TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Entwurf einer effizienten Kategorisierung von Transaktionsdaten zur Schaffung eines Mehrwerts für Bankkunden

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt von:	Markus Glas
Studiengang:	Informatik
Matrikelnummer:	2693624
Erstprüfer:	Prof. Dr. Jens Albrecht
Zweitprüfer:	Prof. Dr.-Ing. Florian Gallwitz

Abstract

Durch die Einführung der Payment Services Directive 2 bricht das Monopol der Banken beim Zugriff auf die Kontodaten ihrer Kunden. Gerade Drittanbieter profitieren davon und bieten Bankkunden Anwendung zur zentralen Verwaltung unterschiedlicher Konten an. Auch die Kategorisierung der Transaktionen ist dabei ein wichtiger Bestandteil, um Ausgaben übersichtlicher darzustellen.

In dieser Arbeit wird untersucht, wie mit Hilfe etablierter Verfahren aus den Gebieten Information Retrieval und Textanalyse eine effiziente Kategorisierung von Banktransaktionen umgesetzt werden kann. Dazu werden sowohl Mittel der Computerlinguistik, als auch des Maschinellen Lernens (Machine Learning) betrachtet. Ziel ist es, eine Kategorisierung von Banktransaktionen zu entwerfen und in einem Prototypen umzusetzen.

Nach Erläuterung der Funktionsweise unterschiedlicher Textklassifikationsverfahren folgt eine Evaluation der Leistungsfähigkeit. Gleichzeitig wird eine Vorverarbeitung der Daten und Optimierung auf Banktransaktionen durchgeführt. Dabei wird gezeigt, welche Parametrisierung für eine Kategorisierung von Banktransaktionen optimale Ergebnisse liefert. Auf Basis dieser Erkenntnisse wird ein Modell für die spätere Implementierung gebildet.

Die anschließende Umsetzung geschieht in Form eines Prototypen, welcher das optimierte Modell nutzt. Dieser bildet ein eigenständiges Kategorisierungssystem und es wird gezeigt, wie dieser in späteren Anwendung zur Finanzplanung eingebunden werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Finanzindustrie befindet sich seit mehreren Jahren in einem digitalen Wandel. Zunehmend werden Geschäftsprozesse digitalisiert und traditionelle Dienstleistungen durch Web-Anwendungen und Mobile-Apps ergänzt oder gänzlich ersetzt [KS16, DSAH14]. Klassische Angebote wie Überweisungen, Auszahlungen, Kreditberatungen oder Kontoeröffnungen werden weitestgehend von IT-Systemen unterstützt oder autonom durchgeführt. Der Bedarf an Personal in den Filialen sinkt und die Verfügbarkeit und Performanz der IT-Systeme gewinnt immer mehr an Bedeutung. Mit steigender Nutzerzahl der Bankensysteme, steigt auch das Datenaufkommen und Banken verfügen über eine große Menge an potentiellen Informationen. Damit bietet sich die Möglichkeit, Analysen über Kontoaktivitäten zu erstellen und Korrelationen aufzuzeigen. Das Potential wurde längst erkannt und immer mehr Banken-Applikationen machen sich dies zu Nutzen, um den Endkunden einen Mehrwert hinsichtlich ihrer persönlichen Finanzplanung zu bieten.

1.1 Motivation

Die Zuordnung von Banktransaktionen in vordefinierte Kategorien, wie etwa *Lebenshaltung*, *Freizeit* oder *Versicherungen* ist, aufgrund der knappen und dennoch vielfältigen Informationen eines jeden Transaktionsdatensatzes, keine triviale Aufgabe. Neben der Analyse von Feldern mit festgelegter Struktur, wie der *Gläubiger-ID* oder *IBAN/BIC*, müssen auch Freitextfelder, wie der *Verwendungszweck* oder *Zahlungsempfänger*, untersucht werden. Da diese Felder oft keiner allgemein bekannten Konvention folgen, ist es notwendig die lexikalische Semantik zu untersuchen.

Als gängiges Verfahren werden häufig statische Regelwerke, mit einer Sammlung bekannter Transaktionen, eingesetzt. Die Kategorisierung von unbekannten Transaktionen ist damit jedoch nicht möglich. Die steigenden Popularität von Machine-Learning-Verfahren, vor allem durch die Verfügbarkeit einer Vielzahl freier Software-Bibliotheken und verteilten Rechenkapazitäten, geben einen Anreiz, neue Taxonomien für Banktransaktionen zu untersuchen.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist es, eine effiziente Kategorisierung von Banktransaktionsdaten zu entwerfen und in einem Prototypen umzusetzen. Dabei sollen zunächst geeignete Verfahren für die vorliegende Problemdomäne bewertet werden, um dem Nutzer ein zuverlässiges System anbieten zu können. Anhand einer prototypischen Umsetzung soll die Praxistauglichkeit getestet werden.

1.3 Methodik

Für den Entwurf der Kategorisierung soll ein Modell, auf Basis bewährter Algorithmen und Verfahren aus den Bereichen Data Mining und Textanalyse, erstellt werden. Dazu sollen diese vorab analysiert und deren Anwendbarkeit auf Banktransaktionen bewertet werden. Die Bewertung soll die Grundlage für den entstehenden Prototypen darstellen.

1.4 Umfeld

Diese Arbeit entstand in Kooperation mit der adorsys GmbH & Co. KG. Adorsys ist ein mittelständisches Software-Unternehmen mit Hauptsitz in Nürnberg. Die Firma wurde 2006 gegründet und ist seitdem als Spezialist für individuelle Software-Lösungen im Banken- und Versicherungssektor tätig. Zu den Kunden gehören unter anderem die Teambank AG, Schwäbisch Hall und ERGO Direkt. Die adorsys ist somit ein Partner mit viel Expertise im Finanzsektor und der Architektur von großen Software-Systemen.

1.5 Aufbau der Arbeit

Diese Arbeit besteht, inklusive der Einleitung, aus 6 Hauptkapiteln. Zur besseren Orientierung werden die Kapitel nachfolgend kurz erläutert.

Im folgenden Kapitel 2 wird zunächst die Ausgangssituation anhand des aktuellen Bankenmarktes und der zunehmenden Bedeutung von Banking-Applikationen erläutert. Im Anschluss werden die Anforderungen an das zu implementierende System festgelegt.

Kapitel 3 legt zunächst grundsätzliche Begriffe aus den Gebieten Information Retrieval und Data Mining und deren Verwendung in dieser Arbeit fest. Anschließend werden bestehende Verfahren und Algorithmen zur Textklassifikation erläutert und Möglichkeiten der Bewertung gezeigt.

Kapitel 4 greift die analysierten Verfahren und Algorithmen auf und legt deren Verwendung im späteren Prototypen fest. Dazu werden die Verfahren mit Hilfe von Trainingsdaten hinsichtlich Banktransaktionen optimiert. Im Anschluss wird die Leistungsfähigkeit bewertet, um ein optimales Modell für den Einsatz im späteren Prototypen zu entwerfen.

Kapitel 5 dokumentiert die Implementierung des Prototypen und aller zugehörigen Subsysteme. Die Umsetzung nutzt das zuvor entworfene Modell. Abschließend werden alle Schnittstellen und Komponenten erläutert, welche den Prototypen erweiterbar für spätere Anwendungen macht.

Kapitel 6 schließt diese Arbeit mit einem Fazit des entworfenen Systems und gibt einen Ausblick auf mögliche Erweiterungen.

2 Anforderungsanalyse

In diesem Kapitel wird die Ausgangssituation anhand aktueller Trends im Finanzwesen und Bankensektor dargelegt. Anschließend werden daraus die Anforderungen abgeleitet, um ein effizientes System zur Transaktionskategorisierung zu entwickeln.

2.1 Ist-Situation

Nachfolgend wird zunächst die Ist-Situation heutiger Banken näher betrachtet. Insbesondere werden technologische Veränderungen durch neue Richtlinien und Gesetze, im Kontext einer Transaktionskategorisierung, veranschaulicht.

2.1.1 Fachliche Betrachtung

Mit dem Projekt *Single Euro Payments Area* (SEPA) wurde der Grundstein für ein einheitliches Verfahren zur bargeldlosen Zahlung in Europa gelegt [Deu17b]. Neben den 28 EU-Staaten ist damit die Zahlung in sechs weiteren nicht EU-Staaten möglich. Zur Schaffung eines einheitlichen Rechtsrahmen wurde von der EU-Kommission eine Richtlinie für Zahlungsdienstleistungen (*Payment Services Directive* - PSD) geschaffen, welche in nationales Recht der teilnehmenden Staaten überging.

Jedoch blieb der rasante Anstieg von Drittanbietern in der, bereits 2007 verfassten (vgl. Abb. 2.1), Richtlinie unberücksichtigt. Vor allem die Vielzahl an mobilen Banking-Apps durch Drittanbieter blieb weitestgehend unbeachtet. Das Anbieten von Zahlungsdienstleistungen war in weiten Teilen den Banken vorbehalten.

Abbildung 2.1: Zeitstrahl von PSD zu PSD2

Die *Payment Services Directive 2* (PSD2) der Europäischen Union, welche im Januar 2018 in nationale Rechte innerhalb Europas überging, soll unter anderem das lukrative Monopol der Banken beim Zugriff auf Kontodaten brechen. Zukünftig müssen Banken auch Drittanbietern eine Schnittstelle (API) bereitstellen, um auf Wunsch des Kunden auf dessen Kontodaten zugreifen zu können. Mit Hilfe dieser Daten können anschließend weitere Dienste, wie etwa die Beratung zu Krediten, Wertpapieranlage oder allgemein zur persönlichen finanziellen Situation, angeboten werden.

FinTech

Unter dem Begriff *Financial Technology* oder kurz *FinTech* werden Finanzinnovationen und neue Wege der Finanzdienstleistung zusammengefasst [Deu18]. Umgangssprachlich wird häufig nur der Begriff FinTech verwendet, um ein FinTech-Unternehmen oder FinTech-Start-up zu beschreiben.

Grundsätzlich können FinTech-Unternehmen in vier Hauptkategorien eingeordnet werden, welche sich nach ihren Geschäftsbereichen orientieren [DHSW16]. Man unterscheidet dabei zwischen den klassischen Bereichen einer Universalbank: Finanzierung, Vermögensmanagement und Zahlungsverkehr (vgl. Abb. 2.2). FinTechs, die keiner dieser Kategorien zuzuordnen sind, werden unter *sonstigen FinTechs* gelistet. FinTech-Unternehmen, welche software- und appbasierte Dienste zur privaten Finanzplanung anbieten, werden unter dem Begriff Personal Financial Management (PFM) zusammengefasst. PFM zählt zum Bereich Vermögensmanagement wie in Abbildung 2.2 dargestellt.

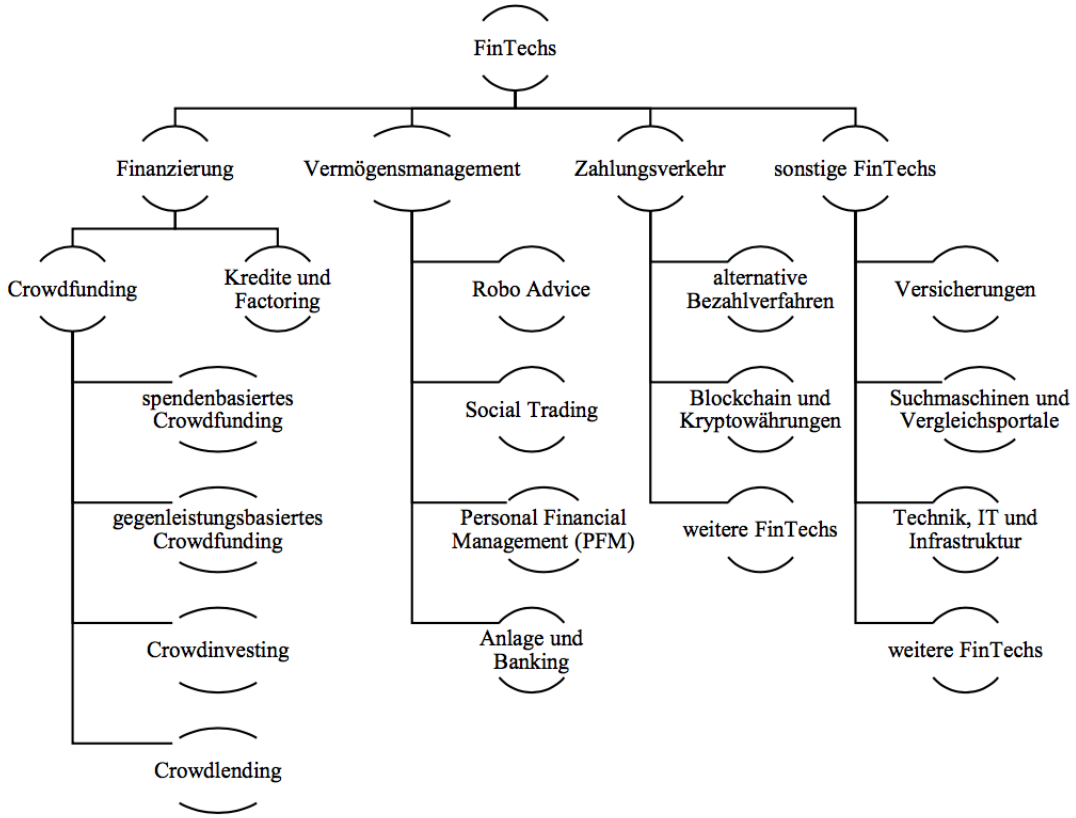

Abbildung 2.2: Kategorien der FinTech-Branche [DHSW16]

Aktuell angebotene PFM-Anwendungen verwenden, aufgrund fehlender Schnittstellen, oft das als *Web-* oder *Screen-Scraping* bekannte Verfahren zum softwarebasierten Auslesen von Bildschirmtexten. Dieses Verfahren hat diverse Nachteile und kann, vor allem bei sich häufig ändernden Benutzeroberflächen, zu Problemen bei der Erkennung von Webseiten-Elementen führen. Umso bedeutender ist es für Drittanbieter eine einheitliche Regelung für den Zugriff auf Kontodaten zu erhalten. PSD2 schafft einen Anreiz, zukünftig weitere PFM-Systeme zu entwickeln oder bestehende auszubauen.

Aktuell gibt es 699 aktive FinTech-Start-ups in Deutschland [com17]. Auch wenn die Neugründung der FinTech-Start-ups 2017 im Vergleich zu den Vorjahren zurückging, ist die Höhe des eingesetzten Risikokapitals (Venture Capital) weiterhin hoch. Zum Ende des dritten Quartals 2017 lag diese mit 579 Mio. Euro fast auf dem Niveau des gesamten Jahres 2016.

2.1.2 Technische Betrachtung

Überweisungen und Lastschriften folgen seit der Einführung von PSD einem einheitlichen Format, welches innerhalb des SEPA Credit Transfer Schema festgelegt ist [Eur17]. Zum Nachrichtenaustausch zwischen Bank und Kunde existiert ein spezielles Format namens Cash-Management (CAMT). Diese kann lokal leichte Abweichungen enthalten und werden in Deutschland in der Anlage 3 des DFÜ-Abkommens spezifiziert [Die17]. Unter anderem werden dort die Bezeichnung der einzelnen Felder, deren Länge und der Zeichensatz festgelegt.

```
{
   "owner": "MUSTER LEBENSVERSICHERUNG AG",
   "bic": "ABC54321",
   "accountNumber": "1234567890",
   ...
   "currency": "EUR",
   "valutaDate": [2017,11,21],
   "bookingDate": [2017,11,21],
   "amount": -50.00,
   "text": "LASTSCHRIFT",
   "usage": "123456789098 MLV LEBENSVERS. / 01.01.2017 50,00",
   ...
}
```

Listing 2.1: Ausschnitt einer SEPA-Transaktion

Listing 2.1 zeigt einen Ausschnitt einer Nachricht gemäß des CAMT-Schemas im JSON-Format. Von besonderem Interesse für die spätere Textanalyse sind dabei *Empfänger* (owner), *Buchungstyp* (text) und *Verwendungszweck* (usage), da diese häufig Rückschlüsse auf den Begünstigten zulassen.

Aktuelle Kategorisierungsverfahren

Zur Kategorisierung von Banktransaktionen wird, als gängiges Verfahren, auf statische Stichwort-Listen zurückgegriffen. Diese können, bei sorgfältiger Pflege und genügend Expertise über den Aufbau von Transaktionen, gute Ergebnisse liefern. Des Weiteren ist die zugrundeliegende Datenstruktur meist wenig komplex und in kurzer Zeit implementiert.

Tabelle 2.1 zeigt den typischen Aufbau der Felder *Zahlungsempfänger*, *Verwendungszweck* und *Buchungstyp* einer Transaktion anhand von drei anonymisierten Beispielen. Zu erkennen ist, dass in jeder Transaktion das Wort „Versicherung“ in dieser oder verkürzter Form vorkommt. Somit wäre die Kategorie *Versicherung* zutreffend.

Zahlungsempfänger	**Verwendungszweck**	**Buchungstyp**
MUSTER LEBENSVERSICHERUNG AG	123456789098 MLV LEBENSVERS. / 01.01.2017 50,00	Lastschrift
AUTOMOBIL VERSICHERUNGS-AG	123456789 GAV N-AB 123 / 01.01.2017 300,00	Lastschrift
Gesund BKK	1234567 AKV KRANKENVERS. R/ 01.01.2017 6,00	Lastschrift

Tabelle 2.1: Beispiel-Transaktionen zur Kategorie Versicherung

Die Stichwort-Listen zur Kategorisierung werden oft auch als manuell gepflegte Regeln (hand-coded-rules) oder standing queries bezeichnet [MRS08]. Der Abgleich der Regeln läuft meist sequentiell ab. Jeder vordefinierten Kategorie wird vorab eine Sammlung von Stichwörtern zugeordnet, die manuell gepflegt werden. Als Wiedererkennungswert dienen oft die Firmenbezeichnungen im Empfängerfeld oder bestimmte Wörter im Verwendungszweck. Einzelne Stichwörter können dabei auch in Regeln zusammengefasst werden. Meist kommen dabei Reguläre Ausdrücke oder Boolesche Operatoren zum Einsatz.

Eine Regel für die, in Tabelle 2.1 dargestellten, Transaktionen zur Kategorie Versicherung könnte sich wie folgt darstellen:

$$(\textit{Versicherung} \quad \textit{OR} \quad \textit{Lebensversicherung} \quad \textit{OR} \quad \textit{Krankenvers})$$

Da meist nicht alle Wortkombinationen in den Regeln enthalten sind, ist die Kategorisierung oft ungenau. Des Weiteren reicht eine geringfügige Veränderung der Transaktionsdaten aus, etwa das Einfügen eines Leerzeichen innerhalb des Terms „Versicherung“, um keine eindeutige Zuordnung treffen zu können. Die manuelle Pflege und Anpassung der einzelnen Regeln ist zudem fehleranfällig und zeitaufwändig.

Ein weiteres Defizit manuell gepflegter Regeln ist, dass unbekannte Transaktionen gänzlich unkategorisiert bleiben. Abhilfe könnte die Verwendung von Ähnlichkeitsmaßen, unscharfen Suchen und Machine-Learning-Verfahren schaffen.

2.2 Soll-Konzept

Um eine effiziente Kategorisierung von Transaktionsdaten zu entwerfen, sind gewisse Anforderungen an den Business-Kontext und die technische Umsetzung erforderlich. Somit lassen sich diese in fachliche und technische Anforderungen aufteilen.

2.2.1 Fachliche Anforderungen

Fachliche Anforderungen beschreiben Bedingungen, welche das System erfüllen muss, um Bankkunden einen Mehrwert hinsichtlich ihrer privaten Finanzplanung zu schaffen.

Auswahl geeigneter Kategorien
Vor der Kategorisierung ist es notwendig, geeignete Kategorien auszuwählen. Hierfür können Kategorien aus Publikationen zu Lebenshaltungskosten und Konsumausgaben des Statistischen Bundesamtes dienen[1]. Die Tabellen zu Laufende Wirtschaftsrechnungen (LWR) und zur Einkommens- und Verbrauchsstichprobe (EVS) geben bereits eine grobe Aufteilung nach Art der Konsumausgaben vor[2].

Jedoch können diese nur als Ausgangspunkt dienen und umfassen nicht alle Bereiche, welche in Bankumsätzen auftreten können. Des Weiteren werden relevante Bereiche zur privaten Finanzplanung nur abstrakt abgedeckt. Beispielsweise ist die, für die private Finanzplanung relevante Aufteilung von Versicherungen und anderen Finanzdienstleistungen, wie etwa Hypotheken oder Wertpapieren, nicht vorhanden. Zudem fehlen bankspezifische Kategorien, wie etwa die Barentnahme, welche einen großen Teil der gesamten Transaktionsdaten einnehmen kann und deshalb ebenfalls kategorisiert werden muss. Neben dem Statistischen Bundesamt dienen auch Erfahrungen aus anderen PFM-Systemen und -Apps, um geeignete Kategorien zu finden.

Durch die enge Kooperation von adorsys mit einigen großen deutschen Banken, werden deren Kompetenzen herangezogen, um eine sinnvolle Einteilung zu gestalten. In Kooperation mit adorsys wurden die Hauptkategorien, wie in Tabelle 2.2 dargestellt, gewählt.

[1] https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/Konsumausgaben/Konsumausgaben.html
[2] https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/Konsumausgaben/Tabellen/Gebietsstaende.html

Kategorien	Beispiele
Barentnahme	Geldautomaten (Inland/Ausland)
Finanzen	Spenden, Kredite, Sparen
Freizeit & Lifestyle	Shopping, Reisen, Unterhaltung
Lebenshaltung	Lebensmittel, Gesundheit
Mobilität & Verkehrsmittel	sämtl. KFZ-Kosten, ÖPNV-Fahrkarten
Sonstiges	-
Versicherungen	Kranken-, Haftpflicht-, Lebensversicherung
Wohnen & Haushalt	Miete, Nebenkosten, Einrichtung

Tabelle 2.2: Kategorien

Rückweisungsklasse

Da nicht ausgeschlossen werden kann, dass Transaktionen aufgrund fehlender Trainingsdaten oder fehlerhafter Benennungen nicht kategorisiert werden, ist es notwendig, eine Rückweisungsklasse bereitzustellen. Im vorliegenden Fall übernimmt die Kategorie „Sonstiges“ diese Aufgabe. Dabei ist auch darauf zu achten, dass gänzlich unbekannte Transaktionen nicht fälschlicherweise einer der anderen Kategorien zugeordnet werden. Andererseits soll das System neue Transaktionen der bestmöglichen Kategorie zuordnen. Die Zuweisung Rückweisungsklasse soll erst nach Abgleich mit den übrigen Kategorien geschehen.

SEPA Category Purpose Code

Bereits vor der Vereinheitlichung des europäischen Zahlungsverkehrs durch das SEPA-Projekt wurden, im Rahmen des Datenträgeraustauschverfahrens (DTA), Textschlüssel eingeführt, um wiederkehrende Überweisungen und Lastschriften kategorisieren zu können. In SEPA wurden die Textschlüssel in den *SEPA Category Purpose Code* transferiert.

Bestimmte Zeichenketten, wie „Cred+“, „Einreicher-ID“ oder „Mandat“ können Informationen zu Gläubiger oder der Mandatsreferenz enthalten und müssen deshalb bei der Textanalyse des Verwendungszweck entsprechend behandelt werden.

2.2.2 Technische Anforderungen

Technische Anforderungen umfassen alle Bedingungen, die das System erfüllen muss, um die Kategorisierung möglichst effizient und zuverlässig durchzuführen.

Zweistufiges-System

Wie im vorigen Kapitel angesprochen, werden die Freitextfelder häufig sehr unterschiedlich bestückt. Eine reine Textanalyse gestaltet sich deshalb schwierig und es müssen, neben den Feldern Zahlungsempfänger und Verwendungszweck, weitere Kriterien hinzugezogen werden. Neben den Freitextfeldern, besitzt jede Banktransaktion, gemäß den SEPA-Vorgaben, weitere Felder, welche jedoch nicht alle Pflichtfelder sind und je nach Buchungstyp variieren [Eur17].

Bei Lastschriften kann als Identifikation des Unternehmens die Gläubiger-ID dienen (vgl. Tabelle 2.3). Diese wird von der Deutschen Bundesbank an Unternehmen ausgegeben, welche am SEPA-Lastschriftverfahren teilnehmen [Deu17a]. Eine Anforderung ist deshalb, ein mehrstufiges System zu entwerfen, um bei Lastschriften zuerst die Gläubiger-ID zu überprüfen. Ist das zugehörige Unternehmen bereits bekannt, kann eine erste Entscheidung zur Kategorisierung getroffen werden.

Zahlungsempfänger	**Verwendungszweck**	**Buchungstyp**	**Gläubiger ID**
MUSTER LEBENSVERSICHERUNG AG	...	Lastschrift	**DE96ZZZ00000012345**
AUTOMOBIL VERSICHERUNGS-AG	...	Lastschrift	**DE83ZZZ00000054321**
Gesund BKK	...	Lastschrift	**DE47ZZZ00000098765**

Tabelle 2.3: Beispiel-Transaktionen zur Kategorie Versicherung mit Gläubiger-ID

Anders gestaltet es sich bei Transaktionen des Typs *Kartenzahlung*, wie etwa bei Kartenzahlungen mittels eines EC-Kartenterminals in Supermärkten oder Tankstellen. Wird die Zahlung durch Eingabe der PIN bestätigt, ist die Gläubiger-ID in der Transaktion nicht sichtbar, da keine Lastschrift stattfindet (vgl. Tabelle 2.4). Ausnahme bilden Kartenzahlungen, die statt der Eingabe der PIN, eine Unterschrift verlangen. Hier findet eine Lastschrift gemäß SEPA-Richtlinie statt und somit wird eine Gläubiger-ID verpflichtend [Deu17a].

Zahlungsempfänger	Verwendungszweck	Buchungstyp	Gläubiger ID
DANKE, IHR LIDL//-Nuernberg/DE	2017-01-01T10:00:00 Karte1 2017-12	Kartenzahlung	-
NORMA SAGT DANKE//Nuernber/-DE	2017-02-01T11:00:00 Karte1 2017-12	Kartenzahlung	-
AUGENOPTIK SCHMIDT	1234567890987654321 ELV12345678 01.01 10.00 ME1	SEPA-ELV-Lastschrift	DE31ZZZ00001900514
6850 EDEKA//ROT-TENDORF/DE	2017-03-01T12:00:00 Karte1 2017-12	Kartenzahlung	-

Tabelle 2.4: Beispiel-Transaktionen zur Kategorie Lebenshaltung

Als weiteres Indiz kann in diesen Fällen die Kontonummer des Empfängers dienen. Da große Unternehmen jedoch häufig mehrere unterschiedliche Konten besitzen, ist eine eindeutige Zuordnung, wie bei der Gläubiger-ID, nicht möglich. Somit ergibt sich die Anforderung, dass bei nicht Vorhandensein einer Gläubiger-ID, zwangsläufig eine Textanalyse der Freitextfelder durchgeführt werden muss.

Erweiterbarkeit des Modells

Damit auch zukünftig Transaktionen zuverlässig kategorisiert werden können, muss eine Möglichkeit vorhanden sein, das System hinsichtlich neuer Transaktionen zu erweitern. Das verwendete Verfahren soll deshalb kontinuierlich, durch Ergänzung weiterer Transaktionen, verbessert werden. Dazu wird eine Schnittstelle benötigt, welche es erlaubt weitere Daten in das System einzupflegen.

Rückmeldung durch die Benutzer

Für den Fall, dass eine Transaktion nicht kategorisiert werden kann und somit in der Rückweisungsklasse eingeordnet wird, soll dem Benutzer die Möglichkeit gegeben werden, manuell eine Kategorie zu bestimmen. Die so gewonnene Information wird zur Verbesserung der Kategorisierung genutzt, indem die manuell kategorisierte Transaktion in das System eingepflegt wird.

3 Verfahren zur Textanalyse

In diesem Kapitel werden verschiedene Verfahren zur Textanalyse, hinsichtlich ihrer Anwendbarkeit auf die im vorigen Kapitel gestellten Anforderungen, untersucht. Zunächst wird eine Einführung in die Bereiche Data Mining und Information Retrieval gegeben. Anschließend werden Terminologien und deren Verwendung in dieser Arbeit definiert. Das Kapitel schließt mit der Erläuterung verschiedener Bewertungsverfahren für Klassifikatoren.

3.1 Fachgebiete und Terminologien

Um Daten effizient auswerten zu können, ist es notwendig spezielle Datenanalyse-Techniken zu verwenden [CL16]. Insbesondere bei großen Datenmengen ist der Einsatz notwendig, da die manuelle Analyse bei Massendaten schnell an ihre Grenzen stößt.

3.1.1 Data Mining

Unter Data Mining versteht man die Suche nach Mustern oder Zusammenhängen in Daten, um daraus Wissen zu extrahieren [CL16].

Verarbeitungsschritte im Data Mining
Um eine korrekte Extraktion von Wissen zu erreichen, wird Data Mining in mehrere Phasen aufgeteilt [CL16]:

1. Selektion: Auswahl der geeigneten Datenmenge
2. Verarbeitung: Behandlung fehlerhafter oder fehlender Daten
3. Transformation: Umwandlung in adäquate Datenformate
4. Data Mining: Suche nach Mustern
5. Evaluation und Interpretation: Interpretation der Ergebnisse und Auswertung

Der gesamte Prozess wird auch unter dem Begriff *Knowledge Discovery in Databases (kurz KDD)* verstanden, worin Data Mining eine eigene Phase bildet. Abbildung 3.1 zeigt den typischen Ablauf eines KDD-Prozesses.

KDD und Data Mining werden jedoch auch synonym verwendet. Im weiteren Verlauf dieser Arbeit wird lediglich der Begriff Data Mining, mit entsprechenden Hinweisen zur Abgrenzung zwischen Prozess und Phase, verwendet.

Die drei ersten Phasen *Selektion*, *Verarbeitung* und *Transformation* werden unter dem Begriff *Datenvorverarbeitung* zusammengefasst und umfassen alle notwendigen Tätigkeiten, um die Daten in ein einheitliches Format zu fassen. Dazu zählen u.a. die Behandlung fehlender oder fehlerhafter Daten und die Umwandlung in ein adäquates Datenformat [CL16].

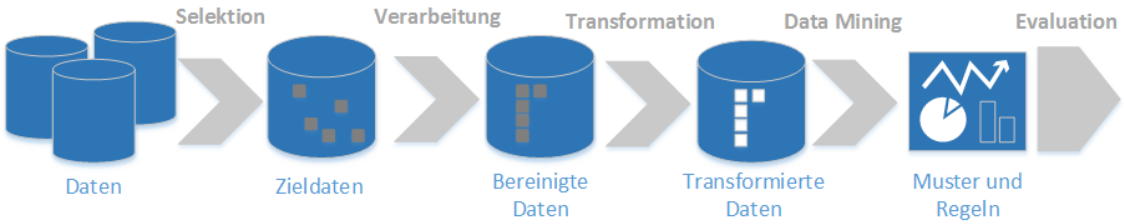

Abbildung 3.1: Ablauf eines Data-Mining-/KDD-Prozesses [CL16]

Die Phase *Data Mining* umfasst, wie bereits anfangs des Kapitels erwähnt, die Suche nach Mustern oder Zusammenhängen.

Text Mining

Textanalyse, auch Text Mining genannt, beschreibt eine Sonderform des Data Mining auf textuellen Daten [BM14]. Dabei spielen, neben dem üblichen Data-Mining-Prozess, linguistische Prozesse eine entscheidende Rolle. Der Aufbau domänenspezifischer Ontologien ist ebenso Bestandteil des Text Mining, wie die automatisierte Überprüfung von Syntax, Semantik und Pragmatik. Es bildet somit einen wichtigen Baustein für natürlichsprachliche Systeme.

3.1.2 Merkmalsextraktion

Merkmalsextraktion beschreibt den Prozess zur Auswahl einer Untermenge an Termen innerhalb eines Trainings-Datensatzes [MRS08]. Die Merkmale werden anschließend zum Training eines Text-Klassifikators genutzt. Die Hauptziele der Merkmalsextraktion liegen zum einen in der Verkleinerung des Vokabulars, zum anderen in der Eliminierung von Noise-Features, um die Genauigkeit zu erhöhen. Noise-Features erhöhen die Fehlerrate eines Klassifizierers durch eine falsche Generalisierung, beispielsweise auf Grundlage eines selten auftretenden Terms im Trainingssatz.

3.1.3 Information Retrieval

Der Begriff Information Retrieval (IR) kann sehr weitläufig verwendet werden. Sei es das einfache Nachschlagen einer Telefonnummer oder eine komplexe Suchanfrage in einer Web-Suchmaschine. Meist wird damit das Auffinden von Informationen in großen Datenmengen beschrieben [Fer03]. Während früher vor allem reine Textdokumente, wie Bücher oder Zeitungsartikel durchsucht wurden, findet IR heutzutage auch auf Webseiten im HTML-Format oder anderen Datenformaten, wie XML, statt. Im Bereich Big-Data werden oft Techniken und Verfahren aus dem IR in ihrer ursprünglichen oder angepasster Form angewendet [Fer03]. Ein gängiges Verfahren ist dabei die Klassifikation, welche in den nachfolgenden Abschnitten erläutert und deren Anwendbarkeit auf das Anwendungsszenario Banktransaktionen geprüft wird.

3.2 Datenvorverarbeitung

Nachfolgend werden Verfahren zur Vorverarbeitung von textuellen Daten beschrieben. Dabei wird speziell deren Anwendbarkeit auf Banktransaktionen berücksichtigt.

3.2.1 Tokenisierung

Vor der Textanalyse und -verarbeitung ist es notwendig, Texte in linguistische Einheiten zu segmentieren [CEE+09]. Tokenisierung beschreibt ein Segmentierungsverfahren, welches eine Wortsequenz in ihre Einzelteile, *Tokens* genannt, aufteilt. Ein trivialer Ansatz ist die Trennung anhand der Leerzeichen, wie nachfolgend gezeigt.

Eingabe:	DANKE, IHR SUPERMARKT//Nuernberg//DE		
Ausgabe:	DANKE,	IHR	SUPERMARKT//Nuernberg//DE

Leerzeichen-Tokenisierung

Oft wird dieser Schritt mit der Entfernung bzw. Ersetzung von Interpunktionen und Sonderzeichen kombiniert. Das vorherige Beispiel ändert sich wie folgt.

Eingabe:	DANKE, IHR SUPERMARKT//Nuernberg//DE				
Ausgabe:	DANKE	IHR	SUPERMARKT	Nuernberg	DE

Tokenisierung und Bereinigung

Tokens werden nachfolgend auch als Wörter oder Terme bezeichnet. Ein Typ bezeichnet eine Sammlung aller Tokens mit derselben Zeichenfolge.

Die Segmentierung eines Textes anhand der Leer- und Sonderzeichen bildet eine gute Grundlage für die weitere Verarbeitung, jedoch gibt es Fälle, in denen eine Trennung unerwünscht ist. Diese Fälle sind stark sprachabhängig. So sind Komposita in der englischen Sprache oft durch einen Bindestrich oder Leerzeichen getrennt, in der deutschen Sprache des Öfteren gänzlich ohne Trennung [MRS08].

Bei Produkt- oder Firmen-Namen ist die Verwendung von Leerzeichen auch in der deutschen Sprache nicht ungewöhnlich. Sei es durch die Angabe der Unternehmensform, etwa *Volkswagen AG*, oder die Zusammensetzung zweier Worte, wie *Deutsche Bahn*. Eine Trennung nur anhand der Leer- und Sonderzeichen kann somit zu unerwünschten Resultaten führen und ist nicht ausreichend.

Gängige Bibliotheken zur Computerlinguistik, wie etwa das Natural Language Toolkit[1] (NLTK), bieten deshalb Tokenisierungs-Funktionen an, welche auf großen Text-Korpora unterschiedlicher Sprachen trainiert werden. IR-Systeme für die deutsche Sprache verwenden zudem häufig *Komposita-Trenner* (engl. compound-splitter), welche die Existenz eines Wortes innerhalb eines anderen Wortes erkennen [MRS08].

3.2.2 Stoppwörter

Häufig vorkommende Wörter, wie Bindewörter oder Artikel, haben oft keinen Mehrwert hinsichtlich der Kategorisierung eines Dokuments und können aufgrund ihrer hohen Anzahl das Ergebnis verfälschen. Diese Wörter werden als *Stoppwörter* bezeichnet. Um sie zu entfernen werden Filterlisten, auch bekannt als Stopplisten, verwendet. Die Länge der Stopplisten kann stark variieren und ist vom jeweiligen Einsatzzweck abhängig. Große Stopplisten können mehrere hundert Wörter, kleine lediglich zehn Wörter oder weniger enthalten. Dabei ist auch zu beachten, dass durch Stopplisten Kontext verloren gehen kann, weshalb auf deren Einsatz bei Suchmaschinen und IR-Systemen oft verzichtet wird [MRS08].

Alternativ zu Stopplisten kann das Auftreten der Terme gezählt werden. Anschließend werden die am häufigsten auftretenden Terme einer Dokumenten-Sammlung entfernt [MRS08].

[1] http://www.nltk.org/

3.2.3 Normalisierung

In einigen Fällen lassen sich gleichbedeutende Wörter nicht durch den reinen Abgleich der Zeichenfolge feststellen. Unterschiedliche oder fehlerhafte Schreibweisen erfordern eine Vernachlässigung von oberflächlichen Unterschieden. Die Normalisierung von Tokens beschreibt den Prozess, um gleichbedeutende Wörter ohne Berücksichtigung ihrer genauen Zeichenfolge kenntlich zu machen [MRS08].

Eine einfache Variante ist die Bildung von Äquivalenzklassen. Dabei wird die Klasse häufig nach einem der enthaltenen Wörter benannt. Beispielsweise könnte eine Äquivalenzklasse *Krankenversicherung* die Terme *Kranken-Versicherung* und *Kranken-versicherung* enthalten.

Lowercasing - Kleinschreibung
Zur Normalisierung gehört ebenfalls, die Terme in ein einheitliches Format zu ändern. Üblicherweise werden alle Buchstaben in Kleinschreibung geändert. Sofern keine Interpretation der Wörter anhand ihrer Schreibweise stattfindet, etwa zur Erkennung von Grammatiken, ist dies ein probates Mittel.

3.2.4 Stemming und Lemmatisierung

Stemming und Lemmatisierung bilden Möglichkeiten, um ein Wort auf dessen Wortstamm zurückzuführen. Ziel ist es, unterschiedliche Schreibweisen, etwa durch Deklinationen oder Konjugationen, auf dasselbe Wort zu vereinheitlichen.

Beim Stemming, auch Stammwortreduktion genannt, wird eine lexikonfreie Suffixanalyse durchgeführt [CEE+09]. Wörter mit unterschiedlichen Suffixen werden auf denselben Teilstring reduziert, der jedoch keinem Wort, gemäß einer vorgeschriebenen Grammatik, entsprechen muss. Beispielsweise können die Wörter *Statistik*, *statistisch* und *Statistiker* auf den Teilstring „*statisti*“ reduziert werden.

Lemmatisierung kann dabei als eine Sonderform des Stemming betrachtet werden und bietet eine Form der lexikografischen Reduktion. Das entstandene Lemma ist somit immer ein lexikografisch korrektes Wort. Eine Reduktion des vorigen Beispiels wäre somit auf das Wort *Statistik* korrekt [CEE+09].

3.3 Vektorraum-Modell

In einigen Verfahren zur Textklassifikation geschieht die Repräsentation eines oder mehrerer Dokumente in Form von Vektoren. Die als *Vektorraum-Modell* (engl. Vector Space Model - VSM) bezeichnete Repräsentation ist ein gängiges Verfahren zur Suche, Klassifikation und Clustering von Text-Dokumenten [MRS08].

Zur Repräsentation eines Dokuments als Vektor wird zunächst die Gewichtung der einzelnen Terme ermittelt. Im Anschluss daran können Dokumente im Vektorraum abgebildet werden. Der Vektorraum stellt hier einen n-dimensionalen linearen Raum dar, welcher für jedes Wort einer Wortsammlung, dem Vokabular, eine Dimension repräsentiert. Zur Veranschaulichung zeigt Abbildung 3.2 einen Vektorraum mit zwei Dimensionen, respektive einem Vokabular vom Umfang zwei (*gossip, jealous*). Innerhalb des Vektorraums sind drei Dokumente $\vec{v}(d_1)$, $\vec{v}(d_2)$ und $\vec{v}(d_3)$ und eine Suchanfrage $\vec{v}(q)$ vorhanden.

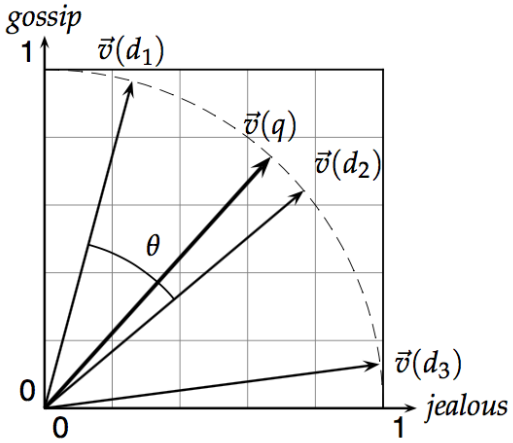

Abbildung 3.2: Vector Space Model [MRS08]

3.3.1 Termgewichtung

In Booleschen Regeln, wie die Regel zur Kategorisierung einer Versicherungs-Transaktion aus Kapitel 2.1, wird lediglich anhand der Existenz eines Terms entschieden, ob die Transaktion zu einer Kategorie gehört oder nicht. Die Anzahl der Terme findet dabei keine Beachtung [MRS08]. Zudem werden Stoppwörter in gleichem Maße behandelt wie die, für eine Kategorisierung weitaus bedeutenderen, Eigennamen und Substantive. Um eine differenzierte Betrachtung zu erhalten, werden Terme in Textanalyse-Verfahren gewichtet.

TF-IDF und Bag-of-words

Der einfachste Ansatz ist, die Gewichtung mit dem Auftreten eines Terms t in einem Dokument gleichzusetzen. Tritt gesuchter Term in einem Dokument oder im hier beschriebenen Fall einer Transaktion häufiger auf, so resultiert dies in einer höheren Gewichtung [MRS08]. Diese Form der Gewichtung wird als *Termhäufigkeit* oder *term frequency* (tf) bezeichnet.

$$\mathrm{tf}_{t,d} = \sum_{t' \in d} f_{t',d} \tag{3.1}$$

Für ein Dokument werden die einzelnen Gewichtungen üblicherweise als quantitativer Auszug betrachtet und resultieren in einer Sammlung von Gewichten, welche als *Bag-of-words* (BoW) bezeichnet werden. Die Anordnung der Terme in einem Dokument bleibt dabei gänzlich unbeachtet, was in einzelnen Fällen zu Nachteilen in der anschließenden Klassifizierung führen kann [MRS08]. Des Weiteren werden alle Terme eines Dokuments mit gleicher Priorität und somit völlig isoliert betrachtet. Gerade bei Dokumenten, mit einem hohen Aufkommen an Stoppwörtern ist es notwendig, diese bei der Gewichtung explizit auszuschließen.

Ein weiterer Ansatz diesem Problem entgegenzuwirken ist, die Anzahl der Dokumente einer Sammlung (N), welche einen Term enthalten, zu zählen. Somit erhält man die Dokumentenhäufigkeit oder *document frequency* [MRS08]. Intuitiv versucht man damit, die wenigen Dokumente mit einem seltenen Term höher zu gewichten, als die vielen Dokumenten mit einem häufig auftretenden Term. Da man für seltene Terme eine höhere Gewichtung erhalten möchte, wird die inverse Dokumentenhäufigkeit oder auch *inverse document frequency* (idf) berechnet.

$$\mathrm{idf}_f = log \frac{N}{df_t} \tag{3.2}$$

Somit ist die idf für einen seltenen Term hoch und für häufigere Terme niedrig. Kombiniert man diese mit der Gewichtung zur Termhäufigkeit, erhält man das Gewichtungsverfahren *term frequency - inverse document frequency* kurz *tf-idf*.

$$\text{tf-idf}_{t,d} = tf_{t,d} \times idf_t \tag{3.3}$$

N-Gramm-Merkmale

N-Gramme entstehen durch die Fragmentierung eines Textes in N-Zeichen-lange aufeinanderfolgende Fragmente [CTo94]. Zur Merkmalsextraktion werden N-Gramme

als Erweiterung zur Optimierung von Bag-of-words genutzt. Statt jedes Wort einzeln zu zählen, werden aufeinanderfolgende Wörter zu einem Feature zusammengefasst.

3.3.2 Abstands- und Ähnlichkeitsmaße

Zeichenketten können auf unterschiedliche Weise miteinander verglichen werden, um deren Ähnlichkeit zu bestimmen. Der Abgleich jedes einzelnen Zeichens ist dabei immer noch ein populäres Mittel. Aufgrund fehlerhafter Eingaben oder verschiedener Schreibweisen bei der Suche, ist dieses Vorgehen jedoch nicht immer sinnvoll [IMF13]. Deshalb werden im Text Mining Ähnlichkeitsmaße eingesetzt, oft auch als Fuzzy-matching beschrieben. Ein gängiges Vorgehen ist dabei, zuerst die Unähnlichkeit, definiert durch den Abstand zweier Zeichenketten x und y, zu messen [CL16]. Daraufhin lässt sich die Ähnlichkeit bestimmen. Je geringer der Abstand $dist(x,y)$ ist, desto größer ist die Ähnlichkeit der beiden Zeichenketten. Zwei identische Zeichenketten haben somit den Abstand 0.

$$dist(x,y) = 0 \quad genau\ dann,\ wenn\ x = y. \tag{3.4}$$

Jaccard-Koeffizient

Der Jaccard-Koeffizient folgt der Annahme, dass zwei gleiche Zeichenketten mehr übereinstimmende Zeichen enthalten, als die restliche Menge einer Sammlung [IMF13]. Zur Berechnung werden die gemeinsamen Zeichen der beiden Strings, als Schnittmenge von A und B durch deren Vereinigungsmenge geteilt:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.5}$$

Levenshtein-Distanz

Die Levenshtein-Distanz, auch Editierdistanz genannt, bietet eine weitere Möglichkeit, um die Ähnlichkeit zweier Zeichenketten festzustellen. Die Distanz spiegelt dabei die Anzahl der Editier-Operationen wider [IMF13]. Diese Operationen können je nach Implementierung variieren, es sind jedoch drei Operationen typisch: Einfügen, Löschen und Ersetzen. Zwei identische Terme benötigen keine Editier-Operation und resultieren somit in der Editierdistanz 0.

Kosinus-Ähnlichkeitsmaß

Die Ähnlichkeit zweier Dokumente wird im Vektorraum-Modell über den Abstand der Vektoren zueinander berechnet. Nachteil hierbei ist, dass bei Dokumenten unterschied-

licher Länge ein signifikanter Längenunterschied entstehen kann [MRS08]. Trotz hoher Ähnlichkeit der Wortfolge, resultiert dies in einem hohen Abstand im Raum. Um diesem Verhalten entgegenzuwirken, wird die Kosinus-Ähnlichkeit verwendet. Zur Veranschaulichung wird die Berechnung der Kosinus-Ähnlichkeit zweier Dokumente d_1 und d_2 betrachtet (vgl. Gleichung 3.6).

$$sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|} \tag{3.6}$$

Zunächst wird das Skalarprodukt der beiden Vektoren, hier $\vec{V}(d_1)$ und $\vec{V}(d_2)$, berechnet und durch das Produkt der euklidischen Länge geteilt. Die Teilung im Nenner hat zur Folge, dass die Vektoren zu Einheitsvektoren $\vec{v}(d_1) = \vec{V}(d_1)/|\vec{V}(d_1)|$ und $\vec{v}(d_1) = \vec{V}(d_1)/|\vec{V}(d_1)|$ werden. Somit lässt sich die Formel als Skalarprodukt der beiden Einheitsvektoren darstellen: $sim(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2)$. Dieses Maß gibt den Kosinus des Winkels θ zwischen den beiden Vektoren an, wie bereits in Abbildung 3.2 zu sehen.

Die Notation $sim(d_1, d_2)$ ist notwendig, um die Ähnlichkeit eines Dokuments $\vec{v}(d_1)$ zu allen anderen Dokumenten einer Sammlung $\vec{v}(d_1), \ldots, \vec{v}(d_N)$ zu beschreiben. Hierzu wird das größte Skalarprodukt $d_1 \cdot d_i$ genommen.

Für die Verwendung in IR ist eine Kosinus-Ähnlichkeit von 0, sehr unterschiedlich, bis 1, identisch, möglich. Dies wird durch den Einsatz einer Gewichtung der Terme zwischen 0 und 1 erreicht.

3.4 Klassifizierung und Kategorisierung

Klassifizierung ist ein Teilbereich der Mustererkennung und beschreibt die Zuordnung eines Musters zu einer von mehreren möglichen Klassen [Nie03]. Ein Muster kann dabei ein Schriftzeichen oder ein Wort darstellen, welches unabhängig von anderen Mustern klassifiziert wird. Klassen werden dabei oft auch als Kategorien oder Label bezeichnet, weshalb die Begriffe Klassifizierung und Kategorisierung in der Literatur, wie auch in dieser Arbeit, synonym verwendet werden [MRS08]. Als weiteres Synonym wird der Begriff Klassifikation benutzt. Nachfolgend werden unterschiedliche Verfahren zu Klassifizierung und deren Einsatzbereiche betrachtet.

3.4.1 Textklassifikation

Die Textklassifikation stellt eine Sonderform der Klassifikation dar und beschreibt die Zuordnung von Text-Dokumenten in vordefinierte Klassen [MRS08]. Die Form der Text-Dokumente kann dabei sehr unterschiedlich sein, etwa in Form von E-Mails, XML-Dokumenten, Webseiten im HTML-Format oder einfachen Text-Dokumenten. Ziel der Textklassifikation ist es, eine der vorab definierten Klassen einem Dokument zuzuordnen. Als Ergebnis wird die Klasse mit der größten Übereinstimmung ausgegeben.

Die Zuordnung kann mittels drei Varianten durchgeführt werden: Manuelle Klassifizierung, Statische Regeln oder Machine Learning [MRS08]. Die manuelle Klassifikation kann mit der Aufgabe eines Bibliothekars verglichen werden. Nicht zwingend ist ein Software-System zur Klassifikation notwendig und in der Vergangenheit war die manuelle Zuordnung zur Lösung von Klassifikationsproblemen ein gängiges Mittel. Im Falle von statischen Regeln geschieht dies mit einer Reihe von manuell erstellten Regeln, welche meist Wörter oder Wortkombinationen enthalten. Dies entspricht dem aus Kapitel 2.1.2 bekannten Verfahren der Stichwort-Listen.

Um eine Transaktion der Kategorie *Versicherungen* zu erkennen, könnte diese auf unterschiedliche Terme der Versicherungssparten untersucht werden. Diese können anschließend mit einem logischen Oder (*OR*) verknüpft werden, um das Auftreten mindestens eines Terms zu prüfen. Die Regel lautet dann wie folgt:

$$(Versicherung \quad OR \quad Lebensversicherung \quad OR \quad Krankenversicherung)$$

Als Ergebnis liefert diese Regel eine binäre Aussage über die Zugehörigkeit einer Klasse. Die Genauigkeit kann bei diesem Verfahren sehr hoch sein, jedoch ist die Erstellung und Pflege der Regeln sehr aufwändig, wie bereits in Kapitel 2.1.2 thematisiert. Aufgrund der genannten Nachteile soll dieses Vorgehen hier nicht weiter analysiert werden.

Bei der dritten Variante, dem Machine Learning, werden die Regeln zur Klassifizierung automatisch, anhand von Trainingsdaten, erstellt. Dieser Ansatz und die Umsetzung in Form verschiedener Algorithmen, wird in den folgenden Abschnitten näher untersucht.

Formale Definition der Textklassifikation

Ziel der Textklassifikation ist es, Dokumente d einer Klasse c zuzuordnen [MRS08]. Die Dokumente sind Teil eines Dokumentenraumes $\mathbb{X}$ und die Klassen Teil einer vorab definierten Menge $\mathbb{C} = \{c_1, c_2, ..., c_n\}$. Daraus folgt $d \in \mathbb{X}$ und $c \in \mathbb{C}$. Als Trainingsdatensatz $\mathbb{D}$ dient ein Satz manuell annotierter Dokumente $\langle d, c \rangle$, wobei gilt $\langle d, c \rangle \in \mathbb{X} \times \mathbb{C}$.

Durch einen Lernalgorithmus soll anschließend eine Klassifikationsfunktion oder kurz Klassifizierer γ entstehen, welcher Dokumente einer Klasse zuordnet $\gamma : \mathbb{X} \to \mathbb{C}$. Dieses Verfahren ist allgemein als überwachtes Lernen bzw. Supervised Learning bekannt [MRS08, CL16]. Der Mensch fungiert, durch seine manuelle Annotation der Trainingsdaten, als Aufseher oder Supervisor des Lernprozesses. Die Lernmethode wird als Γ bezeichnet und erhält die Trainingsdaten aus $\mathbb{D}$. Als Resultat erhält man die Klassifikationsfunktion γ, sodass gilt $\Gamma(\mathbb{D}) = \gamma$.

3.4.2 Eingrenzung der Klassifikationsverfahren

Zur Textklassifikation gibt es diverse Verfahren und Algorithmen, welche teils auf verschiedene Grundideen basieren oder für spezielle Anwendungszwecke entworfen wurden. Da im Rahmen dieser Arbeit nicht alle behandelt werden können, soll vorab eine Eingrenzung stattfinden. Dabei wird versucht, möglichst unterschiedliche Klassifikationsverfahren auszuwählen, um deren Leistungsfähigkeit für den gegebenen Anwendungsfall zu ermitteln.

Instanzen- und modellbasierte Verfahren

Grundsätzlich lassen sich Klassifikationsverfahren in zwei Gruppen einteilen [CL16]. Instanzenbasierte Verfahren nutzen direkt den Trainingsdatensatz, um neue Daten zu klassifizieren. Zur Klassifikation wird das ähnlichste Objekt aus der Trainingsmenge gesucht. Dem gegenüber stehen modellbasierte Verfahren, welche anhand des Trainingsdatensatzes ein Modell erstellen und nur dieses zur Klassifikation benötigen.

Lineare und nicht-lineare Klassifizierer

Des Weiteren werden Klassifikationsverfahren in lineare und nicht-lineare Klassifizierer eingeordnet [MRS08]. Auf ein zweidimensionales Problem beschränkt bedeutet das eine Klassentrennung in Form einer Geraden bzw. Hyperebene oder in Segmenten bzw. komplexeren Funktionen. Für das vorliegende Problem sollen Verfahren aus beiden Bereichen untersucht werden.

Untersuchte Verfahren

Der Naive-Bayes-Klassifikator stellt ein einfaches lineares und modellbasiertes Verfahren dar. Trotz seiner naiven Annahmen der Unabhängigkeit der Attribute [CL16], findet er Anwendung in einer Vielzahl von Problemen. Das Verfahren ist einfach zu implementieren, zudem ist die Anzahl der Parameter gering, was die Parameterauswahl minimiert [WKQ+08]. Der Naive-Bayes-Klassifikator soll deshalb als erstes Verfahren untersucht werden.

K-Nearest-Neighbour (KNN) unterscheidet sich von den meisten anderen Klassifikatoren, da die Klassifizierung anhand aller Trainingsdaten und nicht anhand eines Modells stattfindet [MRS08]. Dadurch ist kein aufwändiges Training notwendig, was KNN zu einem instanzenbasierten Klassifikator macht. Des Weiteren kann dieser Klassifikator mit nicht-linear trennbaren Daten umgehen, weshalb dieser im Anschluss an den Naive-Bayes-Klassifikator untersucht wird.

Support Vector Machine (SVM) ist ein modellbasiertes Klassifikationsverfahren und zählt zu einem der robustesten und genauesten Algorithmen im Data Mining [WKQ+08]. Es existieren diverse Erweiterungen und Modifikationen, u.a. zur Klassifikation nicht-linear separierbarer Daten, was die Untersuchung dieses Verfahrens erstrebenswert macht. Tabelle 3.1 zeigt die Eigenschaften der drei untersuchten Verfahren im Vergleich.

Klassifikator	Linear	Nicht-Linear	Instanzenbasiert	Modellbasiert
Naive Bayes	X			X
K-Nearest-Neighbour		X	X	
Support Vector Machine	X	X*		X

Tabelle 3.1: Verfahren zur Textklassifikation
* Erweiterungen für nicht-lineare Probleme vorhanden

3.4.3 Naive-Bayes-Klassifikator

Der Naive-Bayes-Klassifikator ist ein wahrscheinlichkeitsbasierter Klassifikator. Die Zuordnung eines Dokuments d zu einer Klasse c geschieht auf Grundlage der Wahrscheinlichkeit, welche aus den Trainingsdaten abgeleitet wird. Dabei wird die Annahme getroffen, dass alle Attribute, im vorliegenden Fall die Merkmale einer Klasse, voneinander unabhängig sind [CL16, IMF13]. Diese Annahme ist Grund für die Bezeichnung *naive*. Der Naive-Bayes-Klassifikator basiert auf Bayes' Theorem, auch Bayessche Formel genannt. Sie berechnet die Wahrscheinlichkeit eines Ereignisses unter einer bestimmten

Bedingung, im Falle der Text-Klassifikation das Auftreten eines Dokuments d in einer Klasse c:

$$P(c|d) = \frac{P(d|c) \cdot P(c)}{P(d)} \tag{3.7}$$

Da das Ziel der Textklassifikation ist, die beste Klasse, also die Klasse mit der höchsten Wahrscheinlichkeit zu finden [MRS08], wird die höchst-wahrscheinlichste oder *maximum a posteriori (map)* Klasse c_{map} wie folgt berechnet:

$$c_{map} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c) \cdot P(c)}{P(d)} \tag{3.8}$$

Da der Nenner denselben Wert für alle Klassen und somit keinen Einfluss auf das Maximum hat, kann er weggelassen werden. Das Dokument d wird in der Regel als eine Menge von Features $x_1, x_2, ..., x_n$ dargestellt (vgl. Gleichung 3.9).

$$c_{map} = \underset{c \in C}{\operatorname{argmax}} P(d|c) \cdot P(c) = \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, ..., x_n|c) \cdot P(c) \tag{3.9}$$

Zur Auswahl der Merkmale wird vor dem Training des Klassifizierers eine Merkmalsextraktion durchgeführt (vgl. Kap. 3.1.2) [MRS08].

Multinomiales Modell

Der Naive-Bayes-Klassifikator kann auf unterschiedliche Arten implementiert werden. Im multinomialen Modell stellen Merkmalsvektoren eine Menge von Termen $(\langle t_1, t_2, ..., t_{n_d} \rangle | c)$ gemäß ihres Vorkommens im Dokument d dar.

$$P(d|c) = P(\langle t_1, t_2, ..., t_{n_d} \rangle) \tag{3.10}$$

Dokumente werden als eine Menge von Termen repräsentiert, wobei n die Größe des Vokabulars repräsentiert. Folglich können häufig auftretende Terme, je nach gewähltem Gewichtungs-Verfahren, Einfluss auf die Klassifizierung haben (vgl. Kap. 3.1.2).

Bernoulli-Modell

Bei Anwendung des Bernoulli-Modells bleibt die Anzahl der auftretenden Terme gänzlich unbeachtet. Lediglich die Anzahl der Dokumente, welche den gesuchten Term enthalten, ist von Interesse. Gleichung 3.11 zeigt die Berechnung.

$$P(d|c) = P(\langle e_1, e_2, ..., e_M \rangle | c) \tag{3.11}$$

Der binäre Vektor $\langle e_1, e_2, ..., e_M \rangle$ der Dimension M zeigt für jeden Term an, ob dieser im Dokument d enthalten ist (1) oder nicht (0). Die Repräsentation der Dokumente beschränkt sich demnach auf die Menge $\{0,1\}^M$.

Das Bernoulli-Modell ignoriert mehrfaches Auftreten eines Terms in einem Dokument, was bei langen Dokumenten zu fehleranfälligem Verhalten führen kann [MRS08]. Das Vorhandensein eines einzigen Terms kann zur Klassifizierung genügen und häufiger auftretende Terme ihrer Bedeutung entziehen.

Laplace-Glättung
Ein Problem bei der Berechnung von c_{map} (vgl. Gleichung 3.9) ist, dass ein einzelnes Dokument in einer Klasse die Vorhersage stark beeinflussen kann [MRS08]. Tritt ein Term t_1 nur in einer Klasse auf, ist die bedingte Wahrscheinlichkeit dieses Terms für alle anderen Klassen 0, also $P(t_1|c) = 0$. Um diesem Verhalten entgegenzuwirken kann die sogenannte *Laplace-Glättung* eingesetzt werden. Hierzu wird 1 zu jedem Term addiert, um eine Wahrscheinlichkeit von 0 zu verhindern und einzelnen Termen weniger Gewichtung zu geben.

3.4.4 K-Nearest-Neighbour

K-Nearest-Neighbour ist ein instanzenbasiertes Verfahren und benötigt keinen expliziten Trainingsschritt [ZM16]. Lediglich die Erzeugung eines Index wird vorausgesetzt. Die eigentliche Klassifizierung geschieht anschließend durch die Berechnung des Ähnlichkeitsmaßes (vgl. Kap. 3.3.2). So wird ein Dokument, repräsentiert durch einen Merkmalsvektor, derjenigen Klasse zugeordnet, deren Dokument den ähnlichsten Merkmalsvektor besitzt [Run12]. Veranschaulicht wird dies in Abbildung 3.3, wobei dem gesuchten Objekt, dargestellt durch den schwarzen Punkt, die negative Klasse zugewiesen wird.

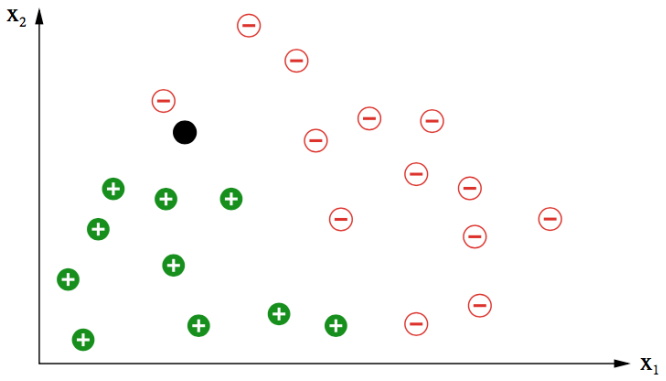

Abbildung 3.3: Nearest-Neighbour-Classifier [Ert09]

Dabei ist es auch möglich, die Ähnlichkeit zu mehreren Objekten (k) zu suchen, wobei gilt $k \in \{2,...,n\}$. Formal wird dabei die Distanz zweier Vektoren gemessen (vgl. 3.12).

$$||x - x_k|| = \min_{j=1,...,n} ||x - x_j|| \tag{3.12}$$

Der Vorteil dieses Verfahrens liegt darin, dass keine aufwändige Trainingsphase stattfinden muss und neue Daten durch einfache Erweiterung des Trainingsdatensatzes hinzugefügt werden können. Der Nachteil liegt im hohen Rechenaufwand bei der Klassifikation, da ein Vergleich mit jedem Merkmalsvektor durchgeführt werden muss [Run12].

Der Ablauf des KNN-Klassifikators gestaltet sich wie folgt:

1. Index der Trainingsdokumente bilden
2. Klassifikation
   a) Vergleiche gesuchtes Dokument d mit allen Dokumenten im Index
   b) Wähle die k-obersten Dokumente
   c) Weise d die häufigste Klasse der k-obersten Dokumente zu

3.4.5 Support Vector Machines

Eine Support Vector Machine ist ein vektorbasiertes Machine-Learning-Verfahren. Dabei werden die Trainingsdaten als eine Menge von Vektoren im n-dimensionalen Raum abgebildet, wobei n der Anzahl der Terme im Vokabular entspricht (vgl. Kap. 3.3). Jedes Objekt in der Menge der Trainingsdaten bildet dabei ein Paar aus Datenpunkt $\vec{x}_i$ und seiner Klassenzugehörigkeit y_i. Zur Veranschaulichung soll zunächst nur von zwei Klassen ausgegangen werden.

Während des Trainings wird versucht, eine Klassengrenze mit maximalem Abstand zu den Punkten der beiden Klassen zu finden. Es entsteht somit ein maximaler Bereich, *Margin*, zwischen den nächstgelegenen Punkten der beiden Klassen [MRS08, CL16]. Die Klassengrenze bildet eine Hyperebene und alle nächstgelegenen Punkte Stützvektoren (engl. Support Vectors) im euklidischen Raum (vgl. Abb. 3.4).

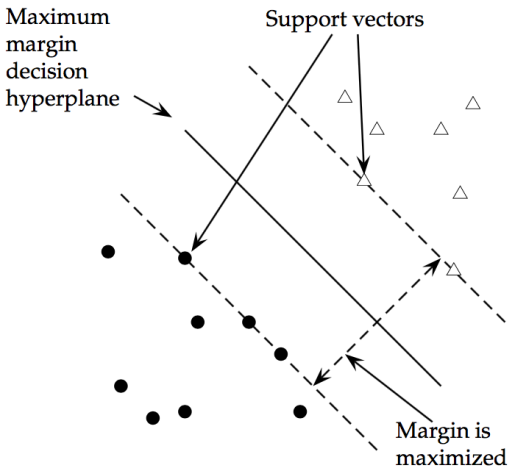

Abbildung 3.4: Support Vector Machine [MRS08]

Die Hyperebene wird durch den Bias b und den Normalenvektor $\vec{w}$ bestimmt [MRS08]. Die optimale Hyperebene wird, bei linear separierbaren Daten, durch $\vec{w}^T\vec{x}+b=0$ beschrieben [CV95]. Ist die Hyperebene definiert, lässt sich die Klassenzugehörigkeit eines Objekt anhand seiner Lage relativ zur Hyperebene bestimmen. Die Klassen werden bei SVM üblicherweise durch +1 bzw. -1 dargestellt. Zur Klassifikation der Objekte genügt es zu berechnen, ob diese größer oder kleiner als Null sind (vgl. 3.13).

$$Klasse(\vec{x}) = \begin{cases} 1, & falls \quad w^T \cdot x + b > 0 \\ -1, & falls \quad w^T \cdot x + b < 0 \end{cases} \tag{3.13}$$

Zur Bestimmung der Stützvektoren wird die kürzeste Distanz zwischen einem Vektor $\vec{x}$ und der Hyperebene gesucht. Da die kürzeste Distanz zwischen einer Hyperebene und einem Vektor senkrecht auf der Hyperebene steht, muss sie parallel zu $\vec{w}$ verlaufen. Der Grenzbereich ($Margin\,\rho$) zwischen den Klassen entspricht $\rho = 2/|\vec{w}|$ [MRS08]. Dieser Bereich soll maximal sein (vgl. Abb. 3.5).

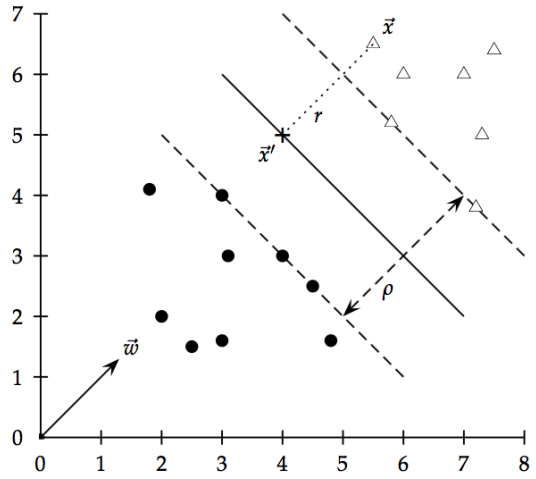

Abbildung 3.5: Support Vector Machine - Margin [MRS08]

Multiclass SVM

Die Klassifikation mit Support Vector Machines beschreibt ein Zweiklassenproblem. Um dieses Verfahren auf ein Mehrklassenproblem anzuwenden, stehen zwei gängige Vorgehen zur Auswahl. Beim *eine-gegen-alle-anderen*-Verfahren, auch *One-vs-Rest* (OvR), werden zunächst k Klassifikatoren trainiert und jede Klasse k von *k-1* verbleibenden Klassen unterschieden [Nie03]. Der erste Klassifikator unterscheidet also die Klasse c_1 von den Klassen $\{c_2, ..., c_k\}$.

Ein anderes Verfahren, *eine-gegen-eine*, auch *One-vs-One* (OvO), unterscheidet Paare von Klassen, d.h. $k(k-1)/2$ Klassifikatoren. Es entstehen somit mehrere binäre Entscheidungen. Um eine endgültige Klasse zu bestimmen, wird die Klasse, welche von den meisten Klassifikatoren ausgewählt wurde, übernommen [Nie03]. Aufgrund der kleineren Stichprobenumfänge ist der Rechenaufwand bei diesem Verfahren, gegenüber dem *eine-gegen-alle-anderen*-Verfahren, geringer.

Linear separierbare Daten

In der vorigen Erläuterung wurde davon ausgegangen, dass die Daten linear separierbar sind, sich also mit einer Hyperebene exakt trennen lassen (vgl. Abb. 3.4). Ist dies der Fall, lässt sich das beschriebene Verfahren ohne Weiteres anwenden. In der Realität finden sich jedoch häufig Datensätze, welche sich nicht exakt trennen lassen.

Nicht-linear separierbare Daten

Die Aufteilung der Objekte im Raum kann sehr unterschiedlich verlaufen und oft lassen sich Objekte derselben Klassen nicht mittels linearer Funktionen trennen. Um

dennoch das Prinzip der SVM anwenden zu können, bedient man sich dem sogenannten *kernel-trick* [BGV92]. Dabei werden Daten in eine höhere Dimension projiziert und in dieser Dimension wiederum linear klassifiziert.

Soft-Margin-Klassifikation
Eine weitere Möglichkeit, nicht-linear-separierbare Daten in der SVM zu klassifizieren, ist die Soft-Margin-Klassifikation. Sie erlaubt es einigen Datenpunkten außerhalb der Klassengrenze zu liegen, also im eigentlich falschen Bereich. Dadurch lassen sich Datensätze mit einigen Datenpunkten, welche vom Rest der Klasse stark abweichen, linear separieren.

Optimierung durch stochastischen Gradientenabstieg
Optimierung beschreibt im Machine Learning die Minimierung oder Maximierung einer Funktion *f(x)* durch Änderung von *x* [GBC16]. Die durch Minimierung entstehende Funktion wird als *Kostenfunktion* oder *Straffunktion* bezeichnet. Ziel ist es, die Kosten bzw. die Strafe zu minimieren. Dazu ist es notwendig, das globale Minimum einer Funktion zu ermitteln.

Eine Möglichkeit bietet das *Gradientenabstiegsverfahren*. Zuerst wird der Gradient der Funktion bestimmt. Anschließend wird die entgegengesetzte Richtung genutzt, um das Minimum zu finden. Dieser Schritt wird für alle Daten im Trainingsdatensatz wiederholt und anschließend summiert. Problematisch ist dieses Verfahren bei großen Trainingsdatensätzen, da das Summieren sehr viel Zeit in Anspruch nimmt.

Ein verbessertes Verfahren ist das *stochastische Gradientenabstiegsverfahren* (SGD). Hierzu wird lediglich ein Teil der Trainingsdaten genutzt, um das Minimum zu bestimmen. In jedem Trainingsschritt wird zufällig ein Beispiel aus den Trainingsdaten gewählt und das Minimum bestimmt. Das Verfahren ist damit weniger aufwändig, jedoch auch weniger genau, als das Gradientenabstiegsverfahren mit Hilfe des kompletten Trainingsdatensatzes.

3.5 Bewertung der Verfahren

Um die Leistungsfähigkeit eines Klassifikators zu überprüfen, wird die Funktionalität auf unbekannte Daten getestet. Dazu kann eine Teilmenge des verfügbaren Gesamt-Datensatzes herangezogen werden, welche nicht zum Training verwendet wird. Zeigt

der Klassifikator auf dieser Teilmenge gute Ergebnisse, wird die Annahme getroffen, dass diese auch allgemeine Gültigkeit besitzen [Fer03]. Man spricht dabei auch von einer Generalisierung. Ziel ist es, ein Modell zu erhalten, welches möglichst genau generalisiert [MG16, GBC16].

Bereits vor dem Training wird zu diesem Zweck der Gesamt-Datensatz in Trainings- und Testmengen aufgeteilt. Der Begriff Mengen ist dabei im mathematischen Sinn nicht immer korrekt, da in Mengen keine Elemente mehrfach auftreten dürften. Bei Text- und Dokumenten-Klassifikationen, kann dies jedoch nicht gewährleistet werden. Trotzdem handelt es sich um eine übliche Nomenklatur, weshalb diese auch nachfolgend verwendet wird.

Eine Möglichkeit zur Trennung, ist die Aufteilung in feste Anteile, beispielsweise 80% für das Training und 20% zum Testen. Der Nachteil dieses Verfahrens ist, dass immer dieselben Daten verwendet werden und die Generalisierung des Verfahrens nicht ausreichend getestet wird (vgl. Kap. 3.1.2) [Ras15].

3.5.1 Overfitting und Underfitting

Overfitting tritt auf, wenn ein Modell zu nah an Eigenschaften der Trainingsdaten angepasst wird und somit keine ausreichende Generalisierung stattfindet [Ras15, MG16]. Man spricht dabei auch von einer hohen Varianz des Modells. Als Folge wird eine gute Vorhersage auf Trainingsdaten, jedoch nicht auf ungesehene Daten, getroffen. Grund dafür kann ein zu komplexes Modell für den gegebenen Trainingsdatensatz sein oder zu wenige Trainingsdaten.

Dem gegenüber steht das *Underfitting*, welches auftritt wenn weder auf den Trainingsdaten, noch auf ungesehene Daten eine gute Vorhersage getroffen wird. Das Modell hat einen hohen Bias, auch Verzerrung genannt. Ein zu einfach gewähltes Modell kann ein Grund dafür sein [MG16].

3.5.2 K-fache Kreuzvalidierung

Die k-fache Kreuzvalidierung ist ein Verfahren, um das passende Modell für den gegebenen Datensatz zu finden und somit Over- und Underfitting vorzubeugen [Ras15]. Dabei wird die Gesamt-Datenmenge in k Teilmengen unterteilt, wobei $k\text{-}1$ Teilmengen für das Training und der restliche Teil für das Testen verwendet werden. Die Aufteilung wird

k-mal wiederholt (vgl. Abb. 3.6). Die Leistung eines Modells wird anhand des Durchschnitts aller Iterationen berechnet.

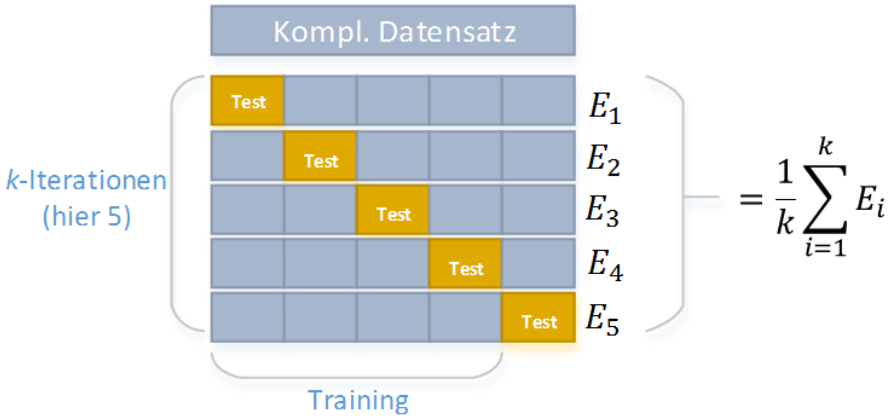

Abbildung 3.6: k-fache Kreuzvalidierung

Dadurch lässt sich ein Leistungswert bestimmen, ohne beim Training auf einen Teil des Datensatzes verzichten zu müssen.

3.5.3 Kennwerte

Zur Darstellung der Vorhersagen eines Klassifikators werden Konfusionsmatrizen, auch Wahrheitsmatrizen genannt, eingesetzt. Dabei wird die Anzahl der richtig positiven, richtig negativen bzw. falsch positiven, falsch negativen Vorhersagen aufgezeigt.

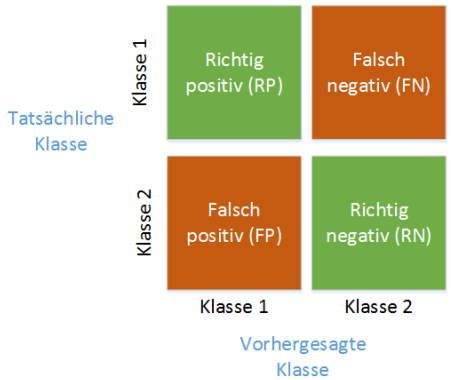

Abbildung 3.7: Konfusionsmatrix

Mit Hilfe dieser Werte lassen sich weitere, wie etwa die *Korrektklassifizierungsrate* (accuracy score), *Trefferquote* (recall) und *Genauigkeit* (precision) berechnen [Ras15].

$$Korrektklassifizierungsrate = \frac{RP + RN}{RP + RN + FP + FN} \tag{3.14}$$

Genauigkeit (vgl. 3.15) und Trefferquote (vgl. 3.16) geben die Rate für richtig-positive bzw. richtig-negative Vorhersagen an.

$$Genauigkeit = \frac{RP}{RP + FP} \tag{3.15}$$

$$Trefferquote = \frac{RP}{RP + FN} \tag{3.16}$$

Aus der Kombination von Genauigkeit und Trefferquote lässt sich das *F-Maß* (vgl. 3.17), auch F_1 genannt, berechnen.

$$F_1 = 2\,\frac{Genauigkeit \times Trefferquote}{Genauigkeit + Trefferquote} \tag{3.17}$$

3.5.4 Rastersuche

Die meisten Klassifizierungsalgorithmen bieten eine Vielzahl manuell einstellbarer Parameter, den sogenannten *Hyperparametern*. Zur Ermittlung der optimalen Werte dieser Parameter werden verschiedene Kombinationen nacheinander getestet.

Rastersuche (engl. Grid Search) bietet eine Möglichkeit zur Ermittlung der optimalen Wertkombinationen der Hyperparameter [Ras15]. Dabei wird eine Liste mit verschiedenen Werten für die einzelnen Hyperparameter erstellt und ein Training des Modells mit allen möglichen Kombinationen durchgeführt.

Im nächsten Kapitel werden die genannten Bewertungsverfahren genutzt, um eine Modellauswahl, inklusive einer optimalen Abstimmung der Parameter, zu treffen.

4 Modellbildung

In diesem Kapitel werden die Verfahren zur Datenvorverarbeitung und Klassifikation aus dem vorherigen Kapitel genutzt, um ein Modell für die gegebene Problemstellung zu erstellen. Die Leistungsfähigkeiten jedes Klassifikationsverfahren werden abschließend durch unterschiedliche Kennwerte validiert, um vergleichbare Ergebnisse zur Auswahl des optimalen Klassifizierers zu erhalten.

4.1 Eingesetzte Technologien

Die Modellbildung und Umsetzung des Prototypen wird in der Programmiersprache *Python* unter Verwendung verschiedener Programmbibliotheken durchgeführt. Nachfolgend werden die einzelnen Komponenten näher erläutert.

4.1.1 Python

Aufgrund zahlreicher Programmbibliotheken und Erweiterungen hat sich die Programmiersprache *Python* in den letzten Jahren zu einer der beliebtesten Sprachen für Datenanalyse und Machine Learning entwickelt. Auch wenn Python ursprünglich nicht eigens dafür entworfen wurde, gilt es, bedingt durch sein großes Ökosystem an Drittanbieter-Paketen, als eine der Standard-Programmiersprachen für Datenanalyse und Machine Learning [Van16].

4.1.2 NumPy

Numerical Python (*NumPy*[1]) ist eine quelloffene Programmbibliothek für wissenschaftliche Datenverarbeitung. Mit ihrer Hilfe lassen sich unter anderem Berechnungen auf mehrdimensionale Arrays einfacher ausführen [Idr13]. Dabei kommen NumPy Arrays zum Einsatz, welche ein spezialisiertes Objekt mit mathematischen Methoden in Python darstellen. Diese sind gerade bei numerischen Operationen effizienter einsetzbar, als die herkömmlichen Python Listen [Idr14].

[1] http://www.numpy.org/

4.1.3 Pandas

Pandas[2] ist eine quelloffene Datenanalyse-Programmbibliothek für Python. Sie bietet Werkzeuge für den einfachen Umgang mit Datenstrukturen und unterstützt den Entwickler bei der Bearbeitung dieser durch eine Abstraktionsschicht [Hey15]. Gerade *DataFrame*-Objekte, welche durch die Programmiersprache *R* inspiriert sind, helfen Entwicklern bei gängigen Datenanalyse-Aufgaben wie Indizieren, Abrufen, Kombinieren und Umformen von ein- oder mehrdimensionalen Daten.

4.1.4 Scikit-learn

Scikit-learn[3] ist eine Python-Programmbibliothek für Machine Learning und bietet eine Vielzahl an implementierten Algorithmen, unter anderem zur Klassifikation [PVG+11]. Darüber hinaus ist sie für die Zusammenarbeit mit anderen Bibliotheken, wie Pandas oder NumPy optimiert [MG16].

4.2 Datenanalyse

Die in Kapitel 3.4 eingeführten Verfahren zur Textklassifikation benötigten einen Trainingsdatensatz, um daraus ein *Modell* zu erstellen oder diese, wie bei K-Nearest-Neighbour, direkt zur Klassifizierung zu nutzen. Modellbasierte Machine-Learning-Algorithmen lernen ein Modell anhand der Trainingsdaten und erlauben Vorhersagen auf ungesehene Daten (vgl. Abb. 4.1). Das in diesem Kapitel erstellte Modell wird im späteren Prototypen genutzt, um die Vorhersage der entsprechenden Kategorie durchzuführen.

[2] https://pandas.pydata.org
[3] http://scikit-learn.org

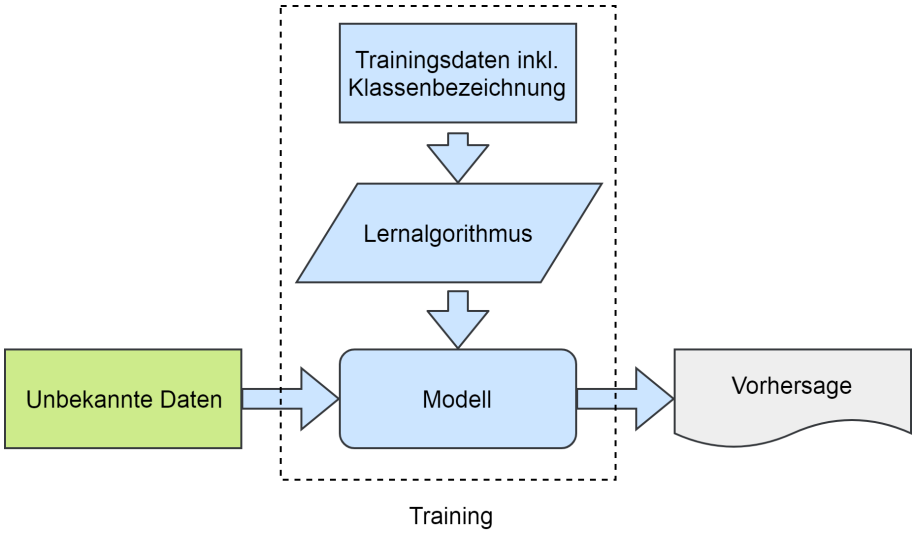

Abbildung 4.1: Training und Vorhersage des Modells

Die Trainingsdaten werden dem Lernalgorithmus des entsprechenden Klassifikators übergeben, um damit ein Modell zu erstellen. Der Trainingsdatensatz enthält manuell annotierte Daten, in diesem Fall Transaktionen (vgl. Tab. 4.1).

Zahlungsempfänger	**Verwendungszweck**	**Buchungstyp**	***Klasse***
MUSTER LEBENSVERSICHERUNG	...	Lastschrift	*Versicherung*
GA NR00001234	...	Auszahlung	*Barentnahme*

Tabelle 4.1: Transaktionen inkl. Klassenbezeichnung

Trainingsdatensatz

Als Trainingsdatensatz dient eine Sammlung von 1063 Transaktionen. Darunter befinden sich anonymisierte Transaktionen aus drei verschiedenen Bankkonten, sowie generierte Transaktionen aus dem Mock-Up-System von adorsys. Zur Textklassifikation werden lediglich die Felder „Zahlungsempfänger“, „Verwendungszweck“ und „Buchungstyp“ verwendet. Diese Felder enthalten textuelle Informationen und lassen durch Firmen- oder Produktbezeichnungen Rückschlüsse auf die Kategorie zu. Des Weiteren lässt sich durch weglassen der restlichen Felder, der Umfang des Vokabulars und damit die Anzahl der Dimensionen reduzieren [MRS08].

Die restlichen Felder einer Transaktion werden erst nach der Kategorisierung durch eine PFM-Anwendung, etwa zur Berechnung aller Ausgaben einer Kategorie, benötigt. Die Gläubiger-ID wird separat überprüft (vgl. Kap. 2.2.2). Zu jeder Instanz der Trainingsdaten existiert ein Label, in unserem Fall der Name der jeweiligen Kategorie. Die Summe der Kategorien wird in dieser Arbeit als Ziele oder Targets bezeichnet (vgl. Abb. 4.2).

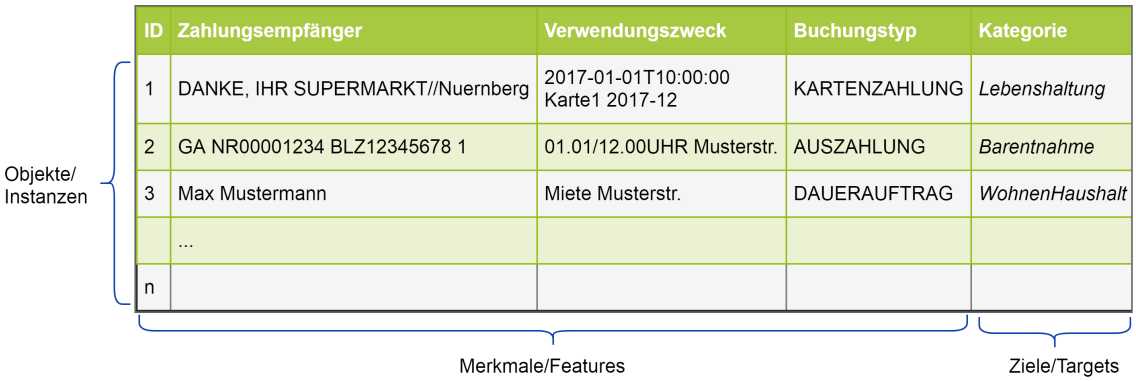

Abbildung 4.2: Merkmale und Ziele der Daten

Die Felder Zahlungsempfänger, Verwendungszweck und Buchungstyp werden zur einfachen Merkmalsextraktion in eine Zeichenkette zusammengefasst.

4.2.1 Datenvorverarbeitung der Trainingsdaten

Die Qualität der Daten ist bei Machine-Learning-Verfahren von großer Bedeutung [CL16]. Nur bei optimaler Vorverarbeitung kann der Lernalgorithmus ein effizientes Modell erstellen. Nachfolgend werden die Vorverarbeitungsschritte der Trainingsdaten beschrieben, welche ebenfalls im späteren Prototypen Anwendung finden.

Tokenisierung

Zur Ermittlung eines passenden Tokenisierers werden drei gängige Tokenisierer, *TreebankWordTokenizer*, *WordPunctTokenizer* und *WhitespaceTokenizer*, aus dem NLTK verglichen [Per14]. Diese werden auf einzelne Transaktionen des Trainingsdatensatzes angewendet, um eine Beurteilung zuzulassen. Zur Veranschaulichung wird hier „Instanz 1“ (vgl. Tab. 4.2) aus dem Trainingsdatensatz genutzt. Diese enthält im Feld des Zahlungsempfängers die Zeichenkette „*DANKE, IHR* ***SUPERMARKT****//Nuernberg*“.

Zahlungsempfänger	**Verwendungszweck**	**Buchungstyp**
DANKE, IHR SUPERMARKT//-Nuernberg	2017-01-01T10:00:00 Karte1 2017-12	Kartenzahlung

Tabelle 4.2: Beispiel-Transaktionen

Für die Kategorisierung relevant ist vor allem der Firmenname (hier *SUPERMARKT*), da er Rückschlüsse auf die Branche und somit oft auch auf die Kategorie zulässt. Des Weiteren kann davon ausgegangen werden, dass Firmennamen weitestgehend konstant bleiben, wohingegen Zusätze wie „*DANKE, IHR*“ variabel sind und für die Kategorisierung keinen Mehrwert bringen.

Der Firmenname ist in diesem Fall nicht mit Leerzeichen vom restlichen Text separiert, sondern schließt mit Schrägstrichen ab. Eine Tokenisierung nur anhand der Leerzeichen ist somit nicht möglich. Die drei Tokenisierer aus dem NLTK gehen mit dieser Situation unterschiedlich um, wie in Tabelle 4.3 gezeigt.

TreebankWordTokenizer	DANKE	,	IHR	SUPERMARKT//Nuernberg		
WordPunctTokenizer	DANKE	,	IHR	SUPERMARKT	//	Nuernberg
WhitespaceTokenizer	DANKE,	IHR	SUPERMARKT//Nuernberg			

Tabelle 4.3: NLTK-Tokenisierer im Vergleich Variante 1

TreebankWordTokenizer und WhitespaceTokenizer können in diesem Fall den Firmennamen nicht separieren, wohingegen WordPunctTokenizer diesen als eigenes Token erkennt. Er liefert somit das gewünschte Resultat. Wendet man die Tokenisierung jedoch auch auf den Verwendungszweck aus 4.2 an, werden Angaben wie Datums- und Zeitstempel ebenfalls getrennt, was deren Auswertung erschwert.

TreebankWordTokenizer	2017-01-01T10:00:00										
WordPunctTokenizer	2017	-	01	-	01	T	10	:	00	:	00
WhitespaceTokenizer	2017-01-01T10:00:00										

Tabelle 4.4: NLTK-Tokenisierer Vergleich mit Zeitstempel

Einige Sonderzeichen sollen deshalb nicht als Tokens dargestellt werden, da die Daten sonst aus ihrem Kontext gerissen werden, wie beim Zeitstempel ersichtlich wird. Die Tokenisierung wird aus diesem Grund um einen Vorverarbeitungsschritt ergänzt. Dieser umfasst die Entfernung bestimmter Sonderzeichen und die Kleinschreibung aller Wörter.

Zur Entfernung unerwünschter Zeichen wird ein Regulärer Ausdruck eingesetzt. Dieser entfernt die Zeichen „+“ und „\“ bei einmaligen oder mehrfachen Vorkommen sowie mehr als drei aufeinanderfolgende Leerzeichen.

Stemming und Lemmatisierung

Das NLTK bietet verschiedene Implementierung für Stemming und Lemmatisierung an. Da diese sprachabhängig sind (vgl. Kap. 3.2.4) werden Stemmer bzw. Lemmatisierer für die deutsche Sprache gewählt. Der *SnowballStemmer* bietet als einziger Stemmer im NLTK Unterstützung für die deutsche Sprache an und kommt deshalb bei der Merkmalsextraktion zum Einsatz [Per14]. Des Weiteren bietet dieser eine Liste mit deutschen Stoppwörtern, welche ebenfalls genutzt wird.

Zur Lemmatisierung deutscher Wörter bietet das NLTK keine Unterstützung an. Es existieren jedoch andere Natural-Language-Tools, etwa spaCy[4], welche Lemmatisierung für deutsche Wörter durchführen. Tests auf Teile des Trainingsdatensatzes zeigen jedoch, dass dadurch teils ungewünschte Veränderungen, vor allem an Eigennamen, stattfinden. Artikel oder Pronomen, welche auch bei Firmennamen Verwendung finden, werden abgewandelt und können die Wiedererkennung der Transaktionen erschweren (vgl. Tab. 4.5).

Original	**spaCy Lemmatisierung**
Ihr Platz	*Mein Platz*
Die Heilsarmee	*Der Heilsarmee*

Tabelle 4.5: SpaCy Lemmatisierung von Firmen- und Organisationsnamen

Gerade Firmennamen bilden ein wichtiges Indiz zur Identifikation der Kategorie (vgl. Kap. 2.1.2) und müssen in möglichst unveränderter Form vorliegen. Im weiteren Verlauf kommt deshalb lediglich ein Stemming und keine Lemmatisierung zum Einsatz.

Ablauf der Datenvorverarbeitung
Aus den vorab beschriebenen Erkenntnissen, ergibt sich folgender Ablauf zur Datenvorverarbeitung der Trainingsdaten und späteren Suchanfragen im Prototypen.

1. Ungewünschte Zeichen entfernen
2. Kleinschreibung
3. Tokenisierung
4. Stemming

4.2.2 Merkmalsextraktion der Trainingsdaten

Zur Merkmalsextraktion werden die Verfahren Bag-of-words und TF-IDF verglichen. Zusätzlich wird eine unterschiedliche Parametrisierung beider Verfahren getestet, um für den vorliegenden Datensatz eine optimale Extraktion der Merkmale zu ermitteln.

Die Auswahl eines Verfahrens wird zusammen mit der Auswahl eines Klassifizierungsverfahrens getroffen. Hierzu werden die Klassifizierer jeweils mit Bag-of-words und TF-IDF validiert.

[4] https://spacy.io

Einlesen der Trainingsdaten

Die Trainingsdaten dieser Arbeit wurden vorab in eine CSV-Datei zusammengefasst. Zur Merkmalsextraktion werden sie zunächst in einen Pandas-DataFrame eingelesen (vgl. Listing 4.1). Anschließend werden die Werte für Buchungstext, Verwendungszweck und Zahlungsempfänger zusammengefasst und im selben Schritt die Vorverarbeitung durchgeführt. Die Variable *targets* speichert die Kategorie und *word_counts* die Merkmalsvektoren gemäß dem gewählten Verfahren. Die Auswahl des Verfahrens geschieht über die Klassenvariable *self.vectorizer*.

```
disturb_chars = '([\/+]|\s{3,})'
df = self.file_handler.read_csv('Labeled_bookings.csv')
df['values'] = df[['bookingtext', 'usage', 'owner']].astype(str)\
                                                    .sum(axis=1)\
                                                    .replace(disturb_chars, ' ')\
                                                    .str.lower()
targets = df['category'].values
word_counts = self.vectorizer.fit_transform(df['values'].values.astype(str))\
                                                    .astype(float)
```

Listing 4.1: Merkmalsextraktion

4.3 Auswahl eines Klassifizierungsverfahren

Um ein geeignetes Klassifizierungsverfahren für den späteren Prototypen zu bestimmen, werden zuerst die Werte der Hyperparameter evaluiert. Anschließend folgt die Validierung der Verfahren mit den optimalen Hyperparametern. Mit Hilfe von Korrektklassifizierungsrate, Genauigkeit, Trefferquote und F1-Maß (vgl. Kap. 3.5.3) wird ein passendes Klassifizierungsverfahren für die vorliegende Problemstellung ausgewählt.

4.3.1 Parameterauswahl

Zur Ermittlung der optimalen Werte der Hyperparameter wird eine Rastersuche (vgl. Kap. 3.5.4) auf einen Teil der Trainingsdaten durchgeführt. Dieser Datensatz enthält insgesamt 420 Instanzen und gleich viele Transaktionen in jeder Kategorie, um Einflüsse durch Ungleichgewichte zu unterbinden [Ras15].

Die Wertebereiche werden zunächst grob bestimmt und mit Hilfe einer Validierungskurve eingegrenzt oder erweitert. Als Initialwerte dienen gängige Werte zur Textklassifikation, etwa aus Beispielen zur Klassifikation des *20-Newsgroups-Korpus*[5] [GPB17].

Im Falle der Support Vector Machine mit stochastischem Gradientenabstiegsverfahren wird etwa für den Regulierungsterm α ein Wertebereich von $10^{-7}, 10^{-6}, ..., 10^{-1}$ als Initialwerte genutzt. Die Validierungskurve (vgl. Abb. 4.3) zeigt ab Werten über 10^{-3} deutlich schlechtere Vorhersageergebnisse. Der optimale Wert wird deshalb in einem Bereich kleiner als 10^{-3} vermutet.

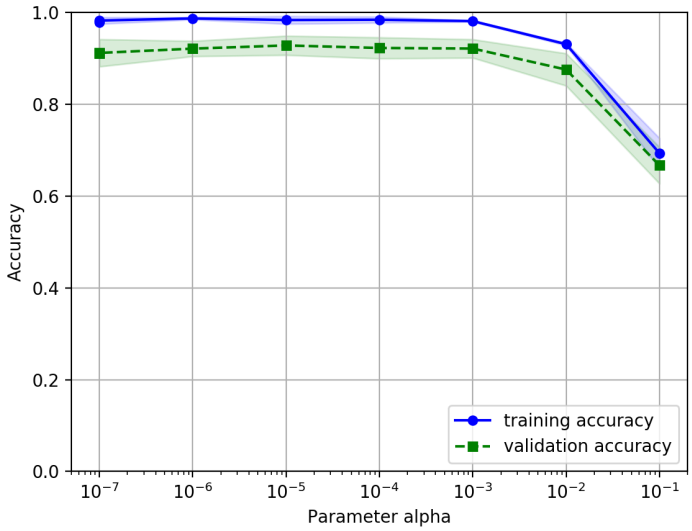

Abbildung 4.3: Validierungskurve für den Regulierungsterm der Support Vector Machine

Dieses Vorgehen wird für alle Klassifikationsverfahren in ähnlicher Form, entsprechend den jeweiligen verfügbaren Hyperparametern, durchgeführt. Nachfolgend werden die Parameter der einzelnen Verfahren erläutert.

Bag-of-words und TF-IDF

Zur Merkmalsextraktion werden die Klassen *CountVectorizer* für Bag-of-words bzw. *TfidfVectorizer* für TF-IDF aus Scikit-learn genutzt. Bei beiden Verfahren werden N-Gramme unterschiedlicher Länge untersucht. Bei TF-IDF wird außerdem eine Längen-Normalisierung (einfacher Durchschnitt: *L1*, Quadratischer Durchschnitt: *L2*) der Merkmale getestet.

[5] http://qwone.com/~jason/20Newsgroups/

Naive Bayes Klassifikator

Der Naive Bayes Klassifikator wird in zwei Varianten, Multinomial und Bernoulli, validiert. Dazu werden die Klassen *MultinomialNB* und *BernoulliNB* aus Scikit-Learn genutzt. In beiden Varianten werden unterschiedliche Werte für die Laplace-Glättung getestet.

K-Nearest-Neighbour

Für K-Nearest-Neighbour-Klassfikation wird die Implementierung *KNeighborsClassifier* aus Scikit-learn genutzt. Dabei wird die Anzahl nächster Nachbarn und deren Gewichtung validiert [sci17].

Support Vector Machine

SVM wird ebenfalls in zwei Implementierungen aus Scikit-learn validiert. Zum einen die klassische SVM-Implementierung *SVC*, welche auf der Open-Source-Bibliothek *libsvm* basiert [CL11]. Zum anderen eine Implementierung mit stochastischen Gradientenabstiegsverfahren, *SGDClassifier* genannt.

In beiden Varianten werden unterschiedliche Kernels zur Trennung der Daten, sowie unterschiedliche Werte für die Straffunktionen validiert. Außerdem werden jeweils One-vs-Rest und One-vs-One getestet. Bei der Implementierung *SGDClassifier* werden außerdem Soft-Margin-Klassifikatoren (*hinge*, *squared_hinge* und *modified_huber*) getestet.

Ablauf der Validierung

Zur Validierung der Hyperparameter wird eine Scikit-learn-Pipeline genutzt, um die Instanziierung der Objekte zur Merkmalsextraktion und Klassifikation zusammenzufassen. Listing 4.2 zeigt die Pipeline des MultinomialNB-Klassifikators zusammen mit einer TF-IDF Merkmalsextraktion. Die restlichen Verfahren folgen demselben Prinzip.

```
1  pipeline = Pipeline([
2      ('tfidf', TfidfVectorizer()),
3      ('clf', MultinomialNB())
4  ])
```

Listing 4.2: Scikit-learn-Pipeline zur Validierung

Die Parameter werden in Form eines Python-Dictionaries gespeichert und anschließend der Klasse *GridSearchCV* zur Rastersuche übergeben (vgl. Listing 4.3).

```
parameters = {
    'tfidf__ngram_range': [(1, 1), (1, 2), (1, 3)],
    'tfidf__norm': ('l1', 'l2'),
    'clf__alpha': (1, 0.1, 0.01, 0.001, 0.0001, 0.00001)
}

grid_search = GridSearchCV(estimator=pipeline, param_grid=parameters,
                           cv=15, scoring='accuracy')
```

Listing 4.3: Parameter für Multinomialen Naive Bayes Klassifikator

Die Klasse GridSearchCV führt gleichzeitig eine 15-fache Kreuzvalidierung aus (Parameter *cv*). Da es sich in dieser Arbeit um einen sehr kleinen Datensatz handelt, wird ein größerer Wert für *cv* als der Standardwert 3 gewählt, um die Anzahl der Trainingsdaten in jeder Iteration zu erhöhen und eine höhere Generalisierung zu erreichen [Ras15].

Zum Abschluss werden die Parameter, welche die höchste Korrektlassifizierungsrate erreicht haben, ausgegeben.

```
print("Best parameters: " + str(grid_search.best_params_))
print("Best score: %0.3f" % grid_search.best_score_)
```

Listing 4.4: Ausgabe der Rastersuche

Resultate der Rastersuche

Die Parameterauswahl der Merkmalsextraktion fällt sehr unterschiedlich aus. Bei den Varianten des Naive-Bayes-Klassifikators stellen sich Trigramme bzw. Tetragramme als optimal heraus, wohingegen bei den Implementierungen der Support Vector Machine Bigramm bzw. Tetragramme die besten Vorhersagen liefern (vgl. Tab. 4.6). K-Nearest-Classifier erzielt mit Unigrammen die besten Ergebnisse.

Klassifikator	**Bag-of-words**	**TF-IDF**	
	N-Gramme-Bereich (min, max)	*N-Gramme-Bereich (min, max)*	*Normalisierung*
MultinomialNB	(1, 3)	(1, 4)	L1
BernoulliNB	(1, 3)	(1, 3)	L1
K-Nearest	(1, 1)	(1, 1)	L1
SVC	(1, 2)	(1, 4)	L2
SGDClassifier	(1, 4)	(1, 4)	L1

Tabelle 4.6: Parameterauswahl der Merkmalsextraktion

Die Hyperparameter können aufgrund der unterschiedlichen Klassifikationsverfahren nicht eindeutig miteinander verglichen werden. Tabelle 4.7 zeigt die Ergebnisse der Rastersuche. Zu beachten ist, dass die jeweiligen Klassifizierer bei unterschiedlicher Merkmalsextraktion teils andere Hyperparameterwerte zur optimalen Klassifikation benötigen.

Klassifikator	**Merkmals-extraktion**	**Hyperparameter**
MultinomialNB	*BoW*	alpha: 1e-05
	TF-IDF	alpha: 1e-07
BernoulliNB	*Bow*	alpha: 1e-05
	TF-IDF	alpha: 1e-05
K-Nearest	*BoW*	n_neighbors: 2, weights: 'distance'
	TF-IDF	n_neighbors: 2, weights: 'distance'
SVC	*BoW*	kernel: 'sigmoid', decision_function: 'ovo', C: 100, gamma: 0.01
	TF-IDF	kernel: 'linear', decision_function: 'ovo', C: 100, gamma: 0.01
SGDClassifier	*BoW*	loss: 'squared_hinge', alpha: 1e-03, penalty: 'l1', tol: 0.2
	TF-IDF	loss: 'hinge', alpha: 1e-05, penalty: 'l1', tol: 0.2

Tabelle 4.7: Ergebnisse der Rastersuche

Die resultierenden Werte werden genutzt, um die Klassifizierungsverfahren miteinander zu vergleichen.

4.3.2 Modellauswahl

Die Auswahl des Klassifizierers erfolgt, im Gegensatz zur Rastersuche, mit dem kompletten Trainingsdatensatz. Als Kennwerte zur Validierung kommen Korrektklassifizierungsrate, Genauigkeit, Trefferquote und F1-Maß zum Einsatz. Der Klassifizierer mit den besten Kennwerten wird für die Implementierung des Prototypen genutzt.

Dafür werden diese mit denselben Merkmalen trainiert und anschließend validiert (vgl. Abb. 4.4). Die Merkmale werden dabei einmal mittels Bag-of-words und einmal per TF-IDF extrahiert.

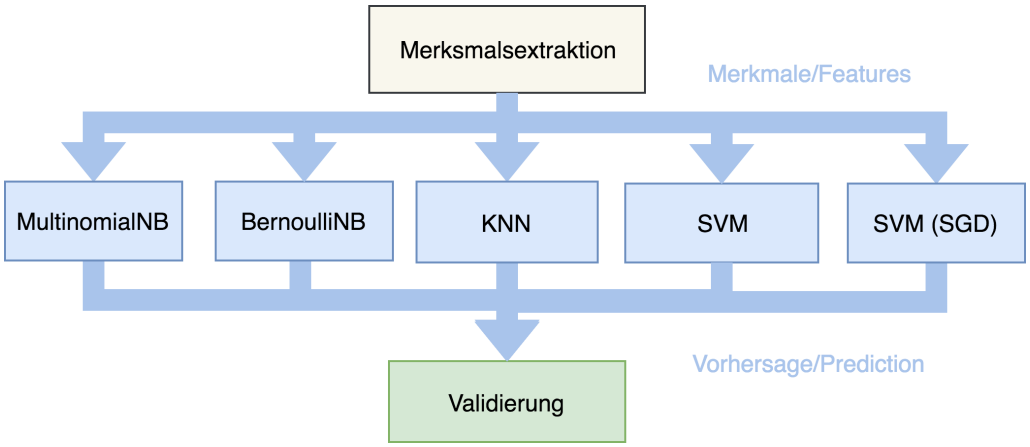

Abbildung 4.4: Auswahl eines Klassifikators

Resultate der Modellauswahl

Bei allen Klassifikationsverfahren stellt sich die Merkmalsextraktion mittels TF-IDF als besser oder, im Falle von BernoulliNB, als gleich gut zu Bag-of-Words heraus. Somit fällt die Auswahl auf TF-IDF als Verfahren zur Merkmalsextraktion.

Die Kennwerte zur Bewertung der Klassifizierer zeigen teilweise nur kleine Abweichungen im einstelligen Prozentbereich (vgl. Tab. 4.8). Die Implementierung der Support Vector Machine sticht mit allen Werten über 90% jedoch hervor und wird deshalb für die Implementierung des Protoypen genutzt. SVM mit stochastischem Gradientenabstiegsverfahren zeigt ähnliche hohe Werte, wird jedoch aufgrund der geringeren Genauigkeit nicht eingesetzt.

Klassifikator	**Merkmals-extraktion**	**Korrektklassifi-zierungsrate**	**Genauigkeit**	**Trefferquote**	**F1-Maß**
MultinomialNB	*BoW*	85,8%	85,3%	87,0%	84,8%
	TF-IDF	91,1%	92,6%	87,5%	88,7%
BernoulliNB	*BoW*	86,1%	87,5%	87,4%	85,8%
	TF-IDF	86,1%	87,5%	87,4%	85,8%
K-Nearest	*BoW*	83.7%	86,8%	83,9%	83,3%
	TF-IDF	87,0%	89,5%	88,2%	87,3%
SVC	*BoW*	88,9%	91,5%	83,5%	85,7%
	TF-IDF	**91,8%**	**94,0%**	**90,7%**	**91,3%**
SGDClassifier	*BoW*	89,1%	89,7%	89,4%	88,3%
	TF-IDF	91,5%	92,0%	91,5%	90,8%

Tabelle 4.8: Validierung der Klassifikationsverfahren

Anhand einer Konfusionsmatrix (vgl. Abb. 4.5) lässt sich die Vorhersage für die einzelnen Kategorien darstellen und die Zusammensetzung der Korrektklassifizierungsrate erkennen. Die Kategorie „Mobilität & Verkehrsmittel“ weist eine geringe Korrektklassifizierungsrate mit lediglich 71% Genauigkeit auf.

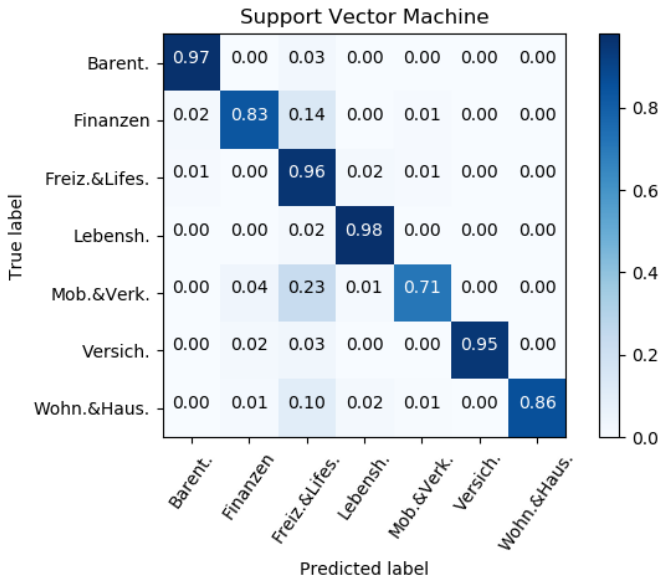

Abbildung 4.5: Konfusionsmatrix zur Support Vector Machine

Ursache hierfür ist zum einen die vergleichsweise geringe Anzahl von lediglich 73 Transaktionen. Andere Kategorien mit größerem Datenbestand, wie „Freizeit & Lifestyle“ oder „Lebenshaltung“, erreichen deutlich bessere Ergebnisse.

Des Weiteren weisen die Transaktionen in der Kategorie „Mobilität & Verkehrsmittel“ mehr Unterschiede auf, als die in anderen Kategorien. Vergleicht man die Summe der Levenshtein-Distanzen der Kategorien „Mobilität & Verkehrsmittel“ und „Versicherungen“, welche beide ähnlich viele Trainingsdaten enthalten (vgl. Abb. 4.6), lassen sich größere Unterschiede erkennen. Dazu wird die Levenshtein-Distanz der einzelnen Transaktionen untereinander berechnet und aufsummiert. Während Transaktionen in der Kategorie „Versicherungen“ eine Distanz von 70 aufweisen, ist diese bei der Kategorie „Mobilität & Verkehrsmittel“ mit 95 deutlich höher.

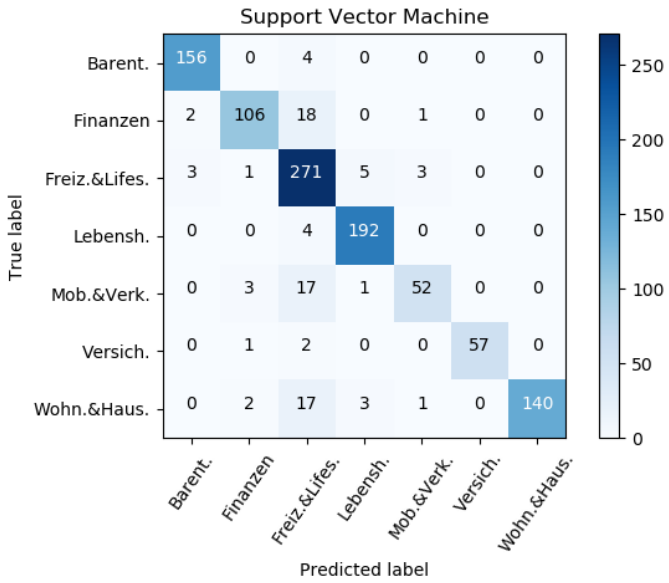

Abbildung 4.6: Konfusionsmatrix mit Absolutwerten zur Support Vector Machine

5 Prototypische Umsetzung

Dieses Kapitel beschreibt die prototypische Umsetzung des Kategorisierungssystems. Dazu wird das Modell inklusive der Datenvorverarbeitung aus dem vorherigen Kapitel integriert und die in Kapitel 2.2 definierten Anforderungen umgesetzt.

5.1 Architektur

Der Prototyp stellt ein System zur Kategorisierung von Banktransaktionen im CAMT-Format dar (vgl. Kap. 2.1.2). Da der Fokus dieser Arbeit auf einer effizienten Kategorisierung liegt und die Anbindung unterschiedlicher PFM-Anwendung vorgesehen ist, wird lediglich das Kernsystem zur Kategorisierung umgesetzt. Weitere Bestandteile einer PFM-Anwendungen, wie GUI-Design oder Login-Mechanismen der Banken, werden deshalb nicht betrachtet. Der implementierte Prototyp steht auf *GitHub*[1] zur Verfügung.

Der Prototyp ist so konzipiert, dass er als Teilsystem einer PFM-Anwendung eingesetzt werden kann. Abbildung 5.1 zeigt die schematische Architektur der Anwendung.

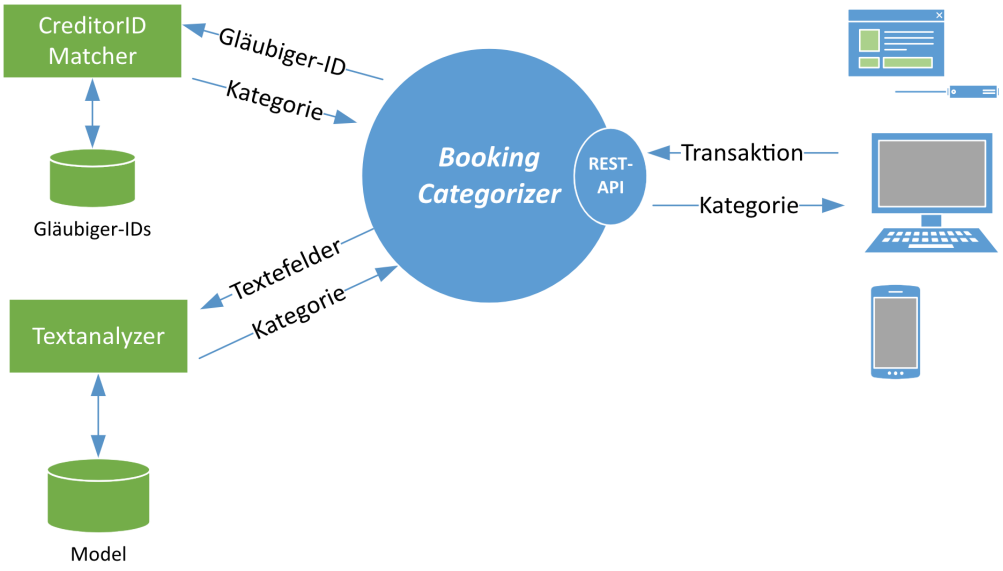

Abbildung 5.1: Architekturbild des Gesamtsystems

[1] https://github.com/markusgl/epayment-categorization

Die Form des Anfragestellers oder Clients ist dabei nicht weiter spezifiziert und kann ein Banksystem, Mobile-App oder Web-Anwendung darstellen. Das System setzt sich aus mehreren Komponenten zusammen, welche bei Eintreffen einer Anfrage, in Abhängigkeit der Transaktion, aufgerufen werden.

5.1.1 REST-Schnittstelle

Zur Kommunikation zwischen Kategorisierungssystem und Client wird eine REST-Schnittstelle eingesetzt, um eine programmiersprachenunabhängige und weit verbreitete Schnittstellentechnologie anzubieten [Pat17]. Die Daten werden dabei im JSON-Format des HTTP-Message-Bodies erwartet.

Neben einem Endpunkt zur Kategorisierung, stehen weitere Endpunkte zur Erweiterung des Modells, Rückmeldung des Benutzers und Beschreibung der Schnittstelle zur Verfügung (vgl. Tab. 5.1).

Pfadnamen	**HTTP-Verben**	**Beschreibung**
/howto	GET	Beschreibung der API
/categorize	POST	Kategorisierung einer Transaktion
/correctbooking	POST	Rückmeldung durch den Benutzer bei unklassifizierten Transaktionen
/addbooking	POST	Hinzufügen neuer Transaktionen zum Trainingsdatensatz des ML-Verfahren

Tabelle 5.1: Endpunkte der REST-API

Die Implementierung der Schnittstelle erfolgt mit Hilfe des Python Webframeworks *flask*[2].

5.1.2 Ablauf der Kategorisierung

Der Ablauf zur Kategorisierung (Pfadname */categorize*) erfolgt in mehreren Schritten (vgl. Abb. 5.2). Wie in Kapitel 2.2 gefordert, wird ein zweistufiges System verfolgt, welches die Gläubiger-ID vor der Textanalyse separat prüft. Außerdem wird eine Klasse zur Validierung und Typprüfung der Transaktionen implementiert.

[2]http://flask.pocoo.org/

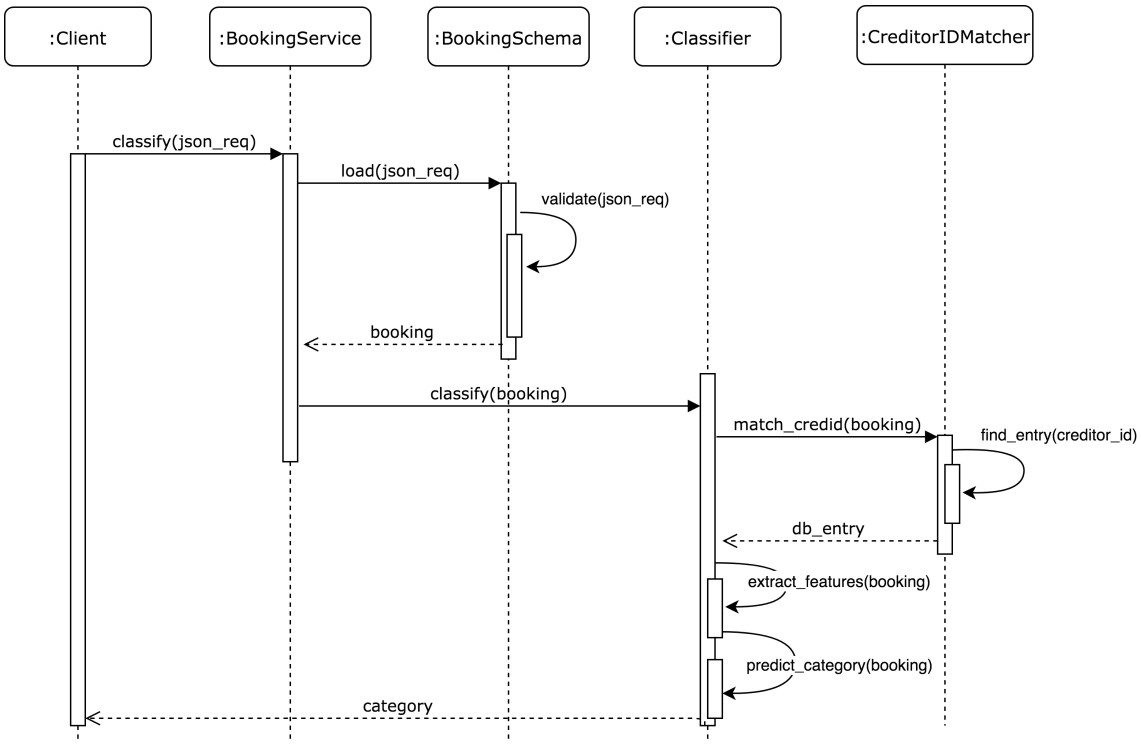

Abbildung 5.2: Sequenzdiagramm des Kategorisierungssystems

Empfang der Transaktionen

Der Client sendet die zu kategorisierende Transaktion im JSON-Format an die REST-Schnittstelle. Listing 5.1 zeigt eine vollständige Transaktion im JSON-Format mit allen möglichen Feldern. Dabei sind die Felder *text*, *usage*, *owner* und *amount* Pflichtangaben, ohne die eine Kategorisierung nicht möglich ist. Alle weiteren Felder sind optional und müssen keinen Wert enthalten.

```
{
  "booking_date":"01.09.2017",
  "valuta_date":"01.09.2017",
  "text":"FOLGELASTSCHRIFT",
  "usage":"KTO 123456789 Abschlag 60,00 EUR faellig 01.09.17 Musterstr. 1",
  "creditor_id":"DE05NAG00000005699",
  "owner":"N-Ergie Aktiengesellschaft",
  "iban":"DE19700500000000055162",
  "bic":"BYLADEMMXXX",
  "amount":"-60.00"
}
```

Listing 5.1: Beispiel-Transaktion im JSON-Format

Validierung und Deserialisierung

Zur Deserialisierung der Transaktion von JSON in ein Python-Objekt und zur Schema-Validierung wird die Python-Programmbibliothek *Marshmallow*[3] eingesetzt. Deserialisierung und Validierung werden bei Marshmallow in einem Schritt durchgeführt. Dabei wird zunächst eine Deserialiserung (*load*) anhand des vorab definierten Schemas (*BookingSchema*) versucht. Schlägt diese aufgrund fehlender oder ungültiger Felder fehl, wird einen Fehler vom Typ *ValidationError* erzeugt. Die Anfrage wird mit dem HTTP-Status-Code *400* als ungültig an den Client zurückgewiesen.

```
1  @app.route("/categorize", methods=['POST'])
2  def categorize():
3    try:
4       req_data = request.get_json()
5       booking_schema = BookingSchema()
6       booking, errors = booking_schema.load(req_data)
7         ...
8     except ValidationError as err:
9       err.messages
10      resp = render_template('400.html'), 400
```

Listing 5.2: REST-Endpunkt zu Klassifizierung

Gläubiger-ID Vergleich

Eine valide Transaktion wird der Klasse *Classifier* übergeben. Diese führt zunächst den Vergleich der Gläubiger-ID durch. Sämtliche Gläubiger-IDs sind in einer dokumentenbasierten *MongoDB*[4] gespeichert und stammen aus dem Datenbestand der Firma adorsys. Der Datenbestand enthält Firmen, welche nur einer Kategorie zugeordnet werden können, wie etwa Versicherungen oder Energieversorger. Die Einträge der Datenbank enthalten die Gläubiger-ID, den Zahlungsempfänger und die entsprechende Kategorie (vgl. Listing 5.3).

```
1  {
2     "_id" : ObjectId("5a7af4406ce88607e7fc922c"),
3     "category" : "wohnenhaushalt",
4     "owner" : "n-ergie",
5     "creditorid" : "de05nag0000000XXXX"
6  }
```

Listing 5.3: Datenbankeintrag mit Gläubiger-ID

[3] https://marshmallow.readthedocs.io/en/latest/
[4] https://www.mongodb.com/de

Textschlüssel im Verwendungszweck

Die Gläubiger-ID kann ebenfalls im Verwendungszweck, gemäß SEPA-Purpose-Code, vorkommen (vgl. Kap. 2.2.1). Ist das Feld „creditor_id“ der Transaktion leer, wird zunächst der Verwendungszweck nach dem Textschlüssel „Einreicher-ID“ untersucht. Diesem String folgt, mit einem Leerzeichen getrennt, die Gläubiger-ID. Wird diese im Verwendungszweck gefunden, findet anschließend die Suche in der MongoDB, gemäß des vorherigen Abschnittes, statt.

Textanalyse und Antwort

Ist die Gläubiger-ID der Transaktion nicht im Datenbestand enthalten oder verfügt die Transaktion über keine Gläubiger-ID, wird die Textanalyse durchgeführt. Die Textanalyse folgt dabei der Vorgehensweise aus Kapitel 4. Somit werden die Felder Buchungstext, Verwendungszweck und Zahlungsempfänger zusammengefasst und eine Vorhersage mit der Support Vector Machine durchgeführt. Liegt die Vorhersage bei keiner Klasse über 70%, wird die Rückweisungsklasse „Sonstiges“ als Kategorie an den Client zurückgeliefert.

5.1.3 Rückmeldung des Benutzers

Um das Kategorisierungssystem stetig zu verbessern, wird dem Benutzer die Möglichkeit gegeben, nicht kategorisierte Transaktionen manuell zu annotieren. Diese werden anschließend in den Trainingsdatensatz aufgenommen und das Modell neu trainiert.

Jede Transaktionen, welche die Rückweisungsklasse zugeordnet bekommt, wird zu diesem Zweck temporär zwischengespeichert. Damit die Anfrage des Clients der gespeicherten Transaktion zugeordnet werden kann, erhält der Client bei Antworten mit der Kategorie „Sonstiges“ ein Session-Cookie.

Der Client kann dann über die Methode */correctbooking* die entsprechende Kategorie zurücksenden. Das Kategorisierungssytem erweitert die entsprechende Transaktion um die erhaltene Kategorie und fügt diese dem Trainingsdatensatz hinzu.

5.1.4 Modellerweiterung

Neben der Möglichkeit nicht kategorisierte Transaktionen zu annotieren, kann auch direkt eine annotierte Transaktion an das Kategorisierungssystem gesendet werden. Diese Methode ist für administrative Zwecke und weniger als Funktion einer Endkunden-Applikation vorgesehen. Da der Client nicht weiter spezifiziert ist, sind jedoch auch andere Verwendungen möglich.

Zur Erweiterung wird die Transaktion, um das Attribut *category* ergänzt und dem Trainingsdatensatz hinzugefügt. Anschließend wird das Training des Modells über einen internen Methodenaufruf angestoßen.

6 Schlussbetrachtung

6.1 Fazit

In dieser Arbeit wurde gezeigt, wie eine effiziente Kategorisierung von Banktransaktionen entworfen werden kann. Dabei lag ein besonderes Augenmerk auf dem Einsatz von Machine-Learning-Verfahren zur Textanalyse und der zielgerichteten Optimierung für Transaktionsdaten. Daraus entstand eine Anwendung, welche genutzt werden kann, um Transaktionen zu kategorisieren.

Zur Vorverarbeitung der Daten wurden zunächst Mechanismen der lexikalischen Analyse genutzt, um die Daten zu vereinheitlichen und somit die Leistungsfähigkeit des Klassifizierers zu erhöhen. Dabei zeigte sich, dass gerade bei der Textklassifikation die Vorverarbeitung einen wichtigen Baustein zur Lösung des Problems darstellt. Auch stellte sich heraus, dass gängige Tools aus Text Mining und Natural Language Processsing nicht ohne weiteres auf Banktransaktionen angewendet werden können und einer zielgerichteten Anpassung bedürfen.

Des Weiteren wurde festgestellt, dass Verfahren des Natural Language Processing aufgrund der lexikographisch nicht immer korrekten Schreibweisen einer Transaktion nur bedingt Anwendung finden können. So kann etwa eine Lemmatisierung die Informationen zu stark verändern und die Wiedererkennung weiter erschweren. Auch Mittel wie Tokenisierung sind bei Banktransaktionen nicht ohne Weiteres anwendbar.

Die Anwendung von Machine-Learning-Verfahren zeigte gute Ergebnisse auf einem relativ kleinen Datensatz. Die Merkmalsextraktion stellte sich dabei als ein wichtiger Faktor zur Verbesserung der Genauigkeit dar. Dabei war es wichtig, nur die Felder mit relevanten Informationen auszuwählen, um den Anteil weniger aussagekräftiger Texte zu minimieren.

Die Support Vector Machine zeigte bei der Validierung die besten Ergebnisse. Doch auch das deutlich einfachere Verfahren des Naive-Bayes-Klassifikators lieferte noch gute Vorhersagen. Durch eine Rastersuche zur Ermittlung der optimalen Hyperparameter konnte die Genauigkeit der Kategorisierung nochmals verbessert werden. Dieses relativ

zeitaufwändige Verfahren erwies sich als überaus wichtig für die Optimierung des Klassifikationsverfahren. Für den produktiven Einsatz der Anwendung sollte deshalb nicht auf deren Einsatz verzichtet werden.

Auch wenn die Vorhersagen des Machine-Learning-Verfahrens bereits gute Ergebnisse lieferten, ist die Nutzung statisch gepflegter Listen nicht unbedingt obsolet. Anhand einer Liste mit Gläubiger-IDs kann die Kategorie in einigen Fällen schneller und genauer bestimmt werden.

Die umgesetzte Anwendung zeigte jedoch auch, dass es zukünftig durchaus möglich ist, vollständig auf statische Listen zu verzichten. Die Kategorisierung lieferte gute Vorhersagen, welche durch stetige Erweiterung des Modells mit neuen Transaktionen und Rückmeldungen des Benutzers noch verbessert werden könnte. Somit ist ein Wegfall der Listen und statischen Regelwerke in Zukunft durchaus denkbar.

6.2 Ausblick

Die hier betrachteten Machine-Learning-Verfahren decken nur einen Teil der heute verfügbaren Verfahren ab. Es ist nicht ausgeschlossen, dass andere oder neuere Verfahren ebenfalls gute Ergebnisse erreichen. Logistische Regression, Random Forests oder Perzeptron sind ebenfalls beliebte Verfahren zur Klassifikation. Die Evaluation deren Leistungsfähigkeit bei Banktransaktionen wäre ebenfalls interessant. Denkbar wäre auch eine Kombination mehrerer Verfahren, um etwa die Trainingsdaten durch Clustering vorab weiter zu vereinheitlichen.

Die fortschreitende Popularität Künstlicher Neuronaler Netze findet auch im Bereich der Textklassifikation Anklang und hat in den letzten Jahren neue Erkenntnisse erbracht. Hervorzuheben sind dabei *Word2Vec* und *GloVe*, welche beide ein Vorhersagemodell nutzen, um die Bedeutung von Wörtern anhand ihrer Nachbarwörter zu erkennen [MSC+13, PSM14]. Auch für Banktransaktionen wäre dieses Vorgehen denkbar, um Wörter aus den Textfeldern im Kontext zu betrachten. Dies könnte auch weitere Erkenntnisse zur Bildung von Unterkategorien erbringen. Auch existieren mittlerweile vortrainierte Modelle zur Textklassifikation, welche mittels Transferlernen für andere Probleme genutzt werden können [HR18].

Literaturverzeichnis

[BGV92] BOSER, Bernhard E. ; GUYON, Isabelle M. ; VAPNIK, Vladimir N.: A training algorithm for optimal margin classifiers. In: *Proceedings of the fifth annual workshop on Computational learning theory*, ACM, 1992, S. 144–152

[BM14] BIEMANN, Chris ; MEHLER, Alexander: *Text Mining: From Ontology Learning to Automated Text Processing Applications*. Springer, 2014

[CEE+09] CARSTENSEN, Kai-Uwe ; EBERT, Christian ; EBERT, Cornelia ; JEKAT, Susanne ; LANGER, Hagen ; KLABUNDE, Ralf: *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Springer-Verlag, 2009

[CL11] CHANG, Chih-Chung ; LIN, Chih-Jen: LIBSVM: A Library for Support Vector Machines. In: *ACM Trans. Intell. Syst. Technol.* 2 (2011), Nr. 3, 27:1–27:27. http://dx.doi.org/10.1145/1961189.1961199. – DOI 10.1145/1961189.1961199. – ISSN 2157–6904

[CL16] CLEVE, Jürgen ; LÄMMEL, Uwe: *Data mining*. Berlin ; Boston : De Gruyter Studium, 2016. – ISBN 978–3–11–045675–2

[com17] COMDIRECT: *comdirect Fintech-Studie: Das Wachstum hält an - Pressemitteilung - Presse | comdirect.de*. https://www.comdirect.de/cms/ueberuns/de/presse/cori1088_1211.html. Version: Juni 2017

[CTo94] CAVNAR, William B. ; TRENKLE, John M. ; OTHERS: N-gram-based text categorization. In: *Ann Arbor MI* 48113 (1994), Nr. 2, S. 161–175

[CV95] CORTES, Corinna ; VAPNIK, Vladimir: Support-vector networks. In: *Machine learning* 20 (1995), Nr. 3, S. 273–297

[Deu17a] DEUTSCHE BUNDESBANK: *Gläubiger-Identifikationsnummer*. https://www.bundesbank.de/Navigation/DE/Aufgaben/Unbarer_Zahlungsverkehr/SEPA/Glaeubiger_Identifikationsnummer/glaeubiger_identifikationsnummer.html. Version: November 2017

[Deu17b] DEUTSCHE BUNDESBANK: *SEPA*. https://www.bundesbank.de/Redaktion/DE/Standardartikel/Aufgaben/Unbarer_Zahlungsverkehr/sepa.html. Version: Oktober 2017

[Deu18] DEUTSCHE BUNDESBANK: *FinTechs - Finanztechnologie Unternehmen*. `https://www.bundesbank.de/Redaktion/DE/Standardartikel/Aufgaben/Bankenaufsicht/fintechs.html`. Version: 2018

[DHSW16] DORFLEITNER, Gregor ; HORNUF, Lars ; SCHMITT, Mathias ; WEBER, Martina: *FinTech-Markt in Deutschland*. Oktober 2016

[Die17] DIE DEUTSCHE KREDITWIRTSCHAFT: *DFÜ-Verfahren EBICS*. `https://die-dk.de/zahlungsverkehr/electronic-banking/dfu-verfahren-ebics/`. Version: 2017

[DSAH14] DAPP, Thomas F. ; SLOMKA, Lars ; AG, Deutsche B. ; HOFFMANN, Ralf: Fintech–Die digitale (R) evolution im Finanzsektor. In: *Algorithmenbasiertes Banking mit human touch. Hg. v. Deutsche Bank AG. Frankfurt am Main* (2014)

[Ert09] ERTEL, Wolfgang: Grundkurs Künstliche Intelligenz. In: *Auflage, Wiesbaden* (2009)

[Eur17] EUROPEAN PAYMENTS COUNCIL: *SEPA Credit Transfer Scheme Inter-Bank Implementation Guidelines*. `https://www.europeanpaymentscouncil.eu/document-library/implementation-guidelines/sepa-credit-transfer-scheme-customer-bank-implementation`. Version: November 2017

[Fer03] FERBER, Reginald: *Information Retrieval. dpunkt. verlag*. Heidelberg, 2003

[GBC16] GOODFELLOW, Ian ; BENGIO, Yoshua ; COURVILLE, Aaron: *Deep Learning*. MIT Press, 2016

[GPB17] GRISEL, O. ; PRETTENHOFER, P. ; BLONDEL, M.: *Sample pipeline for text feature extraction and evaluation — scikit-learn 0.19.1 documentation*. `http://scikit-learn.org/stable/auto_examples/model_selection/grid_search_text_feature_extraction.html`. Version: 2017

[Hey15] HEYDT, M.: *Learning pandas*. Packt Publishing, 2015 (Community Experience Distilled). – ISBN 978–1–78398–513–5

[HR18] HOWARD, Jeremy ; RUDER, Sebastian: Fine-tuned Language Models for Text Classification. In: *arXiv preprint arXiv:1801.06146* (2018)

[Idr13] IDRIS, Ivan: *NumPy Beginner's Guide*. Packt Publishing Ltd, 2013

[Idr14] IDRIS, Ivan: *Learning NumPy Array*. Packt Publishing Ltd, 2014

[IMF13] INGERSOLL, Grant S. ; MORTON, Thomas S. ; FARRIS, Andrew L.: *Taming text: how to find, organize, and manipulate it*. Shelter Island : Manning, 2013. – ISBN 978–1–933988–38–2. – OCLC: ocn772977853

[KS16] KOLLMANN, Tobias ; SCHMIDT, Holger: *Deutschland 4.0: Wie die digitale Transformation gelingt*. Springer, 2016

[MG16] MÜLLER, Andreas C. ; GUIDO, Sarah: *Introduction to machine learning with Python: a guide for data scientists*. Ö'Reilly Media, Inc.", 2016

[MRS08] MANNING, Christopher D. ; RAGHAVAN, Prabhakar ; SCHÜTZE, Hinrich: *Introduction to information retrieval*. New York : Cambridge University Press, 2008. – ISBN 978–0–521–86571–5. – OCLC: ocn190786122

[MSC+13] MIKOLOV, Tomas ; SUTSKEVER, Ilya ; CHEN, Kai ; CORRADO, Greg S. ; DEAN, Jeff: Distributed representations of words and phrases and their compositionality. In: *Advances in neural information processing systems*, 2013, S. 3111–3119

[Nie03] NIEMANN, Heinrich: *Klassifikation von Mustern*. Erlangen : Springer, 2003

[Pat17] PATNI, Sanjay: *Pro RESTful APIs*. Springer, 2017

[Per14] PERKINS, Jacob: *Python 3 text processing with NLTK 3 cookbook*. Packt Publishing Ltd, 2014

[PSM14] PENNINGTON, Jeffrey ; SOCHER, Richard ; MANNING, Christopher D.: GloVe: Global Vectors for Word Representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, 1532–1543

[PVG+11] PEDREGOSA, F. ; VAROQUAUX, G. ; GRAMFORT, A. ; MICHEL, V. ; THIRION, B. ; GRISEL, O. ; BLONDEL, M. ; PRETTENHOFER, P. ; WEISS, R. ; DUBOURG, V. ; VANDERPLAS, J. ; PASSOS, A. ; COURNAPEAU, D. ; BRUCHER, M. ; PERROT, M. ; DUCHESNAY, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[Ras15] RASCHKA, Sebastian: *Python machine learning*. Packt Publishing Ltd, 2015

[Run12] RUNKLER, Thomas A.: *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. Vieweg, 2012. – ISBN 3–8348–2588–3 978–3–8348–2588–9

[sci17] SCIKIT-LEARN DEVELOPERS: *scikit-learn 0.19.1 documentation - Nearest Neighbors*. `http://scikit-learn.org/stable/modules/neighbors.html#classification`. Version: 2017

[Van16] VANDERPLAS, Jake: *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc., 2016

[WKQ+08] WU, Xindong ; KUMAR, Vipin ; QUINLAN, J R. ; GHOSH, Joydeep ; YANG, Qiang ; MOTODA, Hiroshi ; MCLACHLAN, Geoffrey J. ; NG, Angus ; LIU, Bing ; PHILIP, S Y. ; OTHERS: Top 10 algorithms in data mining. In: *Knowledge and information systems* 14 (2008), Nr. 1, S. 1–37

[ZM16] ZHAI, ChengXiang ; MASSUNG, Sean: *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining*. New York, NY, USA : Association for Computing Machinery and Morgan & Claypool, 2016. – ISBN 978–1–970001–17–4