in **Abb. 6.6** dargestellt. Wie bereits festgestellt, lässt sich auch hier in der Domäne „Science“ am besten klassifizierern. Es werden Erfolgsraten von etwa 85% erreicht. Dies ist durch eine hohe Anzahl von Merkmalen möglich. Diese Rate wird durch Verfahren wie Complement Naïve Bayes, Naïve Bayes Multinomial und AdaBoostM1 erlangt. Die einfache Naïve-Bayes-Implementierung erreicht bestenfalls 60%.

Im Gegensatz zu vorigen Untersuchungen wird mit dem Erhöhen der Anzahl der Merkmale die Klassifikationsgenauigkeit ebenfalls vergrößert. Vergleicht man die Weka Naïve-Bayes-Implementierung mit der Naïve-Bayes-Implementierung der Arbeit, stellt sich heraus, dass letztere eine deutlich höhere Erfolgsrate erreicht bei 100 Merkmalen.

AdaBoostM1 erzielt in „Arts“ mit maximal fünffacher Verzweigung in der Hierarchie mit kleinem Abstand die besten Ergebnisse. Ansonsten liegt es mit Complement Naïve Bayes und Naïve Bayes Multinomial etwa gleich auf.

Die Ausnahme bildet das Weka-Naïve-Bayes-Verfahren, das teilweise sogar weniger als die Hälfte der Erfolgsraten der anderen Verfahren erreicht.

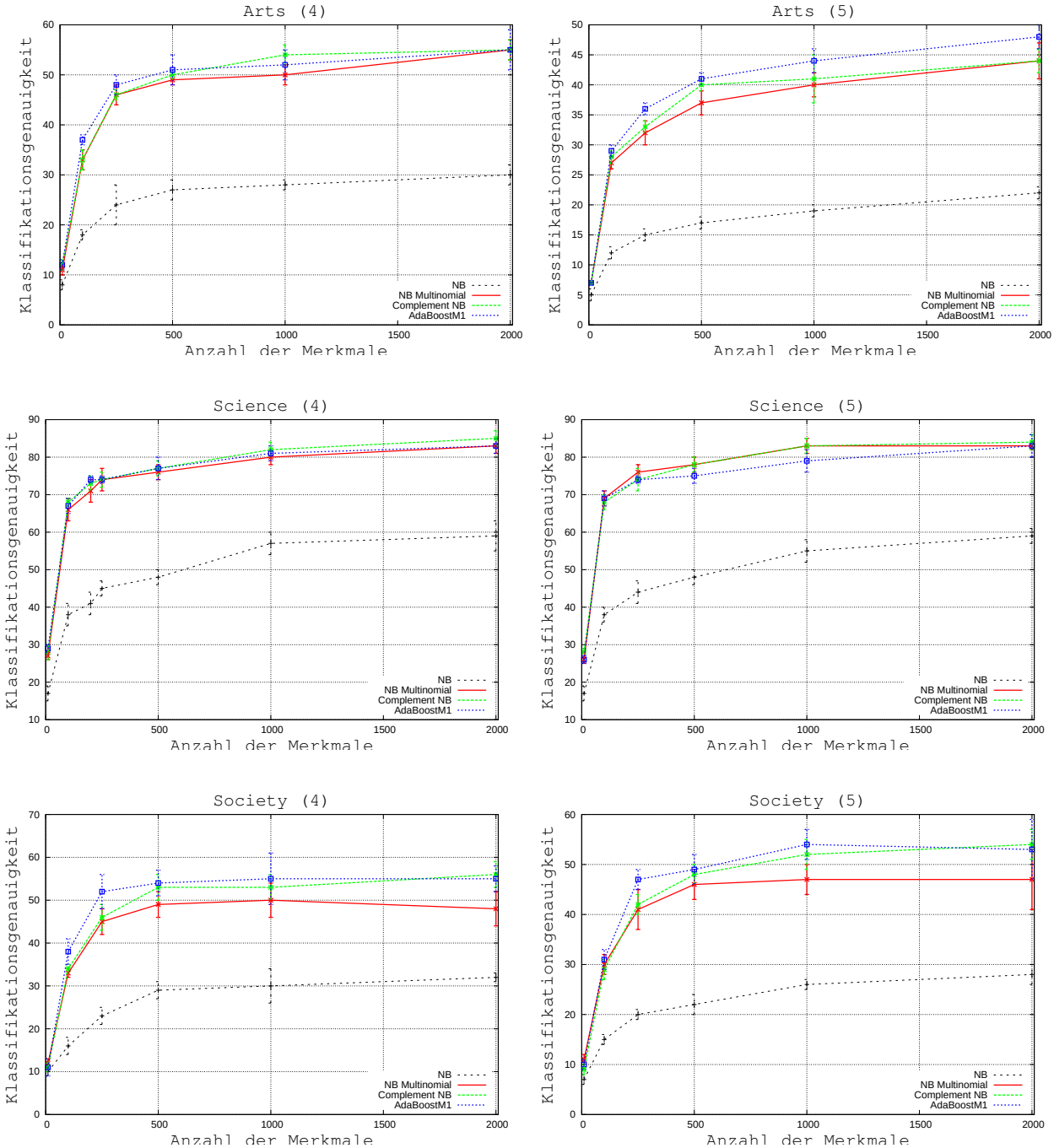


Abb. 6.6.: *Ergebnisse der Weka-Klassifizierer. Die Diagramme zeigen die Erfolgsraten (Klassifikationsgenauigkeit in Prozent) der Klassifizierer Naïve Bayes, Naïve Bayes Multinomial, Complement Naïve Bayes und AdaBoostM1 mit Complement Naïve Bayes in den drei Domänen Arts, Science und Society (jeweils mit den maximalen Verzweigungsgraden 4 und 5).*

Subsumption

Das Subsumptionsverfahren arbeitet mit zwei Klassifizierern. Das Verfahren wählt aus den Ergebnissen der Klassifizierer die speziellste, gemeinsame Kategorie aus. Sei die Subsumptionstiefe die Tiefe, in der die ausgewählte Kategorie liegt.

Für die Subsumptionsexperimente werden zwei besonders gute Klassifizierer ausgewählt: Naïve Bayes Multinomial und Complement Naïve Bayes. Die vorangegangenen Experimente zeigen, dass in der Domäne „Arts“ die Klassifizierer eine relativ geringe Genauigkeit erreichen, während die Genauigkeit in der Domäne „Science“ sehr hoch liegt. Auf Grund dieser Unterschiede werden die folgenden Versuche mit diesen beiden Domänen durchgeführt.

Die Bedeutung der Werte in **Tab. 6.2** und **Tab. 6.3** ist folgendermaßen festgelegt: Der Wert in der Zeile „Gesamt“ gibt an, wieviel Prozent der klassifizierten Dokumente richtig klassifiziert werden. Die darauf folgenden Werte $n \in \{2 \dots 5\}$ zeigen die Erfolgsrate Subsumptionstiefe n. Die Subsumptionstiefen 0 und 1 werden nicht betrachtet, da innerhalb der domänenspezifischen Experimente erst ab Tiefe 2 klassifiziert wird. Durch das Subsumptionsverfahren werden die Dokumente in unterschiedliche Tiefen der hierarchie Klassifiziert. Die Verteilung der Dokumente bezüglich der Tiefen in der Hierarchie ist in der Tabelle in den Klammern angegeben. Beispielweise wurden in der Domäne „Arts“ mit 100 Merkmalen 24% der Testdokumente in die Tiefe 2 klassifiziert.

Bei 100 Merkmalen in der Domäne „Arts (5)“ —vgl. Tab. 6.2— werden ca. 30% der Klassifikationsergebnisse subsumiert; 24% in Tiefe 2, 4% in Tiefe 3 und 2% in Tiefe 4. Die Gesamterfolgsrate des Subsumptionsverfahrens wird gegenüber Complement Naïve Bayes um etwa 5% verbessert. Mit Subsumption beträgt die Genauigkeit 46,74% und mit dem Complement-Naïve-Bayes-Verfahren ohne Subsumption beträgt sie 28,21%, was einer Verbesserung von etwa 5% entspricht.

Eine weitere Auffälligkeit ist, dass 70% der Testdokumente mit der Tiefe 5 klassifiziert werden, wovon etwa 37% korrekt sind. Das bedeutet, dass etwa 26% aller Testdokumente in Tiefe 5 korrekt zugeordnet werden. Dieser Wert lässt sich mit der Erfolgsrate eines Verfahrens ohne Subsumption vergleichen. Der Complement-Naïve-Bayes-Klassifizierer erreicht eine Erfolgsrate von ca. 29%. Die Differenz beträgt damit 3%. Das spricht dafür, dass die Dokumente, die mit Complement Naïve Bayes richtig klassifiziert werden, auch mit einer Subsumption in Tiefe 5 korrekt zugeordnet werden. Der Großteil der korrekt klassifizierten Dokumente in Tiefe 5 bleibt erhalten und werden nicht in eine abstraktere Tiefe klassifiziert. Damit kann man sagen, dass die Subsumption in diesem Fall zu einen sehr geringen Informationsverlust durch Abstrahierung führt.

	Subsumption in Arts (5)		
Subsumptionstiefe	100 Merkmale	250 Merkmale	500 Merkmale
Gesamt	46,74%	49,01%	57,54%
2	71,08% (24%)	54,35% (32%)	76,19% (24%)
3	63,32% (4%)	65,60% (2%)	71,34% (2%)
4	72,07% (2%)	77,01% (2%)	73,47% (3%)
5	36,52% (70%)	44,84% (64%)	50,10% (71%)

Tab. 6.2.: *Split: 80% / 20% (Training/Test), Evaluierung mit 10 Durchläufen. Die Domäne ist Arts mit dem maximalen Verzweigungsgrad 5. „Gesamt“ gibt die Erfolgsrate der Klassifikationen nach der Subsumption über alle Dokumente an. Es folgen die Erfolgsraten für Tiefe n zusammen mit der Verteilung der Dokumente über die Tiefen der Hierarchie.*

Die Werte in **Tab. 6.3** zeigen, dass wenige Dokumente in Kategorien der Science Domäne mit geringer Tiefe zugeordnet werden. In Tiefe 3 fällt kein Dokument. In allen Versuchen werden die Klassifikationsergebnisse im Vergleich zu den Verfahren Complement Naïve Bayes und Naïve Bayes Multinomial verbessert. Ein genauerer Vergleich mit den Ergebnissen in Abschnitt B.4 zeigt, dass für 100, 250 und 500 Merkmale die Genauigkeit um etwa 4% verbessert wird. Auch in dieser Domäne zeigt sich, dass die Anzahl der in Tiefe 5 klassifizierten Dokumente geringfügig nachlässt (ca. 2%), wenn man Subsumption mit einfachem Complement Naïve Bayes und Naïve Bayes Multinomial vergleicht.

Die zwei Versuche haben verschiedene Eigenschaften des Subsumptionsverfahrens gezeigt:

- In den Domänen Arts und Science konnte die Erfolgsrate der Klassifizierer um 4 bis 7% gesteigert werden. ⇒ höhere Klassifikationsgenauigkeit
- Die Anzahl der in Tiefe 5 korrekt klassifizierten Dokumente sinkt nur geringfügig. ⇒ geringer Informationsverlust

An dieser Stelle soll ein weiterer Aspekt betrachtet werden. Wie genau ist das Klassifikationsergebnis, wenn ein Klassifikationsverfahren ausschließlich bis in Tiefe n klassifiziert und was ergibt ein Vergleich mit dem Subsumption-Klassifikationsmodell? Die hierarchische Genauigkeit zeigt die Erfolgsrate für Tiefe n für den Fall, dass alle Testdokumente in n zugeordnet werden. Die **Tab. 6.4** zeigt, dass die Fehlerrate mit der Tiefe in der Hierarchie zunimmt. Ein Vergleich dieser Werte mit der Gesamtgenauigkeit der Subsumption in Abschnitt 6.3 verdeutlicht, dass das Subsumptionsverfahren die Erfolgsrate eines Klassifikationsmodells mit

	Subsumption in Science (4)		
Subsumptionstiefe	100 Merkmale	250 Merkmale	500 Merkmale
Gesamt	74,80%	80,23%	83,77%
2	70,59% (10%)	61,02% (6%)	58,22% (5%)
3	0% (0%)	0% (0%)	0% (0%)
4	55,56% (1%)	25,00% (1%)	100,00% (1%)
5	75,41% (89%)	81,60% (93%)	85,04% (94%)

Tab. 6.3.: *Split: 80% / 20% (Training/Test), Evaluierung mit 10 Durchläufen. Die Domäne ist Science mit dem maximalen Verzweigungsgrad 4. „Gesamt“ gibt die Erfolgsrate der Klassifikationen nach der Subsumption über alle Dokumente an. Es folgen die Erfolgsraten für Tiefe n zusammen mit der Verteilung der Dokumente über die Tiefen der Hierarchie.*

geringerer Tiefe nicht erreichen kann.

	Hierarchische Genauigkeit Complement NB in Science (4)		
Tiefe in der Hierarchie	100 Merkmale	250 Merkmale	500 Merkmale
2	89,17%	90,67%	89,87%
3	81,07%	84,70%	84,13%
4	79,27%	83,33%	83,13%
5	67,53%	75,97%	77,37%

Tab. 6.4.: *Split: 80% / 20% (Training/Test), Evaluierung mit 10 Durchläufen. Für die Domäne Science und dem Klassifizierer Complement Naïve Bayes werden die Erfolgsraten für Tiefe n präsentiert, unter der Annahme, dass alle Dokumente in Tiefe n klassifiziert werden.*

Das Subsumption-Klassifikationsmodell eignet sich somit gut für Topic Identification, mit der man Label mit hohem Informationsgehalt generieren will. Die Gefahr falsch zu klassifizieren, wird möglichst gering gehalten, indem dynamisch abstraktere Label mit kleinerer Fehlerrate erzeugt werden können.

6.4. Abstrakte Kategorisierung

Unter abstrakter Kategorisierung versteht man die Klassifikation von Dokumenten in allgemeine Kategorien. Die Klassifikation geschieht nicht in einem spezialisierten Teilbaum oder einer bestimmten Domäne, sondern möglichst in den oberen Ebenen der Ontologie. Die Kategorien sind hier noch sehr abstrakt und beinhalten eine große Menge von Dokumenten.

Angewandt zur Topic Identification liefert eine abstrakte Kategorisierung somit eher allgemeine Label für ein Dokument. Man könnte im ersten Moment davon ausgehen, dass diese Label auf Grund ihrer Abstraktheit nicht sehr aussagekräftig sind. Das Gegenteil ist jedoch der Fall. Um dieses zu verdeutlichen nehmen wir an, ein Dokument, das von „Madonna“ handelt, soll in die obersten zwei Ebenen der DMoz-Ontologie klassifiziert werden. Folglich wäre das Ergebnis dieser abstrakten Klassifikation eine sehr allgemeine Kategorie. Die Information, ob sich das Dokument in der allgemeinen Kategorie Top/Society/Religion_and_Spirituality oder in der allgemeinen Kategorie Top/Arts/Music befindet, ist jedoch von großem Wert. Beispielsweise für einen Benutzer, der in einer Suchmaschine nach dem Begriff „Madonna“ sucht.

In Abschnitt 6.4.1 wird der Aufbau der Experimente erläutert. Es wird beschrieben, welche Testhierarchien verwendet werden und was mit den Experimenten gezeigt werden soll. Die Ergebnisse der Experimente werden in Abschnitt 6.4.2 präsentiert.

6.4.1. Experimente

Die Experimente beschränken sich auf die oberen beiden Ebenen der DMoz-Ontologie. Die Ontologie besitzt 17 Kategorien in Tiefe 1 und 645 Kategorien in Tiefe 2, vgl. Abschnitt 2.4.2. In den Experimenten werden die Kategorien Top/Adult und Top/World mit allen ihren Unterkategorien ignoriert. Bei der Kategorie Top/Adult handelt es sich nur um eine Ergänzung der anderen Kategorien mit nicht jugendfreien Dokumenten. Die Kategorie Top/World wird ignoriert, weil sie nicht ausreichend englischsprachige Dokumente enthält. Die Ontologie wird auf Tiefe 2 geprunt. Die Kategorien in Tiefe 2 stellen nun die Blätter der geprunten Ontologie dar. Die Experimente werden an vier Testhierarchien durchgeführt. Alle Testhierarchien sind Teilbäume aus der geprunten Ontologie und unterscheiden sich lediglich in der Breite. Die Testhierarchien werden so gewählt, dass ein gleichmäßiges Training garantiert werden kann. Für jede Testhierarchie gilt, dass alle Kategorien in Tiefe 1 die gleiche Anzahl von Unterkategorien besitzen und jede Blattkategorie die gleiche Anzahl von Dokumenten enthält. Die Anzahl der Nachfolgekategorien aller Kategorien in Tiefe 1 beträgt immer fünf. Pro Blattkategorie werden jeweils 150 Dokumente heruntergeladen.

Die Größe und die Beschaffenheit der vier Testhierarchien ist in **Tab. 6.5** dargestellt. Die Testhierarchie Hier1 besteht aus zwei Kategorien in Tiefe 1 sowie zehn Kategorien in Tiefe 2 und enthält insgesamt 1 500 Dokumente. In **Abb. 6.7** ist die Testhierarchie Hier1 dargestellt, linke Seite. Ebenfalls in Abb. 6.7 ist die Testhierarchie Hier2 zu sehen, rechte Seite. Testhierarchie Hier2 besteht aus drei Kategorien in Tiefe 1 sowie 15 Kategorien in Tiefe 2 und enthält insgesamt 2 250 Dokumente. Die dritte Testhierarchie, Hier3 enthält 7 500 Dokumente und besteht aus zehn Kategorien in Tiefe 1 sowie 50 Kategorien in Tiefe 2. Die vierte Testhierarchie, Hier4 beinhaltet 9 750 Dokumente und besteht aus 15 Kategorien in Tiefe 1 sowie 75 Kategorien in Tiefe 2. Diese Testhierarchie umfasst alle Kategorien der DMoz-Ontologie in Tiefe 1, ohne Top/Adult und Top/World. Die Kategorien, aus denen Hier3 und Hier4 bestehen sind in Anhang C aufgeführt.

Name	# Kategorien in Tiefe 1	# Kategorien in Tiefe 2	# Dokumente	# Wörter im Vokabular
Hier1	2	10	1 500	35 485
Hier2	3	15	2 250	66 252
Hier3	10	50	7 500	123 206
Hier4	15	75	9 750	169 562

Tab. 6.5.: *Daten zu den Testhierarchien. Dargestellt ist der Name der Hierarchie, die Anzahl der Kategorien in Tiefe 1 und die Anzahl der Kategorien in Tiefe 2, außerdem die gesamte Anzahl der Dokumente und die Größe des Vokabulares nach der Indexierung.*

Die Experimente sollen zeigen, wie gut sich die Klassifizierer zur abstrakten Kategorisierung in die DMoz-Ontologie eignen. Weiterhin wird untersucht, wie stark die Klassifikationsgenauigkeit von der Größe der Testhierarchie abhängt. Es wird erwartet, dass die Genauigkeit mit steigender Anzahl von Kategorien abnimmt:
$Accuracy(Hier1) > Accuracy(Hier2) > Accuracy(Hier3) > Accuracy(Hier4)$.
Die Experimente sollen außerdem zeigen, ob die hierarchischen Klassifikationsverfahren einem flachen Klassifikationsverfahren überlegen sind. Des Weiteren wird untersucht, bei welcher Anzahl von Merkmalen die Klassifizierer die besten Ergebnisse liefern.

6.4.2. Ergebnisse

Im Folgenden werden die Ergebnisse der Experimente präsentiert. Eine detaillierte Auflistung aller Ergebnisse befindet sich in Anhang B.1.

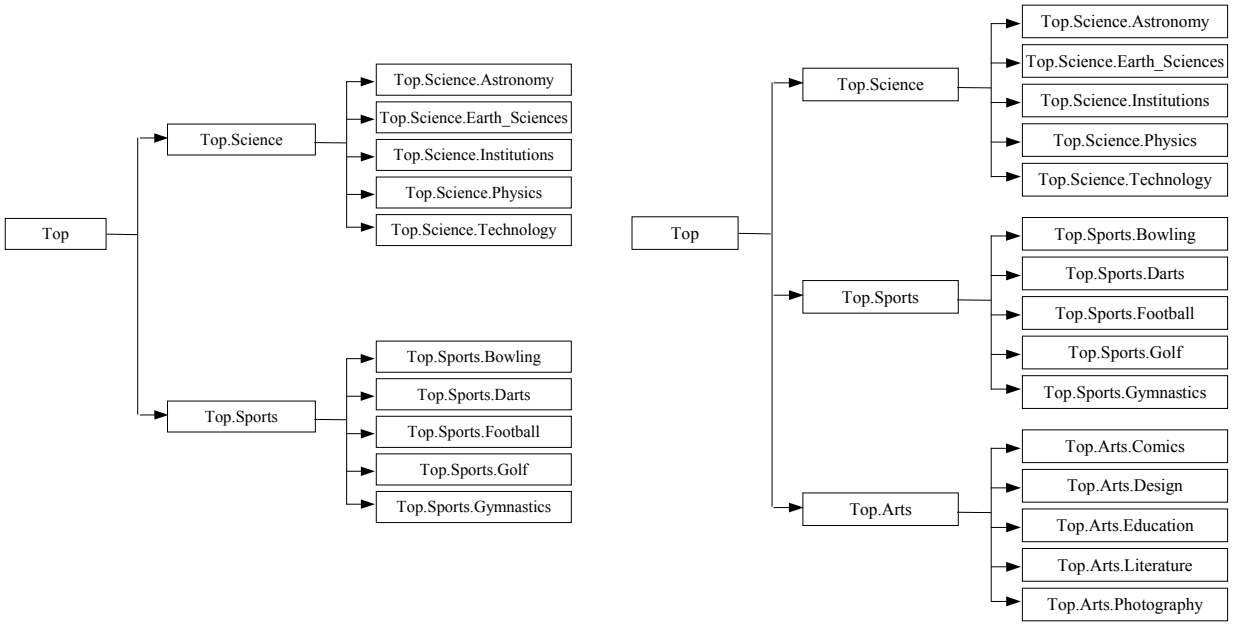


Abb. 6.7.: *Links: Testhierarchie Hier1. Rechts: Testhierarchie Hier2. Dargestellt sind die Hierarchien, mit allen darin enthaltenen Kategorien.*

Naïve-Bayes-Klassifizierer

Die Ergebnisse der Naïve-Bayes-Klassifizierer sind in den Diagrammen in **Abb. 6.8** dargestellt. Der Multi-Pfad-Naïve-Bayes-Klassifizierer stellt einen Spezialfall dar. Daher werden die Ergebnisse des Multi-Pfad-Naïve-Bayes-Klassifizierers im Folgenden gesondert präsentiert.

Wie erwartet nimmt die Klassifikationsgenauigkeit mit steigender Anzahl von Kategorien ab. Während die besten Klassifizierer, z. B. Naïve Bayes Flach, in Hier1 Erfolgsraten um die 80% erzielen, liegt die Erfolgsrate in Hier4 bei ca. 40%.

Im Vergleich der hierarchischen Klassifizierer mit dem flachen Klassifizierer liefert der flache Klassifizierer bessere Klassifikationsergebnisse. In Hier1 und Hier2 erreichen Naïve Bayes Flach und Naïve Bayes in etwa die gleichen Erfolgsraten, ca. 80% in Hier1 und ca. 65% in Hier2. Bei einer höheren Anzahl von Kategorien, etwa in Hier3 oder in Hier4, schneidet der flache Klassifzierer jedoch besser ab. Der Unterschied beträgt etwa 5% bis 10%.

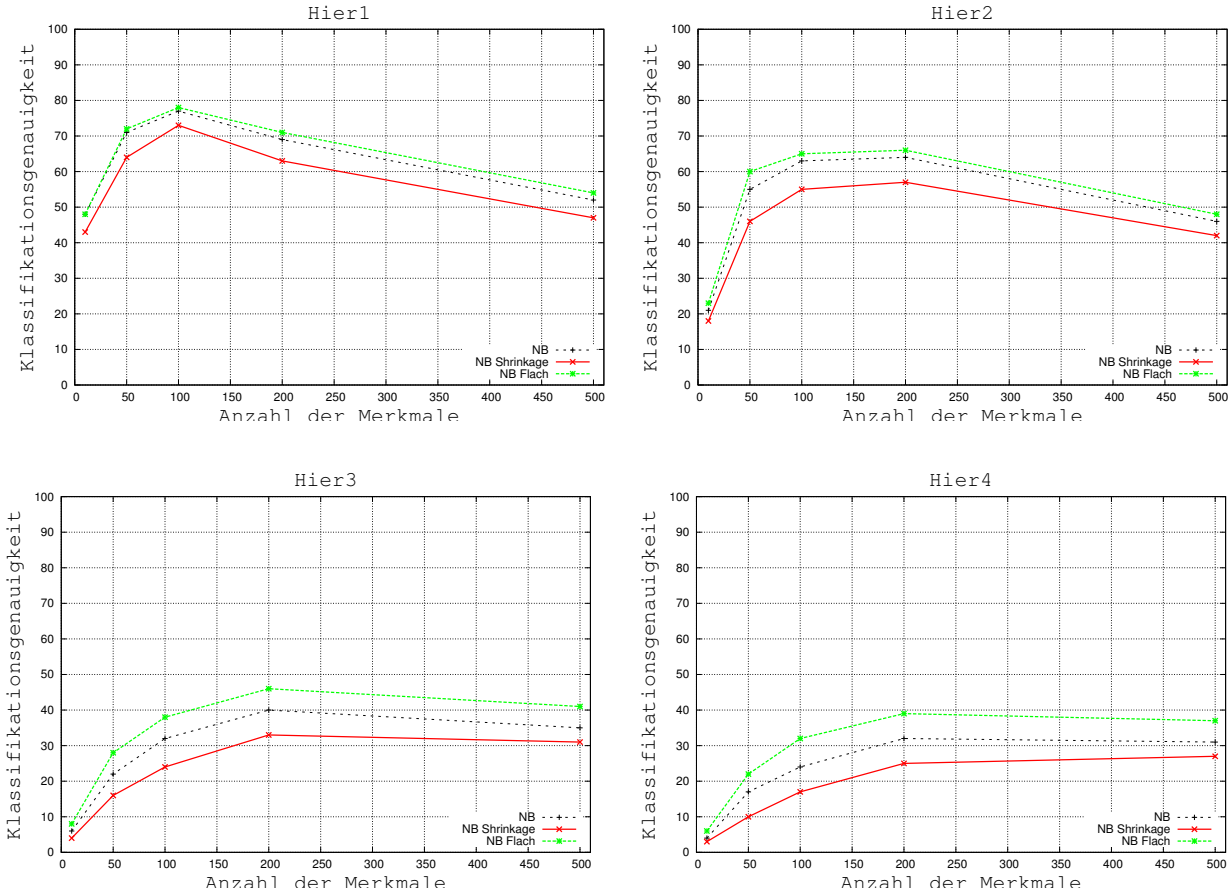


Abb. 6.8.: *Ergebnisse der Naïve-Bayes-Klassifizierer. Die Diagramme zeigen die Erfolgsraten (Klassifikationsgenauigkeit in Prozent) der Klassifizierer Naïve Bayes, Naïve Bayes Shrinkage, und Naïve Bayes Flach in den Testhierarchien Hier1, Hier2, Hier3 und Hier4.*

Das Training eines hierarchischen Naïve-Bayes-Klassifizierers mit Shrinkage bringt keine Verbesserung der Klassifikationsgenauigkeit mit sich. Die Erfolgsrate liegt sogar etwa 5% bis 10% unter der des nicht mit Shrinkage trainierten hierarchischen Naïve-Bayes-Klassifizierers.

Die Anzahl der Merkmale hat eine große Auswirkung auf die Erfolgsrate. Die besten Ergebnisse konnten mit 100 bis 200 Merkmalen erzielt werden. Außerdem konnte beobachtet werden, dass die Klassifikationsgenauigkeit mit steigender Anzahl der Merkmale abnimmt.

Multi-Pfad-Klassifikation

Die Ergebnisse der Experimente mit dem Klassifizierer Naïve Bayes Multi-Pfad sind in **Tab. 6.6** dargestellt. Die Zahl in Klammern ist die durchschnittliche Anzahl der Kategorien, in die ein Dokument pro Klassifikation klassifiziert wird. Betrachtet man beispielsweise

die Klassifikation in Hier1 mit zehn Merkmalen, so wird ein Dokument im Durchschnitt in vier Kategorien klassifiziert. In 91,2% der Fälle ist die richtige Kategorie unter diesen vier Kategorien.

# Merkmale	Hier1	Hier2	Hier3	Hier4
10	91,2% (4)	84,7% (6,4)	36,1% (4,1)	7,0% (2,9)
50	85,8% (3)	79,2% (3,6)	52,1% (3,2)	33,5% (2,6)
100	82,7% (2,5)	75,5% (3)	50,8% (2,8)	38,7% (2,5)
200	71,9% (2,5)	67,9% (2,8)	50,8% (2,5)	40,8% (2,4)
500	47,0% (2,3)	45,1% (2,5)	39,5% (2,4)	34,4% (2,3)

Tab. 6.6.: *Erfolgsraten (Klassifikationsgenauigkeit in Prozent) des Klassifizierers Naïve Bayes Multi-Pfad für Hier1, Hier2, Hier3 und Hier4. Die Zahl in Klammern ist die durchschnittliche Anzahl der Kategorien, in die das Dokument pro Klassifikation klassifiziert wird. Eine Klassifikation ist erfolgreich, wenn die richtige Kategorie dabei ist. Grenzwert: 0,02.*

Die Erfolgsrate bei der Klassifikation in Testhierarchie Hier1 und in Testhierarchie Hier2 nimmt mit steigender Anzahl von Merkmalen ab. Die besten Erfolgsraten werden mit zehn Merkmalen erreicht. Für Testhierarchie Hier3 werden die besten Erfolgsraten mit 50 Merkmalen erreicht und für Testhierarchie Hier4 mit 200 Merkmalen.

Bei der Multi-Pfad-Klassifikation ist nicht nur die Erfolgsrate entscheidend, sondern auch die Anzahl der Kategorien in die ein Dokument klassifiziert wird. Die Anzahl verringert sich mit steigender Anzahl der Merkmale. Dies gilt für alle vier Testhierarchien.

Die Erfolgsrate des Multi-Pfad-Klassifizierers liegt ca. 10% über der des Single-Pfad-Naïve-Bayes-Klassifizierers. Die Ergebnisse des Multi-Pfad-Klassifizierers sind jedoch nicht direkt mit den Ergebnissen der Single-Pfad-Klassifizierer vergleichbar, da ein Multi-Pfad-Klassifizierer ein Dokument nicht eindeutig in eine Kategorie klassifiziert.

Weka-Klassifizierer

Die Ergebnisse der Weka-Klassifizierer sind in **Abb. 6.9** dargestellt. Die Klassifizierer Naïve Bayes Multinomial, Complement Naïve Bayes und AdaBoostM1 liefern für alle Testhierarchien etwa die gleichen Ergebnisse. Die Erfolgsrate des Naïve-Bayes-Klassifizierers hingegen liegt in allen Experimenten ca. 25% darunter.

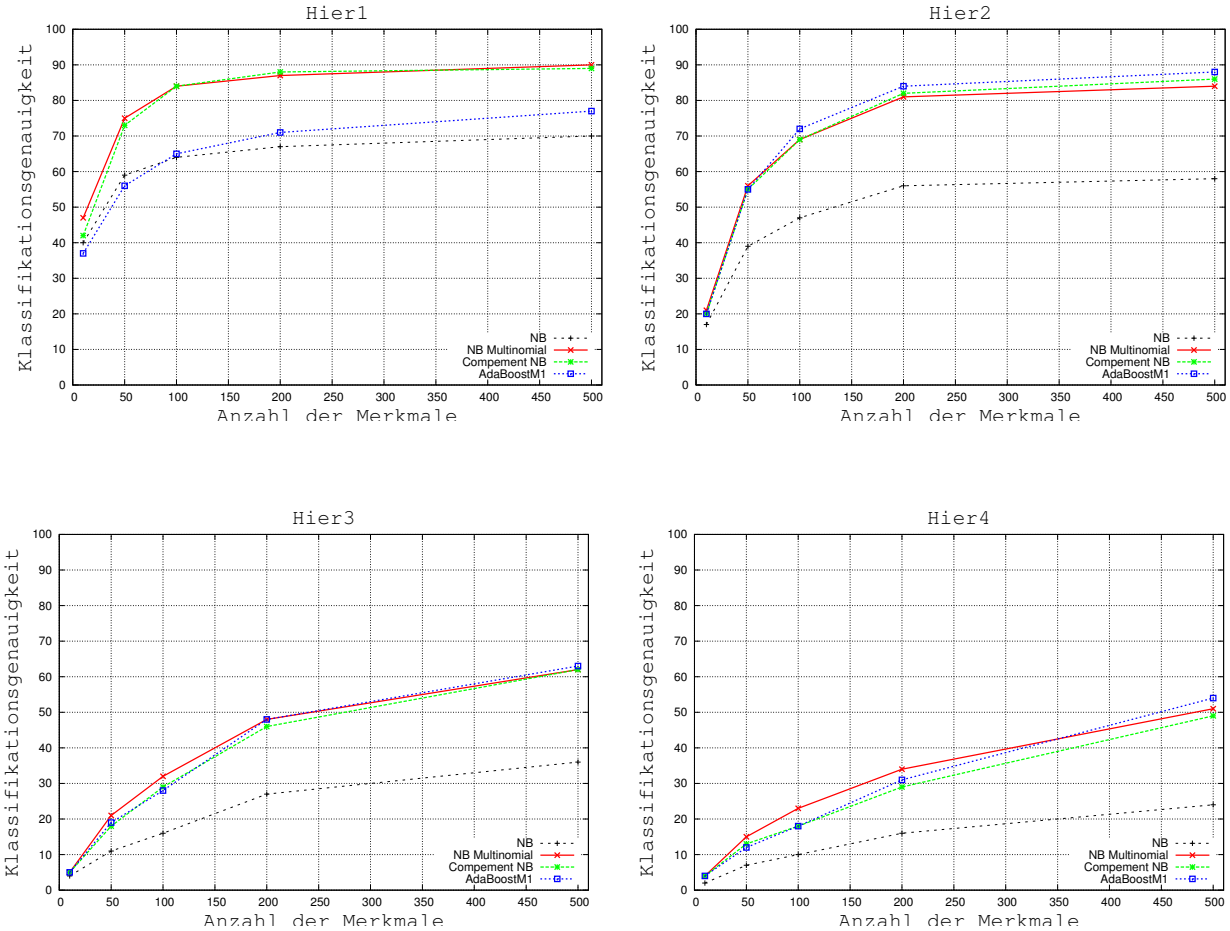


Abb. 6.9.: *Ergebnisse der Weka-Klassifizierer. Die Diagramme zeigen die Erfolgsraten (Klassifikationsgenauigkeit in Prozent) der Klassifizierer Naïve Bayes, Naïve Bayes Multinomial, Complement Naïve Bayes und AdaBoostM1 mit Complement Naïve Bayes in den Testhierarchien Hier1, Hier2, Hier3 und Hier4.*

Auch hier nimmt die Klassifikationsgenauigkeit mit steigender Anzahl von Kategorien ab. Die besten Klassifizierer wie z. B. Complement Naïve Bayes erreichen in Hier1 und Hier2 eine Erfolgsrate um die 90%, während die Erfolgsrate in Hier3 ca. 60% beträgt und in Hier4 ca. 50%.

Die Erfolgsrate aller Weka-Klassifizierer nimmt mit steigender Anzahl von Merkmalen zu. Allerdings verschlechtert sich mit steigender Anzahl von Merkmalen die Performanz bei der Generierung des Klassifikationsmodelles. Eine Verschlechterung der Performanze ist auch bei einer steigenden Anzahl von Kategorien zu beobachten.

Im Vergleich der hierarchischen Weka-Klassifizierer mit dem selbst implementierten flachen Naïve-Bayes-Klassifizierer, schneiden alle Weka-Klassifizierer bis auf Weka-Naïve-Bayes besser

ab, als der flache Naïve-Bayes-Klassifizierer.

7. Zusammenfassung und Ausblick

Als Topic Identification bezeichnet man das maschinelle Erstellen von Überschriften für Dokumente. Viele Topic-Identification-Verfahren sind nicht in der Lage, externes Wissen zu verwenden. Externes Wissen wird jedoch benötigt, um abstrakte Label zu generieren. Durch die Verwendung von externem Wissen lässt sich sicherstellen, dass die erzeugten Label aussagekräftig sind. Wir verwenden in dieser Arbeit als Labels die Deskriptoren der Kategorien der DMoz-Ontologie. Die Dokumente werden in die Kategorien der DMoz-Ontologie eingeordnet, dies geschieht durch Textklassifikation.

Die Klassifikation erfolgt in eine hohe Anzahl von Kategorien, um einen möglichst großen Wissensbereich abzudecken. Bei einer hohen Anzahl von Kategorien liefert die hierarchische Textklassifikation im Gegensatz zur flachen Textklassifikation bessere Ergebnisse. Die DMoz-Ontologie besitzt eine vordefinierte hierarchische Struktur und ermöglicht somit eine hierarchische Klassifikation. In dieser Arbeit wird hierarchische Textklassifikation als Verfahren zur Topic Identification genutzt.

Zur hierarchischen Textklassifikation werden Verfahren eingesetzt, von denen bereits in früheren Arbeiten gezeigt wird, dass sie gute Klassifikationsergebnisse liefern. Es werden verschiedene Naïve-Bayes-Verfahren benutzt. Des Weiteren werden Naïve-Bayes-Klassifizierer und ein Boosting-Verfahren von Weka eingesetzt. Um die Klassifikationsergebnisse zu verbessern, wird zusätzlich Multi-Pfad-Klassifikation und ein Subsumptionsverfahren angewandt. Bei einer Multi-Pfad-Klassifikation wird ein Dokument in die vielversprechensten Kategorien klassifiziert. Das Subsumptionsverfahren subsumiert die Ergebnisse mehrerer Klassifizierer. Beide Verfahren erhöhen die Wahrscheinlichkeit einer korrekten Klassifikation.

Im Rahmen dieser Arbeit ist eine Software entwickelt worden. Hierbei handelt es sich um ein Framework, das die Klassifikationsverfahren sowie verschiedene Evaluierungsfunktionen bereitstellt. Das Framework ist modular aufgebaut und lässt sich daher leicht erweitern. Über eine Schnittstelle lassen sich Weka-Klassifizierer für ein hierarchisches Klassifikationsmodell nutzen.

Folgende Anwendungsbereiche von Topic Identification sind in den Experimenten untersucht worden:

Domänenspezifische Kategorisierung Hierbei handelt es sich um die Klassifikation eines Dokumentes in einen speziellen Wissensbereich. Das Ziel ist, ein Label generieren zu können, das den Inhalt des Dokumentes möglichst genau beschreibt.

Abstrakte Kategorisierung Dies bezeichnet die Klassifikation eines Dokumentes in eine große Menge von allgemeinen Kategorien, um abstrakte Label generieren zu können.

Die Klassifikationsgenauigkeit der domänenspezifischen Kategorisierung ist stark von dem jeweiligen Wissensgebiet abhängig. In gut strukturierten Wissensgebieten wie beispielsweise Science, können gute Klassifikationsergebnisse erzielt werden. Im Falle der abstrakten Kategorisierung konnten mittels hierarchischer Textklassifikation gute Klassifikationsergebnisse erzielt werden. Die Klassifkationsgenauigkeit ist jedoch stark von der Anzahl der Kategorien abhängig.

Weitere Arbeit sollte in die Verbesserung der Klassifizierer investiert werden, um eine gute Klassifikationsgenauigkeit, bei einer noch höheren Anzahl von Kategorien zu erreichen. Ein vielversprechender Ansatz könnte auch eine Kombination der Verfahren zur domänenstezifischen Kategorisierung und der Verfahren zur abstrakten Kategorisierung sein. Hierbei würde ein Dokument zunächst mittels abstrakter Kategorisierung in eine bestimmte Domäne klassifiziert und dann mittels domänenspezifischer Kategorisierung in eine spezielle Kategorie.

A. Daten zur DMoz-Ontologie

Tiefe	# Knoten	# Dokumente	⌀ # Dokumente pro Knoten	⌀ Knotengrad	Varianz
0	1	0	0	17.0	0
1	17	45	2.0	37.9	3.7
2	645	6970	10.0	11.7	1.9
3	7565	145939	19.0	5.1	1.0
4	38581	519030	13.0	2.2	0.4
5	86770	820408	9.0	1.2	0.2
6	105678	960352	9.0	1.5	0.2
7	158960	777838	4.0	1.0	0.2
8	153499	619421	4.0	0.6	0.1
9	96386	481463	4.0	0.5	0.1
10	47548	275915	5.0	0.2	0.1
11	10963	114804	10.0	0.2	0
12	2650	25052	9.0	0.2	0
13	404	3381	8.0	0	0
14	12	117	9.0	0	0

Tab. A.1.: *Größenordnung und Beschaffenheit der DMoz-Ontologie. Dargestellt sind für jede Tiefe die Anzahl der Knoten, die Anzahl der Dokumente, die durchschnittliche Anzahl der Dokumente pro Knoten und der durchschnittliche Knotengrad mit Varianz.*

Tiefe	# Knoten	# Kategorien Σ	# Dokumente Σ	# Dokumente Σ pro Knoten	# Kategorien Σ pro Knoten
0	1	709679	4766935	4766935.0	709679.0
1	17	709678	4766935	280407.9	41745.8
2	645	709661	4766890	7390.5	1100.2
3	7565	709016	4759920	629.2	93.7
4	38581	701451	4613981	119.6	18.2
5	86770	662870	4094951	47.2	7.6
6	105678	576100	3274543	31.0	5.5
7	158960	470422	2314191	14.6	3.0
8	153499	311462	1536353	10.0	2.0
9	96386	157963	900732	9.3	1.6
10	47548	61577	419269	8.8	1.3
11	10963	14029	143354	13.1	1.3
12	2650	3066	28550	10.8	1.2
13	404	416	3498	8.7	1.0
14	12	12	117	9.8	1.0

Tab. A.2.: *Größenordnung und Beschaffenheit der DMoz-Ontologie mit kumulierten Werten. Dargestellt sind für jede Tiefe die Anzahl der Knoten, die Anzahl der kumulierten Kategorien, die Anzahl der kumulierten Dokumente, die durchschnittliche Anzahl der kumulierten Dokumente pro Knoten und die durchschnittliche Anzahl der kumulierten Kategorien pro Knoten.*

B. Klassifikationsergebnisse

B.1. Abstrakte Kategorisierung

		Naïve-Bayes-Klassifikationsmodell			
Hierarchie	# Merkmale	Naïve Bayes	NB Shrink.	NB Multi-Pfad	NB Flach
Hier1	10	48,9%	43,2%	91,2% (4)	48,9%
	50	71,4%	64,5%	85,8% (3)	72,2%
	100	77,4%	73,1%	82,7% (2,5)	78,1%
	200	69,9%	63,7%	71,9% (2,5)	71,6%
	500	52,1%	47,5%	47,0% (2,3)	54,2%
Hier2	10	21,0%	18,2%	84,7% (6,4)	23,9%
	50	55,1%	46,4%	79,2% (3,6)	60,3%
	100	63,9%	55,6%	75,5% (3)	65,7%
	200	64,1%	57,9%	67,9% (2,8)	66,0%
	500	46,2%	42,0%	45,1% (2,5)	48,9%
Hier3	10	6,3%	4,6%	36,1% (4,1)	8,6%
	50	22,6%	16,6%	52,1% (3,2)	28,4%
	100	32,0%	24,4%	50,8% (2,8)	38,1%
	200	40,3%	33,8%	50,8% (2,5)	46,4%
	500	35,7%	31,8%	39,5% (2,4)	41,4%
Hier4	10	4,6%	3,4%	7,0% (2,9)	6,7%
	50	17,3%	10,7%	33,5% (2,6)	22,6%
	100	24,6%	17,8%	38,7% (2,5)	32,0%
	200	32,0%	25,5%	40,8% (2,4)	39,5%
	500	31,3%	27,7%	34,4% (2,3)	37,4%

Tab. B.1.: *Ergebnisse. Split: 80% / 20% (Training/Test), Evaluierung mit 10 runs. Die Klassifizierer sind eigene Implementierungen: Naïve Bayes, Naïve Bayes mit Shrinkage, Naïve Bayes Multi-Pfad und Naïve Bayes Flach. Die Zahl in Klammern bei Naïve Bayes Multi-Pfad ist die durchschnittliche Anzahl der Kategorien, in die das Dokument pro Klassifikation klassifiziert wurde. Grenzwert für Multi-Pfad: 0,02.*

		Weka-Klassifikationsmodell			
Hierarchie	# Merkmale	Naïve Bayes	NB multi.	compl. NB	AdaBoostM1
Hier1	10	40,9%	47,4%	42,9%	43,4%
	50	59,8%	75,4%	73,0%	75,2%
	100	64,0%	84,9%	84,1%	83,8%
	200	67,4%	87,2%	88,0%	90,8%
	500	70,8%	90,1%	89,6%	91,9%
Hier2	10	17,4%	21,5%	20,1%	20,3%
	50	39,9%	56,2%	55,7%	55,5%
	100	47,2%	69,1%	69,8%	72,7%
	200	56,4%	81,4%	82,0%	84,8%
	500	58,6%	84,4%	86,5%	88,6%
Hier3	10	4,7%	5,7%	5,8%	5,6%
	50	11,2%	21,7%	18,5%	19,4%
	100	16,4%	32,9%	29,1%	28,7%
	200	27,9%	48,4%	46,4%	48,1%
	500	36,8%	62,0%	62,1%	63,9%
Hier4	10	2,7%	4,2%	4,5%	4,2%
	50	7,4%	15,5%	13,2%	12,7%
	100	10,1%	23,4%	18,8%	18,8%
	200	16,7%	34,7%	29,1%	31,5%
	500	24,1%	51,1%	49,1%	54,1%

Tab. B.2.: *Ergebnisse. Split: 80% / 20% (Training/Test), Evaluierung mit 10 runs. Die Klassifizierer sind aus dem Weka-Tool: Naïve Bayes, Naïve Bayes Multinomial, Complement Naïve Bayes und AdaBoostM1 mit Complement-Naïve-Bayes-Klassifizierern.*

B.2. Domänenspezifische Kategorisierung

		Naïve-Bayes-Klassifikationsmodell		
Hierarchie	# Merkmale	Naïve Bayes	NB Shrink.	NB Flach
Arts (4)	10	14,22% (1,14)	13,12% (1,04)	16,19% (1,85)
	100	32,98% (1,32)	30,78% (1,84)	39,47% (1,17)
	250	30,79% (0,89)	27,67% (1,08)	36,12% (1,72)
	500	25,02% (0,92)	20,36% (1,70)	29,16% (1,72)
Arts (5)	10	8,08% (0,70)	7,90% (1,19)	11,67% (0,76)
	100	28,75% (1,80)	25,29% (1,43)	33,62% (1,55)
	250	24,65% (1,78)	22,11% (1,85)	29,49% (1,37)
	500	19,40% (0,91)	17,00% (1,23)	22,25% (1,46)
Science (4)	10	28,57% (1,80)	28,07% (2,37)	32,23% (1,11)
	100	56,63% (2,72)	53,53% (2,52)	58,37% (2,40)
	250	45,47% (2,32)	41,87% (2,12)	46,73% (2,34)
	500	34,00% (1,65)	32,53% (1,39)	37,33% (1,07)
Science (5)	10	27,93% (1,39)	25,53% (1,78)	32,10% (2,11)
	100	59,10% (1,63)	51,90% (2,43)	58,33% (2,99)
	250	45,00% (1,40)	42,00% (1,18)	47,40% (1,31)
	500	34,97% (1,62)	33,97% (2,90)	37,63% (2,76)
Society (4)	10	14,96% (1,64)	12,88% (0,84)	18,56% (0,85)
	100	34,98% (1,17)	30,82% (1,71)	44,04% (1,91)
	250	35,98% (1,49)	32,00% (1,95)	41,54% (2,07)
	500	29,68% (1,91)	26,74% (1,21)	34,78% (0,74)
Society (5)	10	13,95% (1,07)	13,71% (1,47)	17,41% (1,42)
	100	25,62% (1,54)	23,27% (1,38)	35,48% (1,46)
	250	28,68% (1,23)	27,06% (1,28)	35,38% (1,53)
	500	22,71% (1,59)	21,59% (1,29)	28,41% (1,40)

Tab. B.3.: *Ergebnisse. Split: 80% / 20% (Training/Test), Evaluierung mit 10 Durchläufen. In Klammern ist die Standardabweichung angegeben. Die Klassifizierer sind eigene Implementierungen: Naïve Bayes, Naïve Bayes mit Shrinkage und Naïve Bayes Flach.*

		Weka-Klassifikationsmodell			
Hierarchie	# Merkmale	Naïve Bayes	NB multi.	compl. NB	AdaBoostM1
Arts (4)	10	8,48% (1,25)	11,55% (1,38)	12,34% (1,09)	12,09% (1,19)
	100	18,10% (1,97)	33,79% (2,01)	33,48% (2,04)	37,16% (1,52)
	250	24,40% (4,01)	46,86% (2,29)	46,57% (1,35)	48,55% (2,25)
	500	27,71% (2,88)	49,83% (1,65)	50,59% (1,97)	51,81% (3,68)
	1 000	28,43% (1,88)	50,81% (2,01)	54,43% (2,09)	52,55% (3,04)
	2 000	30,34% (2,59)	55,03% (2,60)	55,07% (2,08)	55,97% (4,05)
Arts (5)	10	5,07% (1,14)	7,58% (0,86)	7,75% (0,86)	7,01% (0,74)
	100	12,38% (1,10)	27,68% (1,24)	28,21% (1,55)	29,68% (1,71)
	250	15,51% (1,33)	32,71% (2,04)	33,46% (1,71)	36,79% (1,78)
	500	17,40% (1,33)	37,53% (2,95)	40,54% (1,44)	41,11% (1,79)
	1 000	19,65% (1,36)	40,67% (2,06)	41,72% (4,39)	44,78% (2,27)
	2 000	22,85% (1,54)	44,81% (3,43)	44,86% (2,65)	48,61% (2,51)
Science (4)	10	17,80% (2,42)	27,70% (1,55)	28,23% (2,72)	29,97% (1,60)
	100	38,70% (3,68)	66,07% (3,26)	68,93% (1,81)	67,13% (2,89)
	250	45,63% (2,50)	74,83% (3,15)	74,40% (2,57)	74,43% (1,89)
	500	48,17% (2,34)	76,67% (2,76)	77,33% (2,59)	77,97% (3,67)
	1 000	57,60% (3,29)	80,20% (2,19)	82,07% (2,00)	81,07% (2,46)
	2 000	59,23% (4,78)	83,27% (2,01)	85,40% (2,86)	83,50% (2,68)
Science (5)	10	17,73% (2,24)	26,47% (1,50)	28,13% (1,59)	26,93% (1,91)
	100	38,70% (2,85)	69,23% (2,07)	68,87% (2,82)	69,93% (2,86)
	250	44,67% (3,66)	76,37% (2,96)	74,93% (3,10)	74,60% (1,95)
	500	48,00% (2,47)	78,80% (2,54)	78,93% (2,25)	75,47% (2,16)
	1 000	55,13% (3,44)	83,10% (2,20)	83,90% (2,15)	79,23% (3,09)
	2 000	59,40% (2,22)	83,20% (1,67)	84,83% (2,29)	83,10% (3,88)
Society (4)	10	10,02% (1,06)	12,26% (1,12)	11,64% (1,26)	11,84% (2,04)
	100	16,30% (2,60)	33,70% (1,79)	34,84% (1,83)	38,94% (3,64)
	250	23,24% (2,46)	45,48% (3,91)	46,92% (3,99)	52,92% (4,14)
	500	29,34% (2,29)	49,74% (3,78)	53,56% (3,07)	54,76% (3,73)
	1 000	30,28% (4,34)	50,58% (4,82)	53,54% (3,91)	55,08% (6,37)
	2 000	32,76% (1,76)	48,26% (4,31)	56,66% (3,92)	55,32% (3,40)
Society (5)	10	7,59% (1,04)	11,67% (1,34)	9,70% (1,46)	10,20% (1,14)
	100	15,00% (1,34)	30,38% (2,66)	29,05% (2,03)	31,39% (2,12)
	250	20,02% (1,60)	41,41% (4,33)	42,58% (2,87)	47,65% (2,88)
	500	22,26% (2,34)	46,74% (3,55)	48,88% (2,26)	49,05% (3,40)
	1 000	26,71% (1,05)	47,80% (3,86)	52,02% (3,55)	54,64% (3,37)
	2 000	28,68% (2,63)	47,44% (6,61)	54,29% (3,43)	53,79% (6,34)

Tab. B.4.: *Ergebnisse. Split: 80% / 20% (Training/Test), Evaluierung mit 10 Durchläufen. In Klammern ist die Standardabweichung angegeben. Die Klassifizierer sind aus dem Weka-Tool: Naïve Bayes, Naïve Bayes Multinomial, Complement Naïve Bayes und AdaBoostM1 mit Complement Naïve Bayes*

C. Testhierarchien

Name	Tiefe	Kategorien
Hier 3	0	Top
	1	Top/Arts, Top/Business, Top/Computers, Top/Games, Top/-Health, Top/Home, Top/Kids_and_Teens, Top/News, Top/-Science, Top/Sports
	2	Top/Arts/Comics, Top/Arts/Design, Top/Arts/-Education, Top/Arts/Literature, Top/Arts/Photography, Top/Business/Chemicals, Top/Business/Employment, Top/Business/Financial_Services, Top/Business/-International_Business_and_Trade, Top/Business/Opportunities, Top/Computers/Artificial_Intelligence, Top/Computers/-Emulators, Top/Computers/Graphics, Top/Computers/-Hardware, Top/Computers/Software, Top/Games/Board_Games, Top/Games/Card_Games, Top/Games/Online, Top/Games/-Puzzles, Top/Games/Roleplaying, Top/Health/Beauty, Top/Health/Dentistry, Top/Health/Fitness, Top/Health/-Medicine, Top/Health/Pharmacy, Top/Home/Cooking, Top/Home/Family, Top/Home/Gardening, Top/Home/-Home_Improvement, Top/Home/Personal_Finance, Top/-Kids_and_Teens/People_and_Society, Top/Kids_and_Teens/-Pre-School, Top/Kids_and_Teens/School_Time, Top/-Kids_and_Teens/Sports_and_Hobbies, Top/Kids_and_Teens/-Teen_Life, Top/News/Analysis_and_Opinion, Top/News/-Colleges_and_Universities, Top/News/Current_Events, Top/-News/Media, Top/News/Online_Archives, Top/Science/-Astronomy, Top/Science/Earth_Sciences, Top/Science/-Institutions, Top/Science/Physics, Top/Science/Technology, Top/Sports/Bowling, Top/Sports/Darts, Top/Sports/Football, Top/Sports/Golf, Top/Sports/Gymnastics

Tab. C.1.: *Testhierarchie Hier3. Dargestellt sind alle Kategorien der Hierarchie in der entsprechenden Tiefe.*

Name	Tiefe	Kategorien
Hier 4	0	Top
	1	Top/Arts, Top/Business, Top/Computers, Top/Games, Top/-Health, Top/Home, Top/Kids_and_Teens, Top/News, Top/-Recreation, Top/Reference, Top/Regional, Top/Science, Top/-Shopping, Top/Society, Top/Sports
	2	Top/Arts/Comics Top/Arts/Design, Top/Arts/-Education, Top/Arts/Literature, Top/Arts/Photography, Top/Business/Chemicals, Top/Business/Employment, Top/Business/Financial_Services, Top/Business/-International_Business_and_Trade, Top/Business/Opportunities, Top/Computers/Artificial_Intelligence, Top/Computers/-Emulators, Top/Computers/Graphics, Top/Computers/-Hardware, Top/Computers/Software, Top/Games/Board_Games, Top/Games/Card_Games, Top/Games/Online, Top/Games/-Puzzles, Top/Games/Roleplaying, Top/Health/Beauty, Top/Health/Dentistry, Top/Health/Fitness, Top/Health/-Medicine, Top/Health/Pharmacy, Top/Home/Cooking, Top/Home/Family, Top/Home/Gardening, Top/Home/-Home_Improvement, Top/Home/Personal_Finance, Top/-Kids_and_Teens/People_and_Society, Top/Kids_and_Teens/-Pre-School, Top/Kids_and_Teens/School_Time, Top/-Kids_and_Teens/Sports_and_Hobbies, Top/Kids_and_Teens/-Teen_Life, Top/News/Analysis_and_Opinion, Top/News/-Colleges_and_Universities, Top/News/Current_Events, Top/-News/Media, Top/News/Online_Archives, Top/Recreation/-Audio, Top/Recreation/Motorcycles, Top/Recreation/Pets, Top/Recreation/Scouting, Top/Recreation/Travel, Top/-Reference/Encyclopedias, Top/Reference/Flags, Top/Reference/-Libraries, Top/Reference/Maps, Top/Reference/Museums, Top/-Regional/Africa, Top/Regional/Asia, Top/Regional/Europe, Top/Regional/Middle_East, Top/Regional/Oceania, Top/-Science/Astronomy, Top/Science/Earth_Sciences, Top/Science/-Institutions, Top/Science/Physics, Top/Science/Technology, Top/Shopping/Auctions, Top/Shopping/General_Merchandise, Top/Shopping/Gifts, Top/Shopping/Holidays, Top/Shopping/-Music, Top/Society/Folklore, Top/Society/Genealogy, Top/-Society/Law, Top/Society/People, Top/Society/Philosophy, Top/Sports/Bowling, Top/Sports/Darts, Top/Sports/Football, Top/Sports/Golf, Top/Sports/Gymnastics

Tab. C.2.: *Testhierarchie Hier4. Dargestellt sind alle Kategorien der Hierarchie in der entsprechenden Tiefe.*

Literaturverzeichnis

[BLHL01] Berners-Lee, Tim ; Hendler, James ; Lassila, Ora: The Semantic Web. 284 (2001), Mai, Nr. 5, S. 34–43. – ISSN 0036–8733 2.1

[Bur98] Burges, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: *Data Mining and Knowledge Discovery* 2 (1998), Nr. 2, S. 121–167 3.3.1

[CL01] Chang, Chih-Chung ; Lin, Chih-Jen: *LIBSVM: a library for support vector machines*, 2001 5.3

[DC00] Dumais, Susan T. ; Chen, Hao: Hierarchical classification of Web content. In: Belkin, Nicholas J. (Hrsg.) ; Ingwersen, Peter (Hrsg.) ; Leong, Mun-Kew (Hrsg.): *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*. Athens, GR : ACM Press, New York, US, 2000, S. 256–263 3.2, 4.2, 4.2

[DMK02] Dhillon, Inderjit S. ; Mallela, Subramanyam ; Kumar, Rahul: Enhanced word clustering for hierarchical text classification. In: *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA : ACM Press, 2002. – ISBN 1–58113–567–X, S. 191–200 4.2, 4.2

[FPW+99] Frank, Eibe ; Paynter, Gordon W. ; Witten, Ian H. ; Gutwin, Carl ; Nevill-Manning, Craig G.: Domain-Specific Keyphrase Extraction. In: *IJCAI*, 1999, S. 668–673 2.2.1

[Gra03] Granitzer, Michael: *Hierarchical Text Classification using Methods from Machine Learning*. A-8010 Graz, Austria, Institute of Theoretical Computer Science (IGI), Graz University of Technology, Diss., 2003 3, 3.2, 4.2, 4.2

[Joa98] Joachims, Thorsten: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, Claire (Hrsg.) ; Rouveirol, Céline (Hrsg.): *Proceedings of ECML-98, 10th European Conference on Machine Learning*. Chemnitz, DE : Springer Verlag, Heidelberg, DE, 1998, S. 137–142 3.3.1

[KFPH04] Kibriya, Ashraf M. ; Frank, Eibe ; Pfahringer, Bernhard ; Holmes, Geoffrey: Multinomial Naive Bayes for Text Categorization Revisited. In: Webb, G. I. (Hrsg.) ; Yu, X. (Hrsg.): *Proc 17th Australian Joint Conference on Artificial Intelligence* Bd. 3339. Cairns, Australia : Springer, 2004, S. 488–499 3.3.2

[KS97] Koller, Daphne ; Sahami, Mehran: Hierarchically classifying documents using very few words. In: *Proceedings of ICML-97, 14th International Conference on Machine Learning*. 1997, S. 170–178 4.2, 4.2

[Mla98] Mladenić, Dunja: *Machine Learning on Non-Homogeneous, Distributed Text Data*. Ljubljana, Slovenia, Faculty of Computer and Information Science, University of Ljubljana, Diss., 1998 3, 3.1.2, 3.2, 4.2, 4.2

[MRMN98] McCallum, Andrew K. ; Rosenfeld, Ronald ; Mitchell, Tom M. ; Ng, Andrew Y.: Improving text classification by shrinkage in a hierarchy of classes. In: Shavlik, Jude W. (Hrsg.): *Proceedings of ICML-98, 15th International Conference on Machine Learning*. Madison, US : Morgan Kaufmann Publishers, San Francisco, US, 1998, S. 359–367 4.2, 4.2, 4.3, 5, 4.2

[PNTK05] Pang-Ning Tan, Michael S. ; Kumar, Vipin: *Introduction to Data Mining*. Addison-Wesley, 2005. – ISBN 0–321–32136–7 3.3.3, 3.3.3

[Por97] Porter, M. F.: An algorithm for suffix stripping. (1997), S. 313–316. ISBN 1–55860–454–5 3.1.1

[RS02] Ruiz, Miguel E. ; Srinivasan, Padmini: Hierarchical Text Categorization Using Neural Networks. In: *Information Retrieval* 5 (2002), Nr. 1, S. 87–118 4.2, 4.2

[RY02] Rogati, Monica ; Yang, Yiming: High-performing feature selection for text classification. In: *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*. New York, NY, USA : ACM Press, 2002. – ISBN 1–58113–492–4, S. 659–661 3.1.2

[Sch01] Schapire, R. *The boosting approach to machine learning: An overview*. 2001 3.3.3, 3.3.3

[Seb02] Sebastiani, Fabrizio: Machine Learning in Automated Text Categorization. In: *ACM Computing Surveys* 34 (2002), Nr. 1, S. 1–47 3, 3.3.3

[SL01] Sun, Aixin ; Lim, Ee-Peng: Hierarchical text classification and evaluation. In: *ICDM*. 2001, S. 521–528 4.2, 4.2

[SM04] Stein, Benno ; Meyer zu Eissen, Sven: Topic Identification: Framework and Application. In: *Proceedings of the 4th International Conference on Knowledge Management (I-KNOW 2004)*, 2004 2.2.1

[SMW03] Stein, Benno ; Meyer zu Eissen, Sven ; Wissbrock, Frank: On Cluster Validity and the Information Need of Users. In: *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA'03)*. Upper Saddle River, NJ, USA : ACTA Press, 2003. – ISBN 0–88986–390–3,, S. 216–221 3.2

[WF05] Witten, Ian H. ; Frank, Eibe: *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, June 2005. – ISBN 0120884070 5.3, 5.7.4

[YP97] Yang, Yiming ; Pedersen, Jan O.: A comparative study on feature selection in text categorization. In: Fisher, Douglas H. (Hrsg.): *Proceedings of ICML-97, 14th International Conference on Machine Learning*. Nashville, US : Morgan Kaufmann Publishers, San Francisco, US, 1997, S. 412–420 3.1.2, 3.1.2

Stichwortverzeichnis