Effektive Integration von heterogenen Produktkatalogen im schnelllebigen Umfeld des E-Commerce

Dissertation zur Erlangung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.)

Vorgelegt im Fachbereich Elektrotechnik/Informatik der Universität Kassel

Von Oliver Michael Schmidts, M. Sc.

Gutachter:
Prof. Dr. Claudia Fohry
Prof. Dr. Bodo Kraft

Disputation am 08.12.2023

Zusammenfassung

Online-Marktplätze generieren von Jahr zu Jahr einen größeren Anteil des Einzelhandelsumsatzes. Ein wichtiger Faktor für den Erfolg von Online-Marktplätzen ist die korrekte Darstellung der Produktdaten für ihre Kunden. Diese Daten werden häufig von Zulieferern in Form von Produktkatalogen zur Verfügung gestellt, die in den Online-Marktplatz integriert werden müssen.

Um dies zu erreichen, sind insbesondere kleine und mittelständische Unternehmen häufig auf aufwändige manuelle Arbeitsschritte bei der Datenintegration angewiesen. Ein solcher Schritt bei der Datenintegration ist die Ermittlung von Zuordnungen zwischen den Benennungsschemata der Produktattribute der Zulieferer und dem Benennungsschema des Marktplatzes. Der Schritt ist aufwändig, da Zulieferer individuelle und häufig wechselnde Benennungsstrategien verwenden.

Jede manuelle Katalogintegration erzeugt Paare von Produktkatalogen vor und nach dem Integrationsprozess. Bisher wurden diese Daten nicht genutzt, obwohl sie das Potenzial besitzen, die Schemazuordnung zu vereinfachen. Diese Arbeit widmet sich daher der Frage, inwieweit die im Rahmen eines Integrationsprozesses anfallenden Daten genutzt werden können, um das Bestimmen der Zuordnungen zu automatisieren oder zumindest die manuelle Zuordnung durch ein Empfehlungssystem zu unterstützen.

Zwei grundlegende Ansätze wurden verfolgt. Zunächst wurden ausschließlich die Attributnamen der Schemata für die Schemazuordnung verwendet. Dazu wurden mehrere Verfahren des maschinellen Lernens (ML) in Kombination mit verschiedenen Vektorisierungsstrategien für Attributnamen mit Verfahren verglichen, die ohne ML auskommen. Aufbauend auf den dabei gewonnenen Erkenntnissen wurde dann mit Attribut Label Ranking (ALR) ein neues Verfahren entwickelt, das neben Attributnamen auch Attributinstanzen zum Lernen von Zuordnungen verwendet. Die Evaluation der Verfahren erfolgte jeweils anhand von Produktdaten aus Integrationsprozessen eines Online-Marktplatzes für Antikörper.

Die Ergebnisse zeigen das Potenzial der ML-Verfahren. Anhand von üblichen Vergleichsmetriken konnte die Erkennungsrate von Zuordnungen textbasierter Attribute im Vergleich zu ähnlichen Verfahren aus anderen Anwendungsbereichen um bis zu 0.5 verbessert werden. Erreichte Werte für Empfehlungsmetriken zeigen, dass die Verfahren als Empfehlungssystem zur Unterstützung der manuellen Arbeit eingesetzt werden können. Darüber hinaus ermöglicht ALR unter bestimmten Voraussetzungen eine vollautomatische Erkennung der Zuordnungen.

Abstract

Online marketplaces generate a larger share of retail sales every year. An important factor for the success of online marketplaces is the correct presentation of product data to their customers. Suppliers often provide this data through product catalogs that the online marketplace needs to integrate into their web shop.

In order to achieve this, small and medium-sized enterprises, in particular, often have to rely on time-consuming manual steps in data integration. One such step in data integration is the determination of mappings between the supplier naming schemes for product attributes and the marketplace naming scheme. This step is time-consuming because suppliers use individual naming strategies that can also change frequently.

Every manual catalog integration creates pairs of product catalogs before and after the integration process. So far, this data has not yet been used, although it has the potential to simplify schema mapping. This work is dedicated to the question of how the data generated during an integration process can be used to automate the determination of schema mappings or, at least, to support the manual mapping process through a recommendation system.

This work employed two main approaches. First, only the attribute names of the schemas were used for schema mapping. For this purpose, several machine learning (ML) methods in combination with different vectorization strategies for attribute names were compared with methods that do not rely on ML. Second, by building upon the insights gained, a new method was developed with Attribute Label Ranking (ALR) using attribute instances and names to learn mappings. Both methods were evaluated using actual product data from integration processes of an online marketplace for antibodies.

The results show the potential of the ML methods. The recognition rate of mappings of text-based attributes improves by up to 0.5 in precision, recall and F-score compared to similar methods from different application areas. Achieved values for recommendation metrics show that the methods can be used as a recommendation system to support manual work. Furthermore, ALR enables fully automatic schema mappings under specific conditions.

Inhaltsverzeichnis

Zusammenfassung **I**

Abstract **III**

1 Einleitung **1**
1.1 Motivation 2
1.2 Problemstellung 5
1.3 Forschungsfragen 14
1.4 Beiträge der Arbeit 14
1.5 Publikationen 18
1.6 Aufbau der Arbeit 21

2 Grundlagen zur Katalogintegration **23**
2.1 Theoretische Perspektive 23
2.1.1 global-as-view (GAV) 25
2.1.2 local-as-view (LAV) 25
2.1.3 global-as-view vs. local-as-view im Online-Handel 26
2.1.4 Ontologie- und Schema-Matching 26
2.1.5 Integration von Produktkatalogen 29
2.2 Schema-Abgleich für Produktkataloge 31
2.2.1 Katalogschemata von Zulieferern 32
2.2.2 Beziehungen zwischen Eingangs- und Zielschema 32
2.3 Gängige Standards für Produktdaten 35
2.3.1 schema.org/Product 36
2.3.2 GS1-Standard 37
2.3.3 Shopify 38
2.4 Kategorisierung von Produktdaten 41
2.4.1 Identifikationsnummern 42
2.4.2 Numerische Werte 42

2.4.3 Wertelisten . . . 42
2.4.4 Unstrukturierter Text . . . 43
2.4.5 Uniform Resource Locators . . . 43
2.5 Herausforderungen im Umgang mit Produktdaten . . . 43
2.5.1 Identifizierung gleicher Konzepte . . . 44
2.5.2 Erkennung zusätzlicher Informationen . . . 45
2.5.3 Vereinheitlichung . . . 46
2.5.4 Wortreichtum der Attributinstanzen . . . 46
2.5.5 Fehlende (Daten-)Strukturen . . . 47
2.6 Zusammenfassung . . . 47

3 Grundlagen des maschinellen Lernens 49
3.1 Problemlösung durch maschinelles Lernen . . . 49
3.2 Arten von ML-Systemen . . . 51
3.2.1 Supervised ML-Systeme . . . 51
3.2.2 Unsupervised ML-Systeme . . . 52
3.2.3 Generalisierungsstrategien für ML . . . 52
3.3 Entwicklung von ML-Systemen . . . 53
3.4 Typische Herausforderungen . . . 56
3.4.1 Unzureichende Trainingsdaten . . . 56
3.4.2 Irrelevante Merkmale . . . 57
3.4.3 Überanpassung . . . 57
3.4.4 Unteranpassung . . . 58
3.5 Schemazuordnung durch maschinelles Lernen . . . 59
3.5.1 Support Vector Machines . . . 59
3.5.2 Neuronale Netze . . . 62
3.5.3 Entscheidungsbäume . . . 64
3.5.4 Ensemble-Methoden . . . 66
3.6 Metriken zur Bewertung von Matching-Systemen . . . 68
3.6.1 Mögliche Ergebnisse einer Klassifikation . . . 69
3.6.2 Beurteilung von Klassifikationsergebnissen . . . 69
3.6.3 Beurteilung von Vorschlägen und Kategorisierungen . . . 73
3.6.4 Subjektive Zufriedenheit . . . 76
3.7 Zusammenfassung . . . 76

4 Textmerkmale für Matching-Systeme 79
4.1 Textsegmentierung . . . 79

4.2 Normalisierung 81
4.3 Ermitteln von Unterschieden in Zeichenfolgen 82
4.3.1 Lexikalische Ähnlichkeit 84
4.3.2 Phonetische Ähnlichkeit 88
4.3.3 Semantische Ähnlichkeit 91
4.4 Wortvektoren 92
4.4.1 One-Hot-Kodierung 93
4.4.2 TF-IDF 93
4.4.3 Word2Vec 94
4.4.4 FastText 96
4.4.5 Vektorbasierte Kosinus-Ähnlichkeit 97
4.5 Zusammenfassung 97

5 Bestimmung der Schemazuordnungen über Attributnamen 99
5.1 Aufbau als Klassifikationsansatz 99
5.2 Geeignete Vektortransformationen für Klassifikationsansätze 101
5.2.1 Ähnlichkeitsmetriken als Feature-Vektor 101
5.2.2 One-Hot-Kodierung und Tf-Idf-Vektoren 102
5.2.3 FastText 104
5.3 Datengrundlage 105
5.4 Gewählte Konfiguration der Verfahren 108
5.4.1 Referenzansätze 108
5.4.2 Lernende Verfahren 111
5.4.3 Auflistung der Verfahren 116

6 Evaluation der Zuordnungen über Attributnamen 119
6.1 Szenario 1 120
6.1.1 Trainings- und Testdaten 120
6.1.2 Auswertung auf dem Testdatensatz 121
6.2 Szenario 2 122
6.3 Vergleich der Szenarien und Ergebnisanalyse 124
6.3.1 Analyse der Vorhersageergebnisse 124
6.3.2 Ergebnisstabilität 126
6.4 Szenario 3 127
6.4.1 Trainings- und Testdatensatz 128
6.4.2 Auswertung auf dem Testdatensatz 128
6.4.3 Auswertung der Überkreuzvalidierung 130

6.5 Szenario 4 130
6.6 Bewertung der Ergebnisse und Analyse möglicher Fehler 132
6.6.1 Bewertung der Wortvektoren 132
6.6.2 Bewertung von COMA 133
6.6.3 Bewertung der Wortähnlichkeiten 133
6.6.4 Mögliche Verwechslung von Attribut-Zuordnungen 134
6.7 Auswertung in der Praxis 134
6.7.1 Datensatz 134
6.7.2 Ergebnisse der Klassifikation 135
6.7.3 Ergebnisse als Empfehlungsgeber 136
6.7.4 Auswirkungen auf die manuelle Arbeit 137
6.7.5 Einschränkungen in der Praxisauswertung 138

7 Attribut Label Ranking 139
7.1 Aufbau 139
7.2 Bildung von Instanzvektoren 142
7.2.1 Lexikalische Metadaten aus Attributinstanzen 143
7.2.2 Textvektorisierung 144
7.3 Datengrundlage 146
7.3.1 Übersicht und Datenerhebung 146
7.3.2 Erstellung eines Silberstandards 148
7.3.3 Datenerweiterung 151
7.3.4 Einschränkungen bei der Datenerhebung 153

8 Evaluation des ALR-Verfahrens 155
8.1 Szenario 1: ALR mit einem einfachen Netzwerk 156
8.1.1 Datensatz 157
8.1.2 Aufbau des Modells für ALR 159
8.1.3 Zusammensetzung des Feature-Vektors 161
8.1.4 Ergebnisse unter Betrachtung einer einzelnen Zielklasse . . . 163
8.1.5 Ergebnisse mit mehreren Zielklassen 169
8.2 Szenario 2: Multimodales Modell und FastText-Vektoren 171
8.2.1 Aufbau und Konfiguration 171
8.2.2 Klassifikationsergebnisse 174
8.2.3 Ergebnisse als Empfehlungssystem mit mehreren Zielklassen 176
8.3 Einschränkungen in der Evaluation 179

9 Verwandte Arbeiten und Einordnung der Ergebnisse **181**
9.1 COMA/COMA 181
9.2 ProbaMap 183
9.3 Corpus Based Matching 183
9.4 ASMOV 185
9.5 CIDER 186
9.6 Schema-Matching mit Wortvektoren 187
9.7 Zuordnung von Schema-Attributnamen durch Inhaltsanalyse 188
9.8 Erstellen von Vektoren auf relationale Datensätzen für die Dateninte-gration 189
9.9 Einordnung der Ergebnisse in den wissenschaftlichen Kontext 189

10 Schlussbemerkungen **193**
10.1 Zusammenfassung 193
10.2 Ausblick 197

Akronyme **i**

Abbildungsverzeichnis **v**

Listingverzeichnis **vii**

Tabellenverzeichnis **ix**

Eigene Publikationen **xi**

Literaturverzeichnis **xii**

Kapitel 1

Einleitung

Im Jahr 2021 betrug der E-Commerce-Umsatz im Einzelhandel weltweit 5211 Mrd. US-Dollar und wird bis 2026 auf ein geschätztes Volumen von 8148 Mrd. US-Dollar ansteigen [1]. In Deutschland führte Amazon im Jahr 2020 die Rangliste der 100 umsatzstärksten Business to Customer (B2C) Online-Shops mit deutlichem Abstand vor der Otto-Gruppe und Zalando an [2].

Bei dem starken Wachstum im E-Commerce stellen gleichzeitig wachsende Retouren die Shop-Betreiber vor größer werdende finanzielle und logistische Herausforderungen [3, 4]. Dabei werden Produkte hauptsächlich aufgrund falscher bzw. uneinheitlicher Größenangaben zurückgeschickt. Auch mangelhafte Produktbeschreibungen werden von Konsumenten häufig genannt [5]. Durch bessere oder detailliertere Produktinformationen ließe sich folglich der Umfang der Retouren verringern [6].

In diesem Kapitel werden zunächst die vielfältigen Herausforderungen aufgezeigt, die bei der Integration von Produktkatalogen bzw. -daten für E-Commerce Unternehmen, insbesondere auch für kleine und mittelständische Unternehmen (KMU), entstehen. Danach wird die im Rahmen eines Kooperationsprojekts durch die Dissertation betrachtete Problemstellung beschrieben und formalisiert. Dabei wird eine Reihe von Forschungsfragen aufgeworfen. Anschließend wird der wissenschaftliche Beitrag der Dissertation zur Beantwortung der Forschungsfragen dargestellt. Abschließend erfolgt eine Auflistung und Einordnung der eigenen Publikationen, bevor auf den weiteren Aufbau der Arbeit eingegangen wird.

1.1 Motivation

Die Erstellung eines Online-Shops bzw. eines E-Commerce-Angebots wird durch *Services* wie Shopify[1] oder IONOS[2] immer einfacher. Um einem *Kunden*, der im Online-Shop etwas kaufen möchte, Produkte anzubieten, muss ein *Online-Shop-Betreiber* diese Produkte auf einer Webseite anbieten. Der Kunde kann die vorhandenen Produkte durchsuchen, miteinander vergleichen und kaufen. Die genannten Services vereinfachen den Betrieb eines Online-Shops enorm, indem sie die Verwaltung der angebotenen Produkte, die Darstellung auf der Webseite, Zahlungsmöglichkeiten und gegebenenfalls sogar die Logistik übernehmen.

Neben den Services bieten auch größere Plattformen, wie beispielsweise Amazon[3], eBay[4] oder Galaxus[5], die Möglichkeit, einen Online-Shop zu erstellen. Ein so erstellter Online-Shop erhält eine Unterseite innerhalb der genutzten Plattform anstelle einer eigenen Internetadresse. Die Plattformen ermöglichen Kunden mehrere Online-Shops gleichzeitig zu durchsuchen und deren Produkte über den gleichen Zahlungsweg zu kaufen. Solche Plattformen werden in dieser Arbeit als *Marktplatz* bezeichnet.

Die oben genannten Services und Marktplätze vereinfachen vor allem KMU den Betrieb eines Online-Shops, da die technischen Hürden erheblich reduziert werden. Trotzdem muss ein Betreiber Daten zu seinen Produkten in den Online-Shop integrieren. Je nach genutztem Service oder Marktplatz unterscheiden sich die Möglichkeiten der Datenintegration für den Online-Shop-Betreiber. Beispielsweise müssen bei eBay und IONOS Produkte manuell durch die Shop-Betreiber in entsprechende Online-Formulare eingetragen werden. Amazon [7], Galaxus [8] und Shopify [9] bieten dagegen die Möglichkeit, die Daten über Dateien hochzuladen. Eine einzelne Datei kann fast immer mehr als ein Produkt enthalten [10]. In diesem Fall wird die Datei im folgenden *Produktkatalog* genannt.

Im Hintergrund werden durch die Marktplätze sogenannte *extract, transform, load (ETL)-Prozesse* durchgeführt, die mit dem Einlesen der gelieferten Daten beginnen und mit dem Einstellen der Daten in ein Data Warehouse (DWH) enden. Letztendlich werden die Daten aus dem DWH den Kunden angezeigt. Auch in Abbildung 1.1 finden sich diese Prozessschritte grundsätzlich wieder. Die Abbildung zeigt den Ab-

[1] https://www.shopify.de/
[2] https://www.ionos.de/eshop-loesungen/onlineshop-erstellen
[3] https://sell.amazon.de/
[4] https://www.ebay.de/
[5] https://www.galaxus.de/

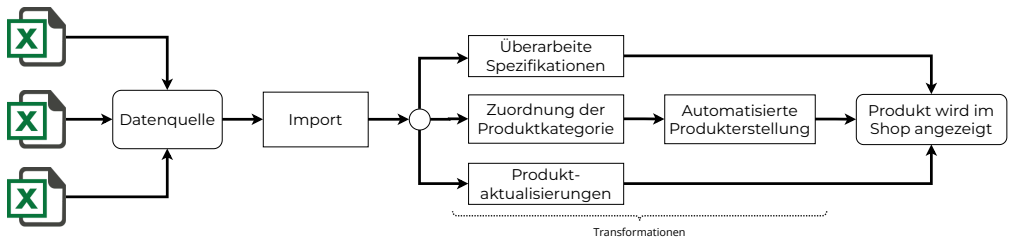


Abbildung 1.1: Ablauf der automatischen Produktdatenintegration für Online-Shop-Betreiber im Marktplatz von Galaxus. [8]

lauf des Integrationsprozesses im Online-Marktplatz Galaxus [8]. Dieser beginnt mit dem Einlesen von Excel-Dateien. Je nachdem, an welchen Stellen in den Produktdaten Änderungen vorgenommen wurden oder falls neue Produkte hinzugefügt wurden, sind weitere Schritte zwischen Einlesen und dem Einstellen in des DWH nötig. Sie *transformieren* die gelieferten Daten. Das Einstellen der Daten ins DWH wird bei Galaxus nicht explizit dargestellt. Es findet implizit vor der Anzeige der Produkte im Shop statt.

Die *Produktdaten* in den Dateien bestehen aus *Attributen*, durch die ein Produkt beschrieben wird. Zu jedem Attribut gehört ein *Attributname* und ein konkreter Wert. Diese Werte werden auch als *Attributinstanzen* bezeichnet. Die Kombination aller Attributnamen eines Produktkatalogs wird als *Schema* bezeichnet. Das genaue Schema der Daten wird durch die Marktplatzbetreiber anhand der angebotenen Produkte bestimmt, da sich je nach Anwendungsfall einzelne Produktattribute unterscheiden können. Weder das Dateiformat noch die Benennung der Attribute unterliegen weltweit geteilten und etablierten *Standards*. Vielmehr wählt jedes Unternehmen selbst, welches Format und welche Attribute genutzt werden, um die eigenen Produkte zu beschreiben bzw. Produktkataloge zu integrieren. In den meisten Fällen haben sich tabellarische Dateiformate sowohl bei den Zulieferern als auch bei Services und Marktplätzen durchgesetzt.

Obwohl verschiedene Standardformate (siehe Abschnitt 2.3) existieren, sind diese häufig nicht etabliert. Große Marktplatzbetreiber können aufgrund ihrer Größe und Marktposition auf eigenen Formaten und Attributen bestehen. Für KMU ist mangels Marktmacht das Durchsetzen eines eigenen Standardformates jedoch nicht möglich, wodurch die KMU auf die Produktdaten ihrer Zulieferer angewiesen sind. Teilweise müssen sie Webseiten nach den zugehörigen Daten durchsuchen, um an die notwendigen Produktinformationen zu gelangen (Web-Scraping, vgl. [11]).

Daher bleibt für KMU die Herausforderung, Produktdaten von Zulieferern so aufzubereiten, dass diese im Online-Shop in einheitlicher Form und guter Qualität angezeigt werden können. Je mehr Zulieferer bzw. Bezugsquellen ein Online-Shop-Betreiber hat, desto mehr unterschiedliche Eingangsformate müssen dabei in ein oder mehrere Zielformate überführt werden. Zusätzlich ändern einzelne Zulieferer bereits bekannte Attributnamen oder Formate ohne Ankündigung und in unregelmäßigen Abständen, wie Abschnitt 1.2 am Beispiel eines Online-Shops für Antikörper zeigt.

Üblicherweise werden die rohen Produktdaten der Zulieferer auch bei KMU über spezielle ETL-Prozesse aufbereitet. Diese Prozesse sind oft an einen einzelnen Zulieferer angepasst und können beliebig viele manuelle Schritte enthalten. Ein solcher manueller Schritt ist die *Zuordnung* der Attribute aus dem Produktkatalog der Zulieferer zu den Attributen der Online-Shop-Betreiber. In der Literatur wird das Problem, automatisch Zuordnungen zwischen zwei Schemata zu finden, üblicherweise als *Schema-* oder *Ontologie-Matching* bezeichnet.

Die Ermittlung der Zuordnungen zwischen dem *Eingangsschema* eines Zulieferers und dem *Zielschema* des Online-Shop-Betreibers wird teilweise bereits durch die Anwendung von Regeln und regulären Ausdrücken automatisiert. Eine solche Regel könnte beispielsweise lauten: Enthält ein Attributname aus dem Eingangsschema ein € oder $, ist eine Zuordnung zu dem Attributnamen `Preis` herzustellen. Des Weiteren werden durch KMU oft Thesauri verwendet, um synonym genutzte Attributnamen zu erkennen. Trotzdem erfolgen die Zuordnungen häufig manuell, denn die bestehenden Regeln müssen stets auf den individuellen Zulieferer angepasst werden. Darüber hinaus sind die genutzten Attributnamen der Zulieferer oftmals nicht in dem Thesaurus enthalten, da viele Zulieferer eine eigene Benennungsstrategie für die Attributnamen verwenden.

Bei der Durchführung des halbautomatischen ETL-Prozesses werden zahlreiche Metadaten erzeugt. Dazu gehören weitere Eingangsattributnamen oder Paare von Produktkatalogen vor und nach der Durchführung eines ETL-Prozesses. Diese Metadaten werden bisher von den KMU wenig oder gar nicht berücksichtigt. Diese Dissertation untersucht, inwiefern die Nutzung der Daten den Prozess zur Ermittlung der Zuordnungen ganz oder teilweise automatisieren kann.

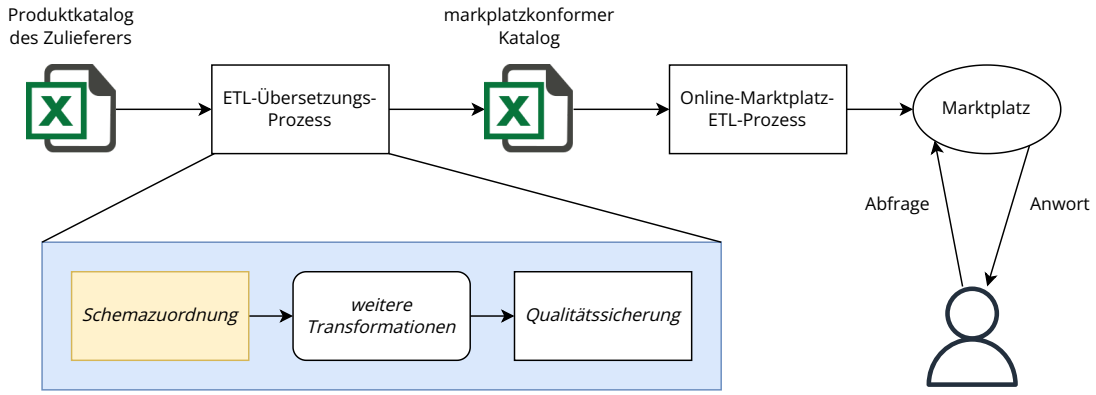


Abbildung 1.2: Übliche Katalogintegration im Online-Handel: Ein Zulieferer stellt Produktdaten in einem tabellarischen Format bereit, die über einen ETL-Prozess in ein marktplatzkonformes Format übersetzt werden, bevor die Daten über einen weiteren ETL-Prozess in den Marktplatz importiert werden.

1.2 Problemstellung

Im Online-Handel werden zwei aufeinanderfolgende ETL-Prozesse unterschieden. Wie in Abbildung 1.2 dargestellt, übersetzt im ersten ETL-Prozess ein Online-Shop-Betreiber die Produktkataloge seiner Zulieferer in einen marktplatzkonformen Produktkatalog. Dieser ETL-Übersetzungsprozess besteht im Wesentlichen aus den Schritten Schemazuordnung, darauf aufbauenden Transformationen und der anschließenden Qualitätssicherung. Der Schwerpunkt dieser Arbeit liegt auf der Automatisierung dieses ETL-Übersetzungsprozesses. Liegt der Produktkatalog dann marktplatzkonform vor, wird der nachgelagerten ETL-Prozess durch den Marktplatz durchgeführt. Ein typisches Beispiel für einen solchen Prozess ist der Prozess von Galaxus in Abbildung 1.1. Der nachgelagerte ETL-Prozess wird in dieser Ausarbeitung nicht weiter betrachtet.

Die wesentliche Herausforderung für Online-Shop-Betreiber selbst besteht also darin, die Daten in einem *marktplatzkonformen Produktkatalog* im vorgegebenen Format und Schema bereitzustellen. Der Marktplatz führt dann alle notwendigen Transformationen zur Darstellung der Produkte für den Kunden aus. Dazu können auch weitere Qualitätssicherungsmaßnahmen gehören, wie die Deduplizierung von Produkten oder die Normalisierung der verwendeten Sprache (Auflösung von Abkürzungen und Synonymen, einheitliche Darstellung von Einheiten usw.). In der Pra-

xis können diese Schritte fast vollständig durch einfache Funktionen wie Thesaurusabfragen, Suchen und Ersetzen oder reguläre Ausdrücke durchgeführt werden [10].

In diesem Abschnitt wird die zu lösende Problemstellung genauer erläutert. Zunächst erfolgt eine kurze Einführung in die Hintergründe der Forschungskooperation, die dieser Dissertation zugrunde liegt. Danach wird auf die speziellen Herausforderungen in der Forschungskooperation eingegangen, bevor die Problemstellung formalisiert wird.

Hintergründe zur Forschungskooperation

In der Forschungskooperation „*Die adaptive verteilte Datenfabrik – Hochperformante Dunkelverarbeitung unstrukturierter Labordaten*“, die sich aus der Fachhochschule Aachen, der antibodies-online GmbH[6] und der labforward GmbH[7] zusammensetzt, mussten Produktdaten verschiedener Zulieferer in den eigenen Online-Shop der antibodies-online GmbH integriert werden. Die antibodies-online GmbH ist ein KMU, das nach eigenen Angaben den weltweit größten Online-Shop für Forschungsreagenzien, wie z. B. Antikörper oder Proteine für die Arzneimittelforschung, betreibt. Der Shop beinhaltet über 4 Millionen Produkte von über 250 Zulieferern [12].

Die Bereitstellung qualitativ hochwertiger, einheitlich dargestellter Produktdaten ist ein entscheidender Wettbewerbsfaktor: Kunden – hier Wissenschaftler – treffen ihre Kaufentscheidung auf Basis der Produktdaten. Die Entscheidung für das richtige Produkt ist folglich von der Datenqualität und Genauigkeit der Produktbeschreibung abhängig. Eine falsche Produktbeschreibung, und damit fehlerhafte oder mehrfach durchgeführte Experimente, kann den Verlust des Kundenvertrauens bedeuten. Daher werden Kunden über 50 verschiedene Produktattribute zur Verfügung gestellt.

Die Aufbereitung der von den Zulieferern bereitgestellten Produktdaten ist ein komplexer Prozess, der geprägt ist von manuellen Eingriffen und mehrfachen Iterationen, um die gewünschte Qualität bei der Anzeige im Marktplatz zu erreichen. Der manuelle Aufwand zur Datenaufbereitung wurde anhand von Stichproben in Zusammenarbeit mit antibodies-online auf ca. vier Stunden pro Datei eines Zulieferers ermittelt. Dabei nahm die korrekte Ermittlung der Zuordnung zwischen dem Schema des Zulieferers und dem von antibodies-online ca. 80 % der Zeit in Anspruch.

[6] https://www.antikoerper-online.de/
[7] ehemals LabFolder GmbH, https://labforward.io/

Die Komplexität entsteht unter anderem durch die Vielzahl von Zulieferern und mangelnde Standardisierung untereinander. Zulieferer benutzen in der Regel tabellarische Formate, wie Excel-Arbeitsmappe (XLSX) oder comma-separated value (CSV), um mehrere Produkte gleichzeitig hinzuzufügen oder aktualisieren zu lassen. Daraus ergab sich als Ziel der Forschungskooperation die Aufgabe, die bestehenden Integrationsprozesse so zu erweitern, dass die Integration von Produktkatalogen mit unbekanntem tabellarischem Schema möglichst automatisiert erfolgen kann, ohne die Qualität der im Online-Shop angezeigten Daten negativ zu beeinflussen.

Die besonderen Herausforderungen bei der Integration von Produktkatalogen im Rahmen der Forschungskooperation sind im Folgenden aufgeführt:

- Jeder Zulieferer verwendet ein eigenes tabellarisches Schema, das sich in der verwendeten Terminologie und Organisation von denen anderer Zulieferer unterscheidet. Häufig ändert sich das Schema eines Zulieferers ohne Ankündigung.
- Obwohl alle Marktteilnehmer Produkte aus der gleichen Domäne beschreiben, hat jeder Teilnehmer ein eigenes Sprach- bzw. Konzeptverständnis, wodurch bei der Datenintegration bei antibodies-online zusätzlich ein spezielles Domänenwissen benötigt wird.
- Teilweise verändern Reseller das Datenformat oder Schema absichtlich, um die Herkunft ihrer Produkte zu verschleiern. So kann ein automatischer Vergleich nicht mehr stattfinden.

Überblick über den Integrationsprozess in der Forschungskooperation

Die Integration der Produktdaten beginnt, wie in Abbildung 1.3 zu sehen ist, mit der Bereitstellung der Daten durch einen Zulieferer. Je nach Zulieferer kann sie täglich, wöchentlich oder monatlich erfolgen. Dabei kann es sich um einfache Preisaktualisierungen für bereits vorhandene Produkte handeln oder um neue Produkte, die in dieser Form bisher nicht auf dem Weltmarkt vorhanden waren (z. B. Sars-Cov-2 Antikörper). Auch außerhalb der üblichen Zeitabstände kann es zu Datenlieferungen kommen, insbesondere wenn ein neuer Zulieferer in das Produktportfolio aufgenommen wird.

Die Produktdaten werden über verschiedene Kanäle bereitgestellt. Einige Lieferanten senden die Daten als Anhang per E-Mail, andere nutzen einen FTP-Server

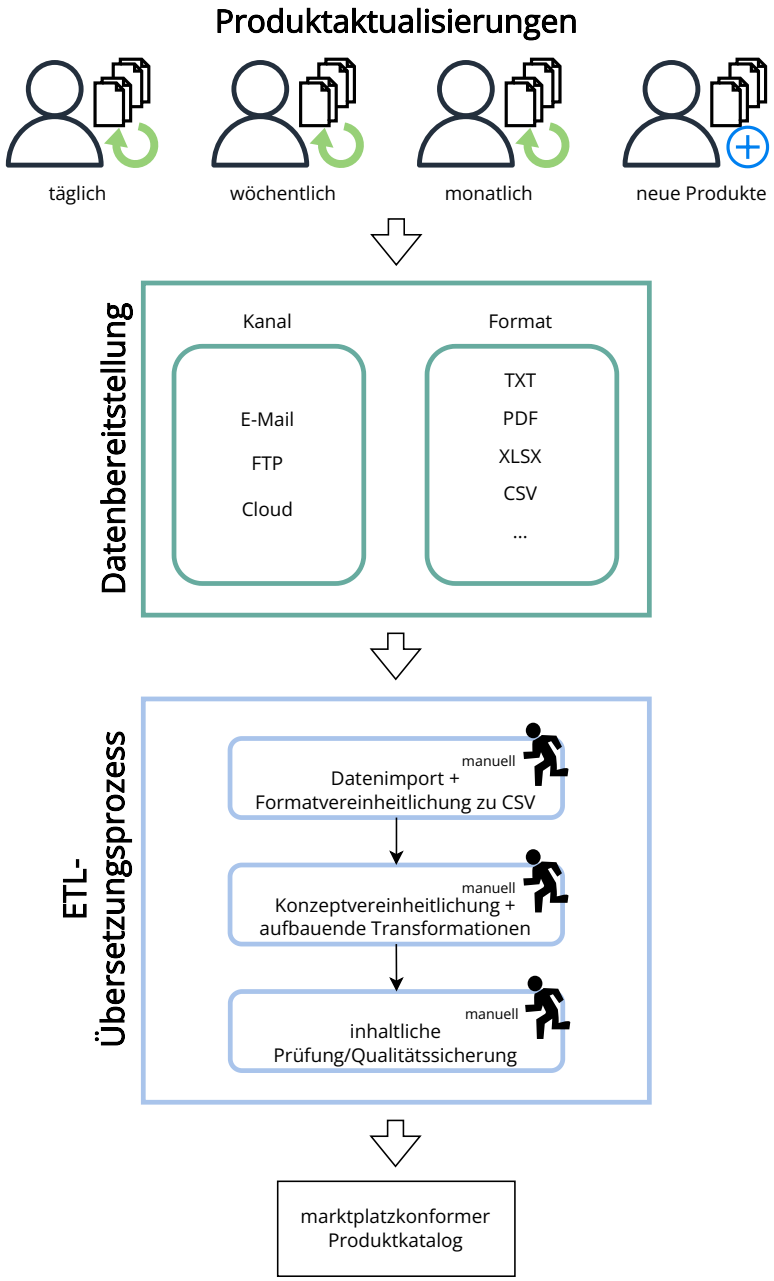


Abbildung 1.3: Vereinfachter Überblick über den Ablauf zur Integration gelieferter Produktdaten in den Online-Shop bei antibodies-online.

oder stellen die Daten über die Cloud (z. B. Dropbox[8], S3-Bucket[9]) zur Verfügung. Das Dateiformat der bereitgestellten Dokumente bzw. Produktkataloge hängt vom jeweiligen Zulieferer ab. Häufige Dateiformate sind XLSX und CSV. Selten werden auch Portable Document Format (PDF) und Word-Dokument (DOCX) genutzt, in denen jedoch nur einzelne Produkte enthalten sind anstelle eines Katalogs. Diese wurden zunächst in ein tabellarisches Format umgewandelt, um weiterverarbeitet werden zu können. Verfahren und Ansätze zur Extraktion der relevanten Produktinformationen aus PDF- und DOCX-Dateien finden sich in Schreiber [11].

In dieser Arbeit werden nur tabellarische Formate betrachtet, die immer mehrere Produkte, bis hin zum kompletten Produktkatalog eines Zulieferers mit mehreren tausend Produkten, enthalten. Die eingehenden Daten werden nach dem derzeitigen Vorgehen danach manuell so angepasst, dass nach dem Prozess ein marktplatzkonformer Katalog vorliegt. Dies findet in Abbildung 1.3 im ETL-Übersetzungsprozess statt. Das Dateiformat und das Schema des eingehenden Produktkatalogs hängen von den jeweils angebotenen Produkten und dem einzelnen Zulieferer ab. Aufgrund des Geschäftsmodells von antibodies-online findet eine Standardisierung nur schleichend statt. Einige große Anbieter haben zwar bereits ihr Schema an das von antibodies-online angepasst. Die Mehrzahl der Anbieter setzt jedoch auf eigene Schemata.

Nach der Vereinheitlichung der tabellarischen Formate in das CSV-Format beginnt die Konzeptvereinheitlichung. Der wesentliche Schritt der Konzeptvereinheitlichung ist die Schemazuordnung, bei dem die Attributnamen und -instanzen eines Zulieferers den Attributnamen des Zielschemas zugeordnet werden müssen. Derzeit erfolgen die Zuordnungen nur zu einem geringen Grad automatisiert.

Der einfachste Weg die Zuordnungen zu ermitteln, ist das Nutzen spezifischer Wörterbücher bzw. Thesauri. Diese können im verwendeten Werkzeug, Talend Studio[10], hinterlegt werden. Zusätzlich kann Talend bereits bekannte Zuordnungen und darauf aufbauende Transformationsschritte speichern. Diese Schritte erfolgen über speziell auf die Zulieferer und die jeweilige Zuordnung angepasste reguläre Ausdrücke. Diese sorgen dafür, dass die Produktdaten korrekt im Zielschema abgelegt werden. Dadurch müssen nicht alle Schritte des ETL-Prozesses angepasst werden, wenn Änderungen im Schema des gleichen Zulieferers auftreten. Dieses

[8] https://www.dropbox.com/
[9] https://aws.amazon.com/de/s3/
[10] https://www.talend.com/de/

Vorgehen funktioniert vor allem dann, wenn es nur eine begrenzte Anzahl an Zulieferern gibt, die mit bekannten und statischen Schemata arbeiten.

Die Schritte bis zur inhaltlichen Prüfung der durchgeführten Transformationen können daher auch in diesem Integrationsprozess nahezu automatisch ablaufen, wenn die Zuordnungen zwischen Eingangskatalog und Zielschema bekannt ist [13]. Sollten im Schritt der inhaltlichen Qualitätssicherung, die üblicherweise durch promovierte Biologen erfolgt, Fehler auffallen, muss der gesamte ETL-Übersetzungsprozess für den vollständigen Eingangskatalog erneut durchlaufen werden. Die Erkennung der Zuordnung ist daher der entscheidende Schritt im ETL-Prozess. Besonders herausfordernd ist die Ermittlung der Zuordnung, wenn ein Attributname aus dem Produktkatalog keinem oder mehreren Attributnamen aus dem Zielschema zugeordnet werden kann. Dies ist insbesondere dann der Fall, wenn Zulieferer mehrere Attribute hinter einem Namen zu verstecken versuchen oder unspezifische Attribute verwenden.

Da die Zuordnungen individuell auf jeden Zulieferer bzw. die verwendete Terminologie zugeschnitten sind, wird für jeden Zulieferer ein eigener ETL-Prozess benötigt. Bei über 250 Zulieferern, die jederzeit ihr Konzeptverständnis hinter den verwendeten Attributnamen ändern, Attribute umbenennen oder hinzufügen bzw. entfernen können, wird deutlich, welches Potenzial in einer höheren Automatisierung der Zuordnungserkennung steckt, da dieser Schritt in jedem vorhandenen ETL-Prozess an jede Änderung des jeweiligen Zulieferers angepasst werden muss.

Problemformalisierung

Ausgangspunkt der Katalogintegration ist ein tabellarischer Produktkatalog P^{in} eines Zulieferers. Dieser besteht aus den Attributnamen L_i^{in} mit $i \in [1, n]$, wobei n die Anzahl der Namen ist, und den Attributinstanzen $I^{in} = [I_{1,1}^{in}, \ldots, I_{m,n}^{in}]$, wobei $I_{j,i}^{in}$ einer einzelnen Instanz entspricht und m der Anzahl der Instanzen. P^{in} wird im weiteren Verlauf als *Eingangskatalog* bezeichnet. Damit der ETL-Übersetzungsprozess P^{in} in einen marktplatzkonformen Katalog P^{out} übersetzen kann, müssen die Zuordnungen zwischen den beiden Katalogen ermittelt werden. Die Attributnamen, die das *Zielschema* definieren, werden als L^{out} bezeichnet. Das Ziel eines Verfahrens ist folglich, jeder *Spalte* $C_i^{in} = [L_i^{in}, I_{1,i}^{in}, \ldots, I_{m,i}^{in}]$ aus P^{in} in Gleichung (1.1) alle relevanten Attributnamen aus L^{out} zuzuordnen.

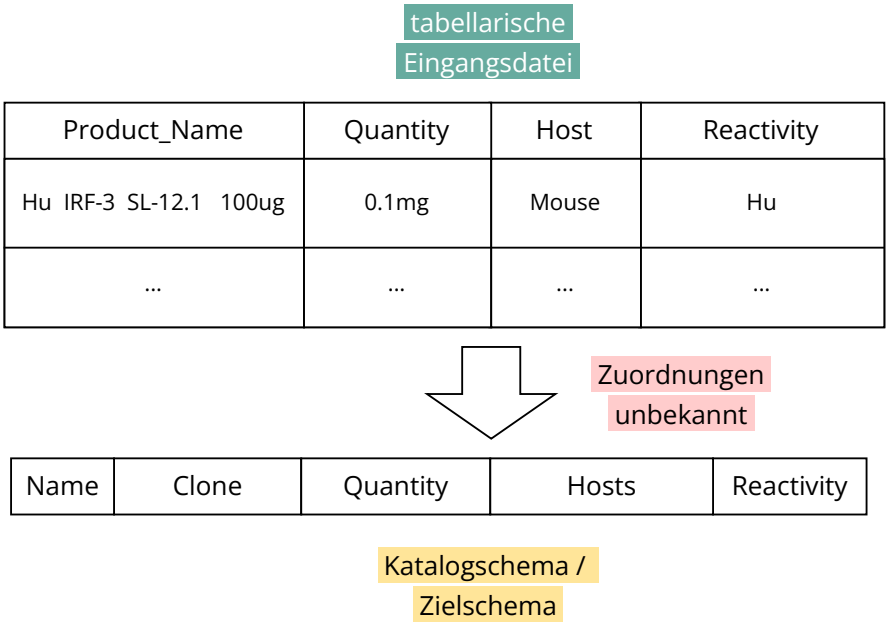


Abbildung 1.4: Ausgangssituation

$$P^{in} = \begin{bmatrix} L_1^{in} & L_2^{in} & \cdots & L_n^{in} \\ I_{1,1}^{in} & I_{1,2}^{in} & \cdots & I_{1,n}^{in} \\ \vdots & \vdots & \ddots & \vdots \\ I_{m,1}^{in} & I_{m,2}^{in} & \cdots & I_{m,n}^{in} \end{bmatrix} \tag{1.1}$$

Sonderfälle dieses Vorgehens lassen sich anhand von Abbildung 1.4 veranschaulichen. Die Spalte `Product_Name` aus dem Eingangskatalog muss mehreren Attributnamen aus dem Zielschema zugeordnet werden. Die *primäre Zuordnung* ist `Name`, da die Attributinstanz vollständig übertragen werden kann. Allerdings enthält die Instanz zu `Product_Name` weitere Informationen, die für andere Attribute aus dem Zielschema relevant sind. Darunter fällt beispielsweise die `Reactivity` (`Hu`) oder die `Quantity` (`100ug`). Das Verfahren muss folglich erkennen, dass weitere relevante Informationen enthalten sind. Ein Matching-System muss möglichst alle Kandidaten für eine Zuordnung erkennen und sie entsprechend ihrer *Konfidenz*, also einem Wert in $[0, 1]$, ordnen.

Je höher der Wert der Konfidenz ist, desto sicherer ist sich das System bei der Zuordnung. Dabei sollte die primäre Zuordnung immer eine höhere Konfidenz erzielen als andere relevante Zuordnungen. In Bezug auf das Beispiel aus Abbildung 1.4 könnte das System die Zuordnung `Product_Name -> Name` mit einer Konfidenz von 0.8 bewerten. Ist der Abstand zwischen den beiden Zielattributen mit den höchsten Konfidenzen kleiner als 0.1, werden die Zuordnungen als *mehrdeutig* bezeichnet. In diesem Fall lässt sich die primäre Zuordnung nicht klar bestimmen. Beide Zuordnungen sind jedoch relevant.

Bei besonders hohen Konfidenzwerten kann ein zuverlässiges System die Zuordnungen automatisch vornehmen. Anhand dieser Zuordnungen können im ETL-Prozess spezielle Transformationsschritte durchgeführt werden, die die Informationen aus den Attributinstanzen wie bisher extrahieren. Bei niedrigen Konfidenzen oder mehrdeutigen Zuordnungen sollte das System Empfehlungen aussprechen und die manuelle Arbeit durch Vorauswahl unterstützen.

Die primäre Zuordnung würde im oben genannten Beispiel automatisch vorgenommen werden. Erkennt das System gleichzeitig weitere relevante Zuordnungen (`Product_Name -> Quantity`, `Product_Name -> Reactivity`), könnten beide Zuordnungen gleichwertig mit einer Konfidenz von 0.4 bewertet werden. Die endgültige Entscheidung über diese beiden weiteren Zuordnungen müsste durch die geringere Konfidenz manuell durch einen Menschen erfolgen.

Attributnamen stellen, sowohl für das automatisierte als auch das manuelle Zuordnen, oftmals den ersten Ansatzpunkt dar, da ähnliche Attributnamen auf eine ähnliche Verwendung der Attribute in beiden Schemata hindeuten. Attributnamen sind in der Regel Abkürzungen oder zusammengesetzte Worte. Um deren Ähnlichkeit zu ermitteln, existieren verschiedene Verfahren. Dazu zählen unter anderem Ähnlichkeitsmetriken oder *Worttransformationen* in Vektorräume, die in Kapitel 4 erklärt werden. Es ist allerdings unklar, welche Kombination von Verfahren im Rahmen der Katalogintegration zu einem möglichst hohen Automatisierungsgrad führt.

Neben den Attributnamen bieten Attributinstanzen, beispielsweise ein konkreter Produktname, die Produktbeschreibung usw., einen Ansatzpunkt für die Zuordnungen. Attributinstanzen können die Qualität der Zuordnungen insbesondere dann verbessern, wenn entweder der Attributname ohne zusätzliche Informationen nicht auf den Inhalt der Instanzen schließen lässt oder die Instanzen zusätzliche Informationen enthalten, sodass mehrdeutige Zuordnungen möglich sind.

Darüber hinaus können die Ergebnisse der Zuordnung von den Attributinstanzentypen abhängen. Besteht beispielsweise eine Attributinstanz aus einer textuellen

Beschreibung, kann das automatische Bestimmen der Zuordnungen schwieriger sein, als wenn alle Attributinstanzen nur aus einem eingeschränkten Wertebereich stammen können. Typische Beispiele hierfür sind, bezogen auf die Produkte im Kooperationsprojekt, der Wirt eines Antikörpers oder die Lagertemperatur, die in der Regel bei −20 °C liegt.

Im Rahmen der Forschungskooperation standen Eingangskataloge, die daraus übersetzten marktplatzkonformen Kataloge, sowie in vielen Fällen zusätzlich vorhandene Thesauri für synonym genutzte Attributnamen aus den ETL-Übersetzungsprozessen zur Bestimmung der Zuordnungen zur Verfügung. Wie beschrieben, wurden diese Metadaten bisher nicht genutzt, um den manuellen Aufwand bei der Bestimmung der Zuordnungen zu reduzieren oder gänzlich zu eliminieren. Da die Anwendung von maschinellem Lernen (ML) bereits in vielen verwandten Bereichen, wie der Textklassifikation, erfolgreich war, könnte sich ML auch beim Lernen der Zuordnungen als hilfreich erweisen.

Für die Anwendung von ML lassen sich die Metadaten nutzen, da sie bereits manuell verifiziert wurden. Auf diese Weise lassen sich Datensätze aufbauen, anhand derer ein System lernen kann, Zuordnungen automatisch vorzunehmen oder Vorschläge für die Zuordnungen zu liefern. Die konkrete Datenbasis aus der Forschungskooperation bilden die jeweiligen Katalogpaare vor und nach dem ETL-Übersetzungsprozess aus Abbildung 1.2. Dabei werden die Produktkataloge der Zulieferer jeweils als P^{in} verwendet. Die marktplatzkonformen Kataloge P^{out} besitzen nach dem ETL-Übersetzungsprozess das Zielschema L^{out}.

Damit die Attributnamen oder -instanzen aus P^{in} in ML-Verfahren genutzt werden können, müssen die zu lernenden Eigenschaften in einen Vektor transformiert werden, der als Eingangswert für die ML-Verfahren genutzt werden kann. Diese Vektoren werden im weiteren Verlauf als *Feature-Vektoren* bezeichnet. Als Feature-Vektor können beispielsweise die bereits erwähnten Vektoren aus Worttransformationen genutzt werden. Ebenso können zusätzliche Informationen über die Attributinstanzen oder Attributnamen, wie deren Anzahl von Zeichen oder welche Buchstaben enthalten sind, in den Feature-Vektor eingebracht werden. Solche zusätzlichen Informationen werden bereits in existierenden Ansätzen wie der Zuordnung von Schema-Attributnamen durch Inhaltsanalyse [14] genutzt, jedoch ohne dabei gleichzeitig auf Worttransformationen zurückzugreifen.

Ziel dieser Arbeit ist die Entwicklung und Evaluierung von Verfahren zur Bestimmung von Zuordnungen zwischen den Schemata von Zulieferern und einem beliebigen vorgegebenen Marktplatzschema. In den Verfahren können die bisher unge-

nutzten Daten aus bereits durchgeführten Integrationsprozessen verwendet werden. Die Entwicklung und Evaluation der Verfahren erfolgt aufgrund der Forschungskooperation am Beispiel des Antikörpermarktes. Grundsätzlich sollten sich die entwickelten Verfahren allerdings auch auf andere Domänen und ähnliche Anwendungsfälle übertragen lassen.

1.3 Forschungsfragen

In dieser Dissertation wurde die folgende Leitfrage untersucht:

> Inwieweit können bisher nicht oder wenig genutzte Daten aus bereits durchgeführten Katalogintegrationen verwertet werden, um den höchstmöglichen Automatisierungsgrad im Teilprozess der Schemazuordnung insbesondere für KMU zu erreichen?

Diese Leitfrage lässt sich in die folgenden drei Forschungsfragen (FF) unterteilen, die jeweils unterschiedliche Aspekte der Schemazuordnung bei der Katalogintegration berücksichtigen und aufeinander aufbauen:

FF1: Wie lässt sich ausschließlich mit Informationen aus Attributnamen und vorhandenen Thesauri der höchste Automatisierungsgrad bei der Schemazuordnung erreichen?

FF2: Inwiefern wird die Schemazuordnung durch Informationen aus Attributinstanzen, die zusätzlich zu Attributnamen verwendet werden, verbessert?

FF3: Wie wirkt sich der Typ der Attributinstanzen auf die Qualität der Schemazuordnung und damit auf die Möglichkeit einer vollständigen Automatisierung aus?

Über die Beantwortung dieser FF lässt sich anschließend die Leitfrage differenziert beantworten.

1.4 Beiträge der Arbeit

In den folgenden Abschnitten werden die Methodik und die erreichten Ergebnisse gegliedert nach den FF dargestellt. Alle entwickelten Verfahren wurden im Rahmen der Dissertation in der Programmiersprache Python implementiert.

FF1

Zur Beantwortung von FF1 wurden Random Forests, SVMs und NNs als ML-Verfahren genutzt, die in Kapitel 3 detailliert vorgestellt werden. Diese wurden kombiniert mit verschiedenen Verfahren zur Transformation von Attributnamen in Feature-Vektoren (siehe Kapitel 4). Das Training erfolgte anhand der verfügbaren Thesauri von antibodies-online so, dass sie zu einem beliebigen Eingangsattributnamen den zugehörigen Zielattributnamen aus L^{out} vorhersagen.

Die ML-Verfahren wurden mit den folgenden drei Referenzverfahren, die ohne ML auskommen, verglichen:

1. **Einfaches Referenzverfahren:** Dieses trifft die Zuordnung nur über eine Ähnlichkeitsmetrik zwischen den Eingangsattributnamen und den Zielattributnamen.

2. **Pragmatisches Referenzverfahren:** Dieses trifft die Zuordnung über die größte Ähnlichkeit mit bekannten Eingangsattributnamen aus den Thesauri ermittelt.

3. **COMA:** COMA ermittelt Zuordnungen über Heuristiken und Ähnlichkeitsmetriken über den Thesaurus (siehe Abschnitt 9.1).

Die Referenzverfahren sind dagegen nicht auf das Bilden von Feature-Vektoren angewiesen. Alle Verfahren wurden anhand der üblichen Klassifikationsmetriken Precision, Recall und F_1-Score bewertet. Details zu diesen Metriken finden sich in Abschnitt 3.6. Sämtliche Verfahren verwenden ausschließlich Informationen aus Attributnamen, um die entsprechenden Zuordnungen zwischen Eingangs- und Zielschema zu bestimmen. Ziel war die Bestimmung der primären Zuordnung, da ohne zusätzliche Informationen keine mehrdeutigen Zuordnungen erkannt werden können.

Die Verfahren wurden in vier Szenarien miteinander verglichen. In allen Szenarien schnitten die ML-Verfahren besser ab als das erste, einfache Referenzverfahren und COMA. Die besten Ergebnisse erzielten jeweils diejenigen ML-Verfahren, die One-Hot (OH)- oder term frequency-inverse document frequency (TF-IDF)-Vektoren verwendeten (vgl. Kapitel 6). Dabei zeigte sich weiterhin, dass die Transformation in einen Feature-Vektor einen deutlich größeren Einfluss besaß, als das genutzte ML-Verfahren. In den Szenarien schwankte die erreichte Leistung der besten Verfahren zwischen einer Precision von 0.7 und 0.92, einem Recall von 0.65 und 0.92 sowie einem F_1-Score von 0.65 und 0.9. Gleichzeitig wurde in der Evaluation gezeigt, dass

diese Ergebnisse in Abhängigkeit von Trainings- und Testdatensatz stark schwanken können. Dies lässt darauf schließen, dass der Inhalt des Trainingsdatensatzes durch die ML-Verfahren teilweise auswendig gelernt wurde.

Aus diesem Grund wurde eine weitere Auswertung der ML-Verfahren im produktiven Betrieb durchgeführt. Dazu wurde ein Microservice entwickelt, der als Empfehlungssystem in die Werkzeuge zur manuellen Bearbeitung des Integrationsprozesses von antibodies-online eingebunden wurde. Dieser Service sollte die drei Zuordnungen mit der höchsten Konfidenz bereitstellen. Dadurch konnten die drei vorgeschlagenen Zuordnungen direkt mit der tatsächlichen Zuordnung verglichen werden. Die beste Kombination von ML- und Transformationsverfahren schlug in 90% der Fälle die primäre Zuordnung korrekt vor. Eine vollständige Automatisierung ließ sich jedoch nicht erreichen.

FF2

Im nächsten Schritt wurde das Attribut Label Ranking (ALR)-Verfahren entwickelt, welches zusätzlich Verbesserungen durch die Verwendung von Attributinstanzen erzielen konnte. ALR besteht aus zwei Schritten. Zuerst werden Attributnamen unabhängig von ihren jeweiligen Attributinstanzen über die oben genannten Verfahren in einen Feature-Vektor transformiert und als Eingangswerte für ein neuronales Netz (NN) verwendet, das die Zuordnung pro Attributinstanz vorhersagt. Im zweiten Schritt werden die Vorhersagen der jeweiligen Zuordnungen anhand ihrer Häufigkeit gewichtet. Das verwendete NN wurde so trainiert, dass es in der Lage ist, mehrere Zielattribute aus L^{out} gleichzeitig vorherzusagen, anstatt wie in FF1 nur eine einzelne Zuordnung vorherzusagen. Durch dieses Vorgehen können neben der primären Zuordnung weitere Zuordnungen identifiziert werden.

In der Evaluation zeigte sich, dass ALR sowohl primäre Zuordnungen zuverlässig vorhersagen, als auch mehrdeutige Zuordnungen erkennen kann. Für die primären Zuordnungen wurden, wie in FF1, die üblichen Metriken Precision, Recall und F_1-Score genutzt. Im Vergleich zu FF1 lag die Vorhersage der primären Zuordnung bzgl. der Precision zuverlässig über 0.81 und erreichte ein Maximum von 0.89. Recall und F_1-Score fielen mit Werten von bis zu 0.8 und 0.84 etwas geringer aus als die Precision. Insgesamt war die Bandbreite der Ergebnisse deutlich geringer als bei der ausschließlichen Verwendung von Attributnamen. Somit wurden die Ergebnisse durch die Hinzunahme der Attributinstanzen unempfindlicher gegenüber Schwan-

kungen in den verwendeten Datensätzen, ohne dass die Ergebnisse in Bezug auf die primäre Zuordnung signifikant schlechter wurden.

Mit ALR ließen sich in den meisten Fällen weitere Zuordnungen vorhersagen. Für die Bewertung der Ergebnisse wurden die Metriken Area Under the Receiver Operating Characteristic Curve (AUC) und Label Ranking Average Precision (LRAP) genutzt (siehe Abschnitt 3.6.3). Diese erlauben neben der Betrachtung der primären Zuordnung auch die Berücksichtigung der weiteren erkannten Zuordnungen. In der Evaluation wurde ein AUC-Wert von bis zu 0.75 und ein LRAP von bis zu 0.86 erreicht. Dadurch ließ sich auch ALR als Empfehlungssystem einsetzen. Für Zuordnungen bei denen ALR mehrere gleichwertige Zuordnungen erkennt, müssen in letzter Instanz Mitarbeitende mit ihrer Expertise entscheiden. Trotzdem kann ALR eine deutliche Hilfestellung bieten.

Diese Ergebnisse zeigen, dass ALR in den meisten Fällen die entsprechenden Zuordnungen ermitteln kann, allerdings nicht in allen Fällen. Trotzdem stellen die erzielten Ergebnisse eine deutliche Verbesserung gegeüber Verfahren aus vergleichbaren Anwendungsfällen dar. Im Vergleich zur bereits erwähnten Zuordnung von Schema-Attributnamen durch Inhaltsanalyse [14] wurden um bis zu 0.5 höhere Werte für Precision, Recall und F_1-Score erzielt.

FF3

Um die Resultate aus FF2 genauer zu verstehen, wurden die Resultate anhand der Attributinstanzen in die Typen *Werteliste* und *unstrukturierter Text* unterteilt. Unter dem Typ Werteliste wurden zusätzlich die Typen Identifikationsnummern (IDs), numerische Werte und Uniform Resource Locators (URLs) zusammengefasst.

Im direkten Vergleich der Zuordnungen zwischen den Attributtypen zeigte sich eindeutig, dass Zuordnungen aus Wertelisten deutlich einfacher erkannt werden können als solche aus Textattributen. Bei Attributen des Typs Werteliste wurde bei der Vorhersage der primären Zuordnung eine Precision von bis zu 0.98, ein Recall von bis zu 0.84 und ein F_1-Score von bis zu 0.87 erreicht. Damit lag ALR in diesem Fall in etwa auf menschlichem Niveau. Bezüglich textbasierten Attributen lag die Erkennung der primären Zuordnung mit einer Precision von 0.9, einem Recall von bis zu 0.79 und einem F_1-Score von bis zu 0.84 leicht unter menschlichem Niveau.

Diese Ergebnisse zeigen, dass mit ALR die Zuordnung von Attributen der Kategorien IDs, Wertelisten, numerische Werte und URLs automatisiert erfolgen kann und lediglich eine menschliche Qualitätskontrolle erfolgen muss. In Bezug auf text-

basierte Attribute sollte ALR nur als Empfehlungssystem eingesetzt werden, da in diesem Attributtyp häufig mehrdeutige Zuordnungen enthalten sind, die wie beschrieben, schlechter erkannt werden können als die primären Zuordnungen. Dennoch sorgt auch die Nutzung des Empfehlungssystems für Zeitersparnis und eine Arbeitserleichterung bei der Integration von Produktdaten.

1.5 Publikationen

Während der Durchführung des Promotionsprojektes war der Autor an folgenden Publikationen, aufgeführt in der Reihenfolge ihres Erscheinens, beteiligt:

[S1] O. Schmidts et al., „Multi-pedestrian tracking by moving Bluetooth-LE beacons and stationary receivers“, in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 18-21 September 2017, Sapporo, Japan*, 2017, Seiten 1–4. Adresse: https://juser.fz-juelich.de/record/842817/files/178.pdf

[S2] O. Schmidts et al., „Continuously evaluated research projects in collaborative decoupled environments“, in *Proceedings of the 5th International Workshop on Software Engineering Research and Industrial Practice*, Gothenburg Sweden: ACM, 29. Mai 2018, Seiten 2–9, ISBN: 978-1-4503-5744-9. DOI: 10.1145/3195546.3195549. Adresse: https://dl.acm.org/doi/10.1145/3195546.3195549 (besucht am 04. 03. 2023)

[S3] O. Schmidts et al., „Schema Matching with Frequent Changes on Semi-Structured Input Files: A Machine Learning Approach on Biological Product Data:“ In *Proceedings of the 21st International Conference on Enterprise Information Systems*, Heraklion, Crete, Greece: SCITEPRESS - Science and Technology Publications, 2019, Seiten 208–215, ISBN: 978-989-758-372-8. DOI: 10.5220/0007723602080215. Adresse: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007723602080215 (besucht am 24. 07. 2020)

[S4] I. Siebigteroth et al., „A Study on Improving Corpus Creation by Pair Annotation“, in *Proceedings of the Poster Session of the 2nd Conference on Language, Data and Knowledge (LDK-PS 2019)*, 2019, Seiten 40–44. Adresse: http://ceur-ws.org/Vol-2402/paper8.pdf

[S5] O. Schmidts et al., „Catalog Integration of Low-quality Product Data by Attribute Label Ranking:“ In *Proceedings of the 9th International Conference on Data Science, Technology and Applications*, Lieusaint - Paris, France: SCITEPRESS - Science and Technology Publications, 2020, Seiten 90–101, ISBN: 978-989-758-440-4. DOI: 10.5220/0009831000900101. Adresse: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0009831000900101 (besucht am 24. 07. 2020)
(nominiert für den „Best Industrial Paper Award“)

[S6] M. Sildatke et al., „Automated Software Quality Monitoring in Research Collaboration Projects“, in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, Seoul Republic of Korea: ACM, 27. Juni 2020, Seiten 603–610, ISBN: 978-1-4503-7963-2. DOI: 10.1145/3387940.3391478. Adresse: https://dl.acm.org/doi/10.1145/3387940.3391478 (besucht am 04. 03. 2023)
(Fortsetzung der Arbeiten aus [S2] in einem anderen Kooperationsprojekt)

[S7] O. Schmidts et al., „Catalog Integration of Heterogeneous and Volatile Product Data“, in *Data Management Technologies and Applications*, S. Hammoudi et al., Herausgeber, Band 1446, Cham: Springer International Publishing, 2021, Seiten 134–153, ISBN: 978-3-030-83013-7 978-3-030-83014-4. DOI: 10.1007/978-3-030-83014-4_7. Adresse: https://link.springer.com/10.1007/978-3-030-83014-4_7 (besucht am 04. 03. 2023)
(Erweiterte Version von [S5] als Buchteil aufgrund der Nominierung)

[S8] P. Kohl et al., „STAMP 4 NLP – An Agile Framework for Rapid Quality-Driven NLP Applications Development“, in *Quality of Information and Communications Technology*, A. C. R. Paiva et al., Herausgeber, Band 1439, Cham: Springer International Publishing, 2021, Seiten 156–166, ISBN: 978-3-030-85346-4 978-3-030-85347-1. DOI: 10.1007/978-3-030-85347-1_12. Adresse: https://link.springer.com/10.1007/978-3-030-85347-1_12 (besucht am 04. 03. 2023)

[S9] A. Büsgen et al., „Exploratory Analysis of Chat-based Black Market Profiles with Natural Language Processing:“ In *Proceedings of the 11th International Conference on Data Science, Technology and Applications*, Lisbon, Portugal: SCITEPRESS - Science and Technology Publications, 2022, Seiten 83–94,

ISBN: 978-989-758-583-8. DOI: 10.5220/0011271400003269. Adresse: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0011271400003269 (besucht am 04. 03. 2023)
(eine der besten Publikationen der Konferenz)

[S10] D. Thewes et al., „Patient feature substitution with synthetic data for developing open corpora“, German Medical Science GMS Publishing House, 19. Aug. 2022, DocAbstr. 2. DOI: 10.3205/22gmds026

[S11] A. Büsgen et al., „From Cracked Accounts to Fake IDs - User Profiling on German Telegram Black Market Channels“, in to appear
(Erweiterung zu [S9] als Buchteil mit den besten Publikationen der Konferenz)

Die meisten dieser Veröffentlichungen haben direkten oder indirekten Bezug zur Dissertation. Insbesondere wurde in [S3] der Grundstein für Kapitel 5 der Dissertation gelegt, indem mehrere Ansätze zur Katalogintegration anhand von Attributnamen verglichen wurden. Im Zuge der Verfassung der Dissertation wurden die Verfahren aus [S3] wesentlich erweitert und zusätzlich nach Python migriert. Die Software-Architektur aus [S2] wurde für die Validierung der Verfahren im industriellen Einsatz in Kapitel 6 verwendet.

Ähnlich wurden in [S5] und [S7] die Ansätze aus Kapitel 7 zur gleichzeitigen Verwendung von Attributnamen und Attributinstanzen entwickelt. Der Fokus von [S5] lag auf den konzeptionellen Grundlagen sowie der Datenbeschaffung, während die zuvor entwickelten Verfahren in [S7] weiter analysiert und verbessert wurden.

Die Veröffentlichungen [S1, S4, S6, S9, S10, S11] sind weitgehend unabhängig vom Promotionsprojekt entstanden, haben jedoch mit diesem gemeinsam,

- dass ML-Verfahren für die zu lösenden Problemstellungen eingesetzt werden [S1, S9, S11],
- Daten zu einem Goldstandard bzw. Korpus aufbereitet werden [S4, S10] oder
- Vorgehensweisen zur Verwendung und Entwicklung von ML-Verfahren in Kooperationsprojekten vorgestellt werden [S6, S8].

Alle Veröffentlichungen wurden arbeitsteilig von mehreren Autoren erstellt, wobei der eigene Anteil von der Beteiligung an gemeinsamen Diskussionen mit den Koautoren [S6, S10] über die Mitwirkung bei der Konzeption des Lösungsansatzes, die Implementierung und Durchführung der Experimente [S4, S8, S9, S11] bis zur

federführenden Entwicklung der Idee und des Lösungsansatzes, der Implementierung und der Formulierung der Veröffentlichung [S1, S2, S3, S5, S7] reichte.

1.6 Aufbau der Arbeit

In Kapitel 2 werden zunächst die Hintergründe und Herausforderungen der Katalogintegration, von der theoretischen Betrachtung über aktuelle Standards, bis hin zu Produktdatenkategorien sowie der Bezug zwischen Schema-Matching und der Katalogintegration vorgestellt.

In Kapitel 3 folgen die Grundlagen von ML-Systemen und -Verfahren, die in der Dissertation genutzt werden. Diese werden anhand der Schemazuordnung für Produktdaten erklärt. Ebenso werden die Metriken eingeführt, mit denen sich ML- und Empfehlungssysteme bewerten lassen.

Sowohl Attributnamen als auch Attributinstanzen bestehen häufig aus Worten bzw. Texten. Diese müssen in einen Feature-Vektor transformiert werden, damit ML-Systeme deren Eigenschaften lernen können. Verfahren, die diese Transformation ermöglichen, werden in Kapitel 4 vorgestellt.

In Kapitel 5 werden die im Abschnitt genannten Ansätze zur Katalogintegration mittels Attributnamen entwickelt und in Kapitel 6 miteinander verglichen und bewertet, um FF1 zu beantworten. Anhand der gewonnenen Erkenntnisse wird in Kapitel 7 das ALR als eigenständiges Verfahren entwickelt, um die weiteren Forschungsfragen zu beantworten. Die Bewertung der Ergebnisse von ALR folgt in Kapitel 8.

Bevor Kapitel 10 diese Ausarbeitung mit der Beantwortung der Forschungsfragen, einer Zusammenfassung und dem Ausblick abschließt, werden in Kapitel 9 die erreichten Ergebnisse bei der Beantwortung der Forschungsfragen in den wissenschaftlichen Kontext eingeordnet und ein Überblick über verwandte Arbeiten und Ansätze gegeben.

Kapitel 2

Grundlagen zur Katalogintegration

Diese Kapitel widmet sich den besonderen Herausforderungen der Katalogintegration. Zunächst werden die theoretischen Hintergründe der allgemeinen Datenintegration aufgezeigt, bevor die Katalogintegration als spezielles Schema-Matching-Problem eingeführt wird. Danach werden gängige Standards für den Austausch von Produktdaten vorgestellt, die entweder durch Konsortien entwickelt wurden oder durch die Marktpostion eines Unternehmens verbreitet wurden. Abschließend werden die Attributinstanzen aus Produktkatalogen in verschiedene Typen eingeordnet und die damit verbundenen Herausforderungen beschrieben.

2.1 Theoretische Perspektive

Die *Datenintegration* ist ein Prozess, der das Ziel hat, Daten aus mehreren heterogenen, unabhängigen Quellen zu kombinieren, um letztendlich einen einheitlichen Zugang bzw. eine einheitliche Ansicht der Daten zu erhalten [15, 16].

Die Herausforderungen im Umgang mit mehreren Datenquellen liegen nach Doan et al. [15] in folgenden Eigenschaften:

- **Anzahl der Datenquellen:** Oftmals ist bereits die Integration von lediglich zwei Datenquellen eine Herausforderung. Mit steigender Anzahl von Datenquellen steigt die Komplexität bei der Integration.

- **Heterogenität:** In der Regel werden Datenquellen unabhängig voneinander und zu verschiedenen Zwecken entwickelt. Dadurch liegen den einzelnen Quellen verschiedene semi- und unstrukturierte Dateiformate (Extensible Markup Language (XML), CSV, Text usw.) und Schemata zugrunde. Üblicherweise ver-

wendet jede Quelle ein eigenes Datenformat und ein eigenes Datenschema, auch dann, wenn alle Quellen die gleiche Fachlichkeit abbilden.

- **Autonomie:** Die zu nutzenden Datenquellen liegen nicht zwangsweise in der Verantwortung einer einzelnen Person, Abteilung oder Organisation. Dadurch kann der Zugriff auf die Daten nur eingeschränkt möglich sein (zeitlich, keine Rohdaten, etc.). Auch müssen besonders schutzwürdige Daten im Integrationsprozess berücksichtigt werden. Darüber hinaus können die Verantwortlichen der Datenquellen jederzeit Formate, Schemata oder Zugriffswege ändern, ohne die Nutzer der Datenquellen zu informieren.

Die genannten Herausforderungen gelten auch bei der Produktintegration im Online-Handel: je mehr Zulieferer bzw. Hersteller Produkte auf einem Marktplatz anbieten wollen, desto schwieriger wird die Datenintegration für den Plattformbetreiber. Wie bereits in Abschnitt 1.2 beschrieben, verwenden viele Zulieferer eigene Formate, die sich beliebig und ohne Einfluss des Marktplatzbetreibers ändern können (Autonomie), oder individuelle Schemata (Heterogenität).

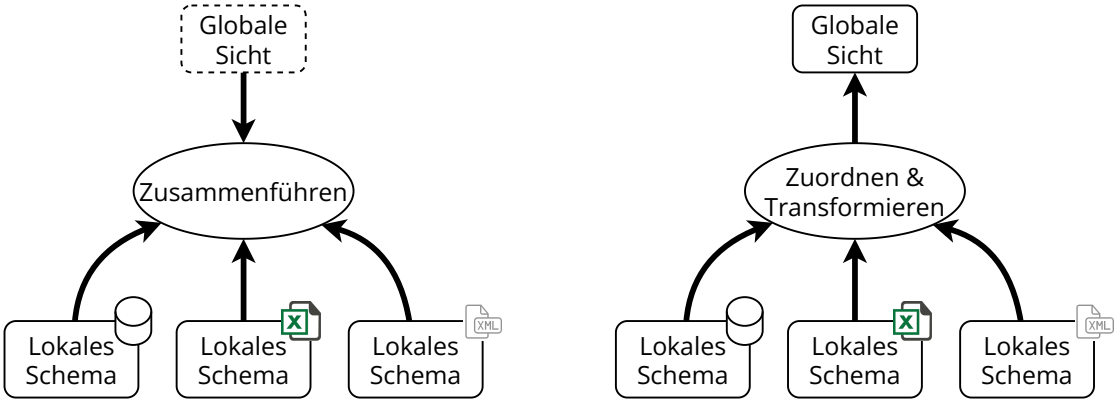
Die Kernaufgabe im Online-Handel ist, Endnutzern Produktdaten von verschiedenen Zulieferern mit jeweils eigenen Schemata über ein (globales) Zielschema eines Online-Shops zugänglich zu machen. Ziel eines Datenintegrationssystems ist, die Zuordnungen zwischen den Datenquellen und dem Zielschema zu ermitteln bzw. während der Integration abzubilden [17]. Lenzerini [16] beschreibt ein allgemeines Datenintegrationssystem I über das Tripel (G, S, M). Dabei sind

- G das globale Schema (Ziel). Diese wird über eine Sprache L_G mit zugehörigem Alphabet A_G ausgedrückt wird, wobei das Alphabet ein Symbol für jedes Element (Attribut) von G enthält.

- S ein lokales Schema der Datenquelle. Diese wird über die zugehörige Sprache L_S mit dem Alphabet A_S ausgedrückt wird, wobei das Alphabet ein Symbol für jedes Element (Attribut) von S enthält.

- M eine Zuordnung zwischen G und S beschreibt, wobei sich die Zuordnung aus einer Menge von Abfragen Q über G und S zusammensetzt.

In den folgenden Abschnitten werden theoretische Grundlagen vorgestellt, wie anhand dieser Definition das Ermitteln von Zuordnungen zwischen Quell- und Zielschema im Allgemeinen bis hin zur Integration von Produktkatalogen erfolgen kann.



(a) global-as-view (GAV) *(b) local-as-view (LAV)*

Abbildung 2.1: Vergleich der grundsätzlichen Abläufe des global-as-view (GAV)- und local-as-view (LAV)-Ansatzes [18]

2.1.1 global-as-view (GAV)

Der GAV-Ansatz verbindet die Zuordnung M für jedes Element g aus G mit einer Abfrage q_S über das Quellschema S, um eine Information aus dem Quellschema über das globale Zielschema darzustellen. Das bedeutet, die Sprache $L_{M,G}$ erlaubt ausschließlich Ausdrücke, die aus einem Symbol aus dem Alphabet A_G bestehen. Aus Sicht der Modellierung wird beim GAV-Ansatz der Inhalt jedes Elements g durch eine Abfrage q_S auf den Quellschemata beschrieben. [16]

Anders ausgedrückt, verknüpft ein GAV basiertes System die Datenquellen durch fest definierte Abfragen direkt mit dem Zielschema (vgl. Abbildung 2.1a). Dieser Ansatz bietet sich daher nach Lenzerini [16] insbesondere bei standardisierten Datenquellen an, da eine neue Datenquelle die Abfragen auf bereits vorhandenen Attributen des globalen Zielschemas negativ beeinflussen kann.

2.1.2 local-as-view (LAV)

Im Gegensatz zum GAV-Ansatz definiert der LAV-Ansatz die Zuordnung über die Verbindung jedes Elements s aus dem Quellschema S mit einer Abfrage q_G über das Zielschema G. Somit erlaubt die Sprache $L_{M,S}$ nur Ausdrücke, die aus einem Symbol aus dem Alphabet A_S bestehen. [16]

Veranschaulicht bedeutet diese Definition, dass die Beziehungen der einzelnen Quellschemata über Zuordnungen und Transformationen explizit abgebildet werden, wie in Abbildung 2.1b dargestellt. Bei LAV basierten Systemen wurde das Ziel-

schema meist unabhängig von den Datenquellen entwickelt. Um Nutzern des Zielschemas Abfragen zu ermöglichen, müssen die Quellschemata in der Folge einzeln eingebunden werden.

Daraus ergibt sich die Stärke von LAV-Ansätzen bei der Datenintegration: Schemazuordnungen sind sehr flexibel erweiterbar, solange das Zielschema stabil und etabliert ist. Das ist in der Regel der Fall, wenn das globale Schema bereits auf einem Unternehmens-Modell oder einer Ontologie basiert. Wird eine neue Datenquelle hinzugefügt, muss ausschließlich die Zuordnung des neuen Quellschemas auf das globale Schema mit den zugehörigen Rahmenbedingungen angepasst werden [16].

2.1.3 global-as-view vs. local-as-view im Online-Handel

Unabhängig davon, welcher Ansatz für die konkrete Datenintegration verwendet wird, muss eine Abfrage über das global definierte Zielschema möglich sein. Ein typisches Beispiel ist die Produktanzeige in einem Online-Shop. Möchte ein Käufer ein bestimmtes Produkt finden, müssen sich alle Anfragen an die Produktdatenbank über das Zielschema ausdrücken lassen. Dabei ist zunächst unerheblich, ob zuvor fest definierte Transformationen und Zuordnungen stattgefunden haben oder ob sich das Zielschema direkt aus mehreren Datenquellen ableiten lässt.

Auf einem Online-Marktplatz, bei dem jeder Zulieferer Daten in einem individuellen Format und Schema liefert, bietet sich ein LAV-Ansatz eindeutig an, da

1. ein etabliertes Zielschema existiert,
2. neue Datenquellen (Zulieferer) leicht hinzugefügt werden können und
3. Schemaänderungen eines einzelnen Zulieferers ausschließlich dessen Zuordnungen betreffen.

2.1.4 Ontologie- und Schema-Matching

Ontologien beschreiben in der Regel eine spezielle Domäne oder eine Spezifikation mithilfe eines festgelegten Vokabulars. Je nach Spezifikation kann eine Ontologie mehrere Datenmodelle, bestehend aus Begrifflichkeiten, Kategorien oder Datenbankschemata, beinhalten [19]. Die Ontologien werden dazu in eigenen Sprachen wie zum Beispiel Web Ontology Language (OWL) beschrieben. Dagegen beinhalten Schemata lediglich die formale Definition oder Struktur eines Artefakts wie einer

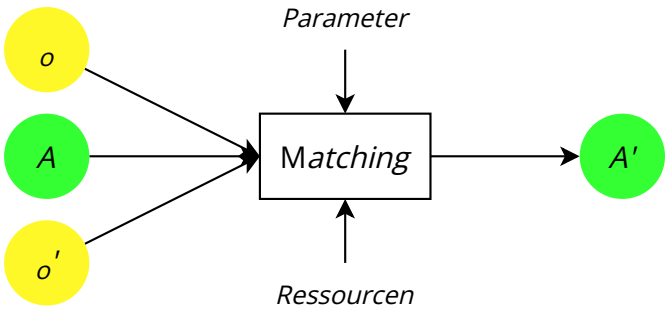


Abbildung 2.2: Ontologie-Matching-Prozess nach Shvaiko und Euzenat [22]

XML-Datei, einer Datenbankstruktur oder eines Produktkatalogs. Auch die Datenmodelle von Ontologien werden über Schemata definiert [20]. Einen detaillierten Überblick zu Ontologie-Arten, ihrer Ausdrucksstärke und Ontologie-Sprachen liefern Euzenat und Shvaiko [21].

Ontologie- und Schema-Matching beschreibt die Problemstellung, Zuordnungen zwischen zwei oder mehr verschiedenen Schemata zu finden. Die Zuordnung drückt aus, dass die einander zugeordneten Elemente die gleiche Notation oder Information enthalten [20]. Da Ontologien und Schemata eine Domäne durch Begriffe aus einem festen Vokabular beschreiben, ähneln sich die Lösungsansätze, sodass häufig nicht zwischen Ontologie- und Schema-Matching unterschieden wird [19]. Der größte Unterschied ist, dass im Ontologie-Matching zusätzliches, explizit definiertes Wissen aus den verschiedenen Ontologien ausgenutzt werden kann, während im Schema-Matching hauptsächlich versucht wird, die eigentliche Bedeutung der Begriffe hinter dem Schema abzuleiten [22].

Schema-Matching-Systeme können nach einer von Euzenat und Shvaiko [23] aufgestellten Methodik, dargestellt in Abbildung 2.3, kategorisiert werden. Zunächst wird auf zwei Ebenen unterscheiden:

1. **Granularität/Interpretation der Daten:** Diese Kategorie berücksichtigt, auf welcher Ebene ein Ansatz arbeitet (z. B. elementweise oder die Gesamtstruktur berücksichtigend) und wie die Informationen durch ein Schema-Matching-System grundsätzlichen interpretiert werden. Zur Unterscheidung werden die folgenden Merkmale genutzt:
    - **Element-basiert oder Struktur-basiert:** Element-basierte Techniken ermitteln die Zuordnungen durch die Analyse von Attributinstanzen. Dabei werden mögliche Beziehungen zu anderen Attributen ignoriert. Im Ge-

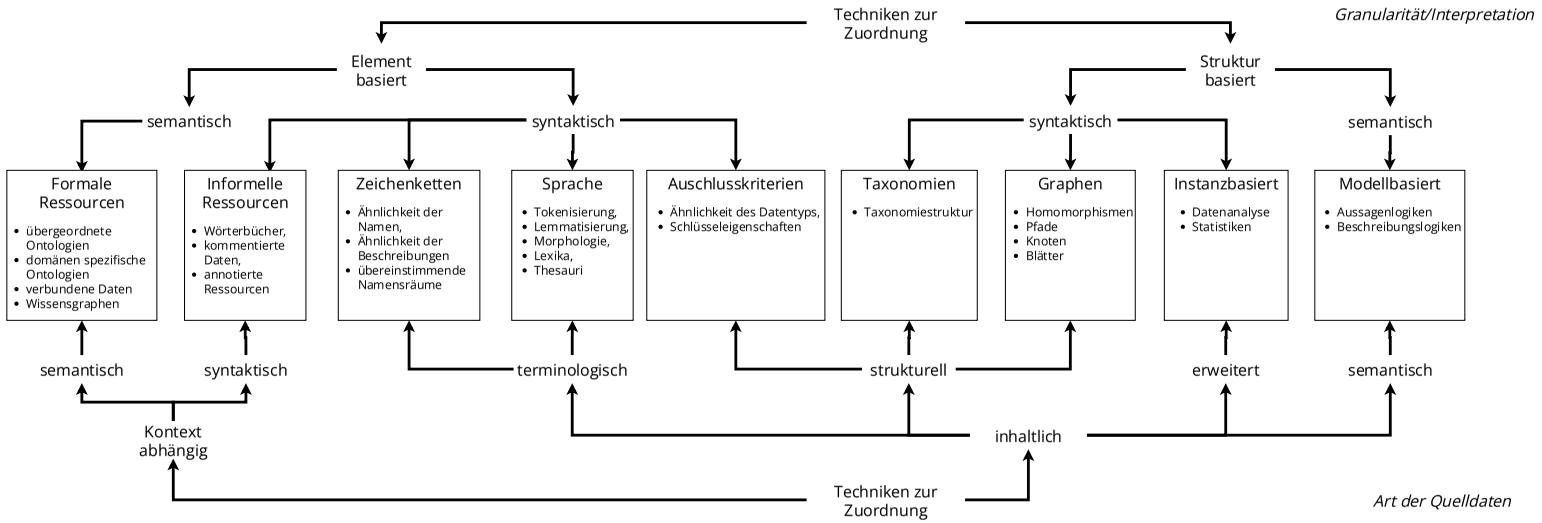


Abbildung 2.3: Kategorisierung der Matching-Ansätze nach Euzenat und Shvaiko [23]. Die obere Kategorisierung erfolgt über die Granularität und die Art und Weise der Interpretation der Daten im Matching-Prozess. Die untere Kategorisierung bezieht sich auf die Art der Informationen und wie sie im Matching-Prozess genutzt werden. Im mittleren Teil der Abbildung werden konkrete Techniken zur Ermittlung des Matchings benannt.

gensatz dazu analysiert ein Ansatz basierend auf der Struktur, wie Attributnamen und -Instanzen in der Gesamtheit genutzt werden.

- **Syntaktisch oder semantisch:** Syntax-basierte Ansätze analysieren ausschließlich die Struktur der Eingangsdaten. Andere Ansätze versuchen, die Bedeutung der Eingangsdaten z. B. mittels formaler Semantik zu ermitteln und die Ergebnisse zu begründen.

2. **Ursprung/Art der Informationen:** Diese Kategorie berücksichtigt, woher die Informationen stammen, die ein Matching-System für die Zuordnung nutzt, und welche Art von Information durch das System genutzt wird. Dazu können die folgenden Kriterien verwendet werden:
   - **Inhaltlich:** Die inhaltliche Unterscheidung erfolgt anhand der Datenkategorien, auf denen der Matching-Algorithmus arbeitet. Dazu gehören Zeichenketten bzw. Worte (terminologisch), Datenstrukturen (strukturell), Datenmodelle (semantisch) oder auch Daten- bzw. Attributinstanzen (erweitert).
   - **Kontextabhängig:** Kontextabhängig sind Ansätz dann, wenn sie zusätzliche, externe Ressourcen verwenden. Dabei erfolgt lediglich die Unterteilung zwischen Ressourcen, die eine zusätzliche Interpretation benötigen (semantisch), und solchen, die ohne Interpretation auskommen (syntaktisch).

Ausführlichere Informationen zu konkreten Matching-Techniken und -Systemen finden sich in Kapitel 9 und [24].

2.1.5 Integration von Produktkatalogen

Fensel et al. [25] bezeichnen die automatische Katalogintegration als *Produktschema-Integration*, wobei Zuordnungen zwischen Attributen aus mehreren Eingangsschemata und einem Zielschema gebildet werden. Mithilfe dieser Zuordnungen können Produktdaten aus verschiedenen Quellen in ein vereinheitlichtes Zielschema überführt werden.

Wie bereits beschrieben, müssen Händler ihre Produktkataloge entweder in ein vorgegebenes Schema überführen oder Mitarbeiter des Marktplatzes müssen diese Aufgabe übernehmen. Abbildung 2.4 stellt schematisch die Integration von Produktkatalogen mehrerer Händler in einen Marktplatz dar. Dabei müssen zusammenge-

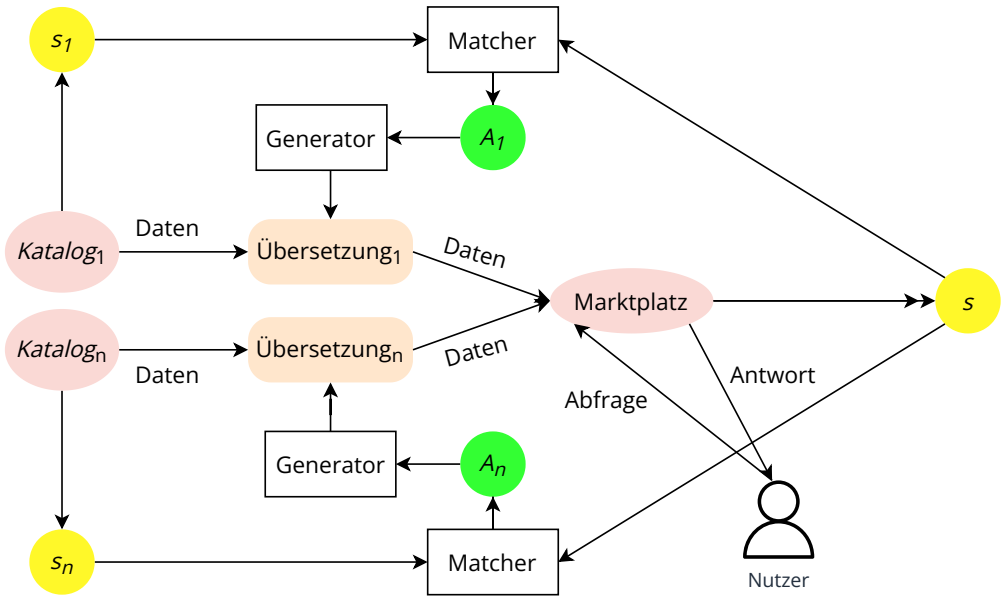


Abbildung 2.4: Katalogintegration nach Euzenat und Shvaiko [26]. Jeder Händler, der ein Produkt auf dem Marktplatz anbieten möchte, muss zugehörige Elemente zwischen seinem Katalog (mit Schema s_i) und dem Schema des Marktplatzes (S) ermitteln. Als Ergebnis der Zuordnung (Alignment, A_i) wird eine Übersetzung generiert, die genutzt wird, um den Katalog in den Marktplatz zu importieren. Ein Nutzer kann den Marktplatz anschließend nach integrierten Produkten durchsuchen.

hörende Attribute zwischen den Schemata der Händler und des Marktplatzes identifiziert werden.

Sobald die Zuordnungen ermittelt wurden, können die Daten aus dem Katalog eines Händlers automatisch in das Schema des Marktplatzes übersetzt werden. Ein Nutzer kann dann über das vereinheitlichte Schema auf die Produktdaten zugreifen. Das abgebildete Szenario bildet einen typischen Anwendungsfall für die Integration mehrerer Datenquellen in ein DWH ab [26].

Umgesetzt wird das beschriebene Vorgehen üblicherweise über sog. *extract, transform, load (ETL)-Prozesse*. Diese ETL-Prozesse beginnen mit dem Einlesen der gelieferten Daten beginnen und enden mit dem Einstellen der Daten in ein Zielsystem. Ein solcher ETL-Prozess besteht aus den folgenden Prozessschritten [11, 27, 28]:

- **Datenimport (extract):** Der erste Schritt in einem ETL-Prozess ist der Datenimport. Aus verschiedenen Datenquellen werden für die weitere Verarbeitung al-

le relevanten Daten aus einer oder mehreren Quellen gesammelt. Im Bereich von Online-Shops bestehen diese Daten häufig aus tabellarischen Produktkatalogen im XLSX- oder CSV-Format. In Abbildung 2.4 werden diese Daten mit Katalog_i bezeichnet.

- **Transformationen (transform):** Im Transformationsschritt werden die zuvor gesammelten Daten in ein einheitliches Zielschema und -format überführt. Dazu werden die Daten in verschiedenen Teilprozessen aufbereitet und ggf. mit weiteren Informationen angereichert. Typische Teilprozesse sind das Schema-Matching sowie das Zusammenführen, Aggregieren und Filtern der Quelldaten. Während Abbildung 2.4 den Schritt der Schemazuordnung weiter unterteilt, finden sich die weiteren Transformation zusammengefasst in der jeweiligen Übersetzung_i der Kataloge in den marktplatzkonformen Katalog.

- **Qualitätssicherung:** Ein Schritt der häufig durchgeführt wird, aber in Abbildung 2.4 nicht dargestellt wird, ist die Qualitätsicherung. Sie wird in der Regel durchgeführt bevor die Daten in den Marktplatz integriert werden. Enthalten die importierten Daten Duplikate, also mehrere Einträge zum gleichen Produkt, sollte dies erkannt werden. Im Online-Handel treten Duplikate unter anderem auf, wenn mehrere Zulieferer das gleiche Produkt anbieten. Neben der Erkennung von Duplikaten können weitere Maßnahmen zur Qualitätssicherung erforderlich sein. Dazu zählen die Standardisierung von Anschriften, die Vereinheitlichung von Abkürzungen, die Überprüfung von Einschränkungen durch Geschäftsregeln sowie die inhaltliche Kontrolle der zu importierenden Daten durch Spezialisten.

- **Speichern (load)**: Am Ende des ETL-Prozesses werden die Ergebnisse der Transformationen persistent in einem Zielsystem, wie z. B. einem DWH oder einer Datei, gespeichert. Je nach Anwendungsfall und Geschäftsmodell sind mehrere Zielsysteme mit unterschiedlichen Datenstrukturen möglich, sofern die Daten entsprechend aufbereitet wurden.

2.2 Schema-Abgleich für Produktkataloge

Grundsätzlich ist der Schema-Abgleich für Produktkataloge ein spezieller Anwendungsfall des in Abschnitt 2.1.4 vorgestellten Schema- bzw. Ontologie-Matching-Problems. In diesem Abschnitt werden Katalogschemata von Zulieferern anhand

der vorliegenden Daten analysiert. Weiterhin wird auf die typischen Kardinalitäten zwischen den Eingangs- und Zielschemata bei der Katalogintegration eingegangen.

2.2.1 Katalogschemata von Zulieferern

Wie beschrieben, mangelt es im Antikörpermarkt an Datenaustauschstandards. Jeder Zulieferer verwendet daher ein eigenes Schema, um seine Produktdaten zu beschreiben. Dies wird durch Untersuchung einer Stichprobe von 19 verschiedenen Antikörper-Herstellern untermauert. Abbildung 2.5 zeigt die Ergebnisse dieser Untersuchung. Der Zulieferer mit den wenigsten Attributen nutzt lediglich 14 zur Beschreibung seiner Produkte, während der Zulieferer mit den meisten Attributen 135 verwendet. Allerdings verwendete dieser Zulieferer direkt das Schema des Marktplatzes.

Die meisten Zulieferer nutzen zwischen 28 und 55 Attribute zur Beschreibung ihrer Produkte, im Median 36. Im direkten Vergleich zu den 135 Attributen werden Produkte von Zulieferern folglich ungenauer beschrieben als durch den Marktplatzbetreiber. Diese Beobachtung ist auch auf weitere Domänen, wie zum Beispiel den Einrichtungsmarkt (Möbel, Beleuchtung usw.), übertragbar [10].

Die Ursache für die Diskrepanz in der Anzahl genutzter Attribute ist in der wirtschaftlichen Notwendigkeit für die Marktplatzbetreiber zu suchen. Qualitativ hochwertige Daten führen zu mehr Verkäufen, weniger Rückgaben und insgesamt einer höheren Kundenbindung. Somit ist eine möglichst genaue Attribuierung von Produkteigenschaften für den Marktplatzbetreiber von großer Bedeutung.

Aus der Beobachtung, dass Zulieferer weniger Attribute nutzen als Marktplatzbetreiber, lässt sich schlussfolgern, dass in der Katalogintegration in der Regel grobe Schemata mit feingranularen Schemata abgeglichen werden müssen.

2.2.2 Beziehungen zwischen Eingangs- und Zielschema

Die Beziehungen in den Zuordnungen zwischen Eingangs- und Ausgangsschema lassen sich durch Kardinalitäten beschreiben. In Abbildung 2.6 findet sich ein vereinfachtes Beispiel aus dem Antikörpermarkt. Darin soll ein tabellarisches Eingangsschema eines Zulieferers mit verschiedenen Attributen in das Zielschema des Marktplatzbetreibers überführt werden.

Typische Kardinalitäten sind $1:1$, wenn genau ein Attribut aus dem Eingangsschema einem Attribut im Zielschema zugeordnet wird, oder $N:1$, wenn beispielsweise

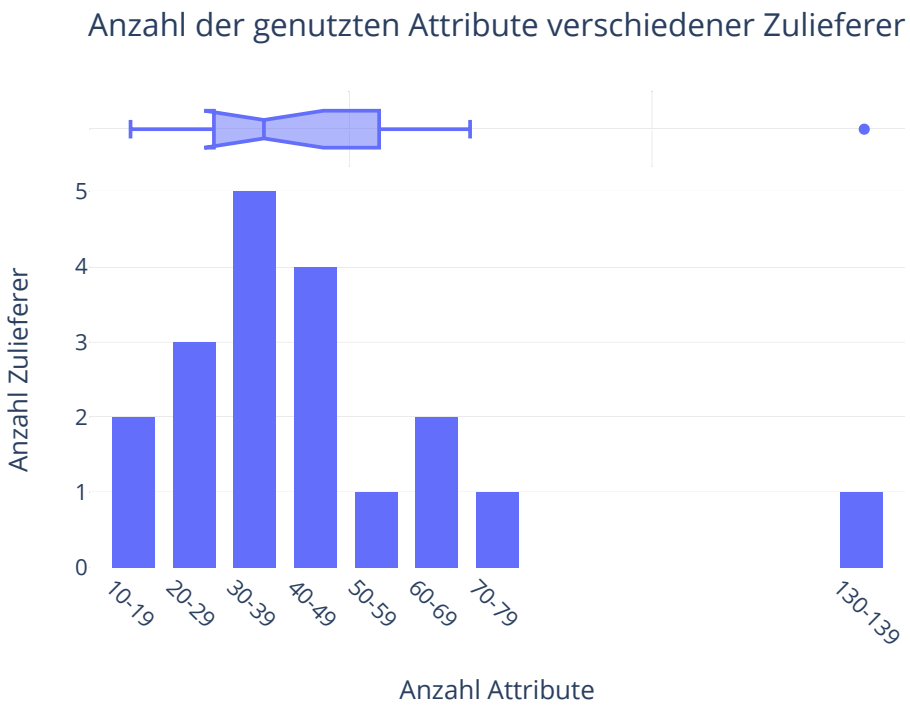


***Abbildung 2.5:** Verteilung der Anzahl genutzter Attribute aus Produktkatalogen von 19 verschiedenen Zulieferern im Antikörpermarkt. Im Median werden 36 Attribute genutzt. Der Zulieferer mit den wenigsten Attributen nutzt lediglich 14, während ein Zulieferer mit 135 Attributen direkt das Zielschema verwendet.*

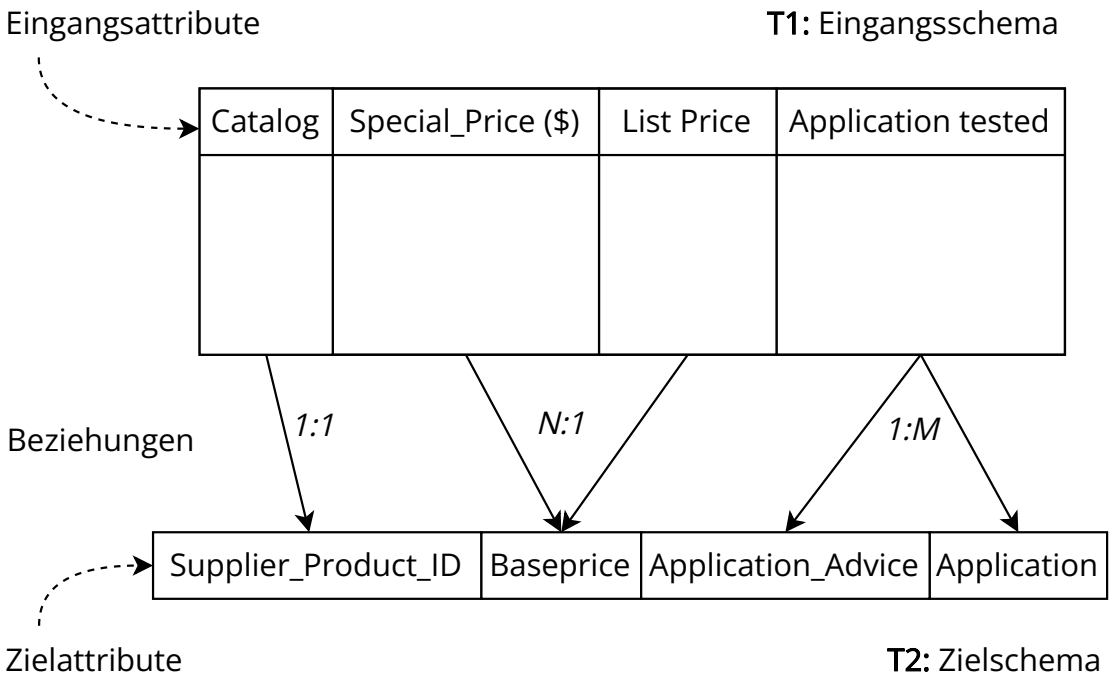


Abbildung 2.6: Beziehung zwischen Attributnamen aus tabellarischen Eingangs- und Zielschemata am Beispiel von Antikörpern (vgl. Schmidts et al. [S3])

mehrere Preisinformationen kombiniert werden müssen. $1 : M$ Kardinalitäten treten vor allem dann auf, wenn Informationen aus einem Attribut des Eingangsschemas auf mehrere Attribute des Zielschemas aufgeteilt werden müssen, weil das Zielschema die Produktdaten genauer erfasst. Häufig enthalten diese Attribute längere Texte, die zur Beschreibung von Informationen verwendet werden, die nicht durch das eigene Schema abgebildet werden. Dennoch treten $1 : 1$ Beziehungen zwischen Katalogen am häufigsten auf.

Nur selten werden Informationen eines Zulieferers in der Praxis verworfen ($N : 0$). Eher werden die gelieferten Informationen im Laufe des ETL-Prozesses durch externe Ressourcen, wie Wissensdatenbanken, weiter angereichert ($0 : M$).

Anstelle der Kardinalitäten lässt sich auch der Ursprung einer Information betrachten. Aus Sicht des Zielschemas in Abbildung 2.7 stammt die Information zu `Clone` aus genau einem Eingangsattribut, dem `Product_Name`. Das Attribut `Reactivity` aus dem Zielschema kann dagegen aus mehreren Eingangsattributen stammen. Attribute, die mehrere mögliche Quellen besitzen, müssen im Laufe des ETL-Prozesses weiter konsolidiert werden.

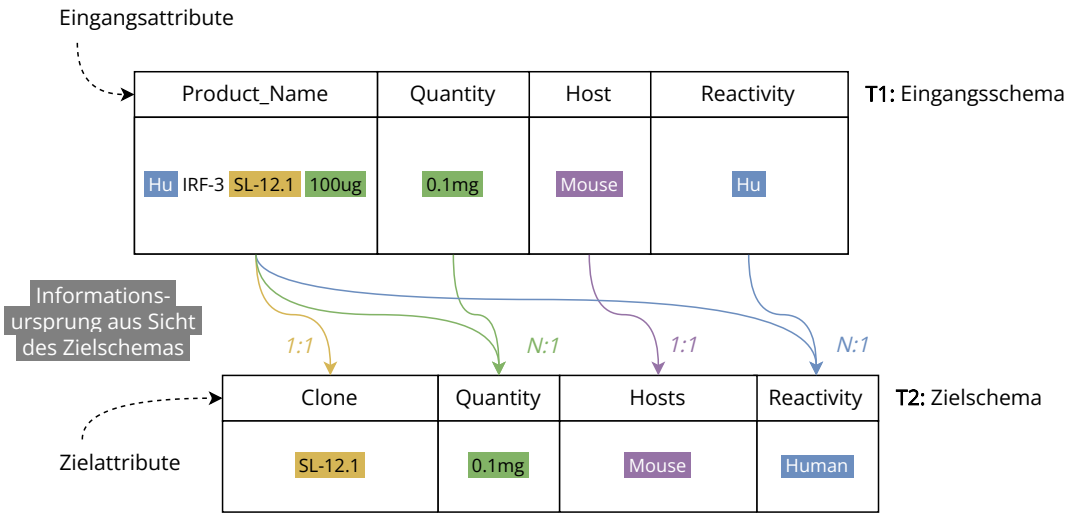


Abbildung 2.7: Informationsursprung aus Sicht des Zielschemas. Eine Information kann unabhängig von Kardinalitäten aus einer oder N Quellen stammen, die durch nachfolgende Transformationen konsolidiert werden müssen.

Die Betrachtung des Informationsursprungs bietet vor allem für aufbauende ETL-Schritte Vorteile. Ist bekannt, woher eine Information stammt, können spezialisierte Transformationen angewendet werden und so zu einer höheren Datenqualität führen.

2.3 Gängige Standards für Produktdaten

Je nach Anwendungsfall und Domäne werden unterschiedliche Formate (aus technischer Sicht) und Schemata (aus fachlicher Sicht) zur Beschreibung der Produktdaten verwendet. Während Organisationen sich für eine Standardisierung einsetzen (z. B. schema.org, GS1) und den Austausch von Produktdaten vereinfachen wollen, nutzen Online-Marktplätze eigene Schemata, um Händlern die Bereitstellung von Produkten zu ermöglichen. Dazu zählen unter anderem Amazon [7], Galaxus [8] oder Shopify [9]. Jeder dieser Marktplatzbetreiber nutzt aufgrund seiner Autonomie ein eigens entwickeltes Schema, das genau auf die Bedürfnisse des Marktplatzbetreibers zugeschnitten ist. Im Falle von Amazon werden Schemata sogar auf die jeweilige Produktkategorie angepasst.

Hervorzuheben ist, dass jedes Schema allgemeine Produktattribute enthält, aber Benennungen und Schreibweisen häufig nicht übereinstimmen. In den folgenden Abschnitten wird auf ausgewählte Schemata eingegangen und Unterschiede werden hervorgehoben. Zuerst werden Schemata von Organisationen vorgestellt, bevor Shopify stellvertretend für Betreiber von Online-Marktplätzen mit entsprechender Marktmacht herangezogen wird.

2.3.1 schema.org/Product

schema.org wurde 2011 gegründet, um ein gemeinsames maschinenlesbares Datenformat zu entwickeln, das den Informationsaustausch über einzelne Webseiten hinaus ermöglicht. Es wurden Schemata erarbeitet, die in einer möglichst allgemeingültigen Struktur Daten im Internet beschreiben. Dazu stellt schema.org ein Vokabular bereit, das in verschiedenen Kodierungen (z. B. Resource Description Framework (RDF)) als Metainformation in eine Webseite eingebunden werden kann.

Über das Vokabular können Entitäten und deren Beziehungen abgebildet werden. Der Fokus liegt dabei nicht, die Realität mit allen Details vollständig abzubilden, sondern auf der einfachen Zugänglichkeit für Anwender. Dadurch soll das standardisierte Schema von einer breiten Masse genutzt werden. So können Suchmaschinen beispielsweise Fragen beantworten, indem sie zuvor den Quellcode und die darin enthaltenen Metainformationen von schema.org ausgewertet haben [29, 30]. Für Online-Shops stellt schema.org ein Schema für Produkte[1] zur Verfügung, das für alle angebotene Produkte und Services, wie z. B. Kleidung, Tickets oder Abonnements, gedacht ist [31].

Tabelle 2.1 zeigt einen Ausschnitt der Schemadefinition von schema.org für Produkte. Über die im Schema definierten Eigenschaften lassen sich Produkte allgemein beschreiben. Typische Produkteigenschaften sind beispielsweise die Beschreibung (`description`), eine Produktkategorisierung (`category`), eine Marke (`brand`) sowie ein Name und eine eindeutige ID. Zusätzlich können Verweise über URLs angegeben werden. Der angegebene Auszug verdeutlicht, dass das Schema im allgemeinen geeignet ist, Produkte oder Dienstleistungen abzubilden.

Für einen Online-Shop können jedoch zusätzliche, spezielle Eigenschaften von Bedeutung sein, die nicht im Schema von schema.org abgebildet sind. Darunter fallen unter anderem Haltbarkeitsdaten, die für Lebensmittel und in der Pharmazie unerlässlich sind. Im Standard enthalten sind lediglich Produktionszeitpunkt, Kaufda-

[1] https://schema.org/Product

Eigenschaft	Beschreibung
description	A description of the item.
category	A category for the item. Greater signs or slashes can be used to informally indicate a category hierarchy.
brand	The brand(s) associated with a product or service, or the brand(s) maintained by an organization or business person.
name	The name of the item.
productID	The product identifier, such as ISBN. For example: meta itemprop=„productID“ content=„isbn:123-456-789“.
weight	The weight of the product or person.
url	URL of the item.

Tabelle 2.1: Auszug von Eigenschaften mit der zugehörigen Beschreibung der Schemadefinition für Produkte von schema.org [31]. Die Eigenschaften sind allgemein gehalten, sodass grundsätzlich jedes Produkt oder jede Dienstleistung durch das Schema beschrieben werden kann.

tum und Veröffentlichungsdatum. Ein Haltbarkeitsdatum müsste von jedem Online-Shop über zusätzliche Eigenschaften definiert werden. Je genauer ein Produkt über Eigenschaften beschrieben werden soll, desto weniger eignet sich folglich der Standard von schema.org.

2.3.2 GS1-Standard

GS1[2][3] hat sich zum Ziel gesetzt, Lieferketten durch standardisierte Produktdaten zu optimieren, indem der Aufwand zum Datenaustausch und der Datenverarbeitung reduziert wird. Durch den Standard sollten Nutzer von besseren Suchergebnissen, verbesserter Datenqualität und optimierter Endnutzererfahrung profitieren. Aus technischer Sicht ist er eine Erweiterung des schema.org-Standards und in der Verwendung direkt kompatibel [32, 33].

Der GS1-Standard konzentriert sich u.a. auf die speziellen Domänen Kleidung, Lebens- und Futtermittel, sowie die Alkohol- und Tabakindustrie. Neben den speziellen Attributen der jeweiligen Domäne werden zusätzlich allgemeine Eigenschaften

[2] GS1 ist eine nicht auf Profit ausgerichtete Organisation (non profit organization). Sie ist spezialisiert auf die Standardisierung von Produktdaten und deren Austausch zwischen Unternehmen, beispielsweise über weltweit eindeutige Barcodes.

[3] `https://www.gs1.org/`

– wie im schema.org Standard – erfasst. Selbst die allgemeinen Eigenschaften sind jedoch spezifischer auf Produktdaten zugeschnitten als es im schema.org Standard der Fall ist.

Tabelle 2.2 zeigt einen Ausschnitt aus den allgemeinen Produkteigenschaften des GS1-Standards. Die aufgelisteten Eigenschaften sind direkt mit den Eigenschaften von schema.org vergleichbar (vgl. Tabelle 2.1). Im direkten Vergleich der beiden Standards fällt auf, dass im GS1-Standard die Eigenschaften bzw. Attribute spezifischer benannt und genauer beschrieben sind als bei schema.org. So werden z. B. die Beschreibung (`productDescription`), der Name (`productName`) und die ID im GS1-Standard mit dem Präfix *product* benannt, da im Standard mehrere Namen, Beschreibungen und IDs enthalten sind. Dadurch können Eigenschaften im GS1-Standard eindeutig identifiziert werden, ohne eine Eigenschaftshierarchie aufzubauen.

Der Detailgrad des Standards wird bereits bei allgemeinen Eigenschaften wie dem Gewicht deutlich. Während bei schema.org nur die Eigenschaft `weight` existiert, finden sich im GS1-Standard (siehe Tabelle 2.2) `netWeight`, `drainedWeight`, `grossWeight` und `textileMaterialWeight` [33]. Dadurch lassen sich auch Produkte, die nicht aus den Spezialdomänen stammen, genauer beschreiben als mit dem schema.org-Standard.

Zusätzlich zu allgemeinen Produkteigenschaften finden sich im GS1-Standard spezielle Attribute, die insbesondere für Lebensmittel und Kleidung relevant sind. Dazu zählen die Eigenschaften in Tabelle 2.3. Dabei muss der Standard gesetzlich vorgeschriebene Eigenschaften berücksichtigen, wie zum Beispiel das Mindesthaltbarkeitsdatum (`expirationDate`) oder die Angabe von Inhaltsstoffen. Andere Eigenschaften sind lediglich Informationen für den Endverbraucher (z. B. `clothingCut`), die eine Kaufentscheidung unterstützten sollen.

Zusammenfassend lässt sich sagen, dass der hohe Detailgrad den GS1-Standard auszeichnet. Insgesamt umfasst er 395 Eigenschaften, wovon mit 187 Eigenschaften Lebensmittel bzw. lebensmittelähnliche Produkte beschrieben werden können. Der GS1-Standard ist ein Beispiel dafür, wie komplex eine einheitliche Benennung von Produktattributen sein kann.

2.3.3 Shopify

Shopify stellt Unternehmen eine E-Commerce-Plattform bereit, mit deren Hilfe Online-Shops aufgebaut oder verwaltet werden können. In den erstellten Online-Shops

Eigenschaft	Beschreibung
brandName	The brand(s) associated with a product or service, or the brand(s) maintained by an organization or business person.
gpcCategoryDescription	A description of the code specifying a product category according to the GS1 Global Product Classification (GPC) standard.
productDescription	An understandable and useable description of a product using brand and other descriptors. This attribute is filled with as little abbreviation as possible, while keeping to a reasonable length. [...]
productName	Consumer friendly short description of the product suitable for compact presentation.
productID	Additional means to the GTIN to identify a product.
netWeight	Used to identify the net weight of the product. Net Weight excludes all packaging material, including the packaging material of all lower-level GTINs. Example:11.5 kg.
referencedFileURL	Simple text string that refers to a resource on the internet, URLs may refer to documents, resources, people, etc.

Tabelle 2.2: Ausschnitt aus den allgemeinen Produktattributen des GS1-Standards [33]

können Produkte oder Dienstleistungen angeboten werden. Zur einfachen Integration der Produkt- bzw. Dienstleistungsdaten stellt Shopify einen eigenen Standard bereit, mit dessen Hilfe jeder Shop-Betreiber Produkte über eine CSV-Datei in den Shop importieren kann. Sollten Produkteigenschaften nicht durch den Standard abgebildet werden können, bietet Shopify die Möglichkeit eigene Metafelder zu definieren. [9, 34, 35]

Tabelle 2.4 zeigt einen Ausschnitt der von Shopify vorgegebenen Attribute für den Import über eine CSV-Datei. Beim Vergleich des Shopify-Standards mit den beiden vorherigen Standards fällt auf, dass ein anderes, eher technisches Vokabular zur Beschreibung der Produkteigenschaften genutzt wird.

Eigenschaft	Beschreibung
bestBeforeDate	Best before date on the label or package signifies the end of the period under which the product will retain specific quality attributes [...]. Best before date is primarily used for consumer information and may be a regulatory requirement.
expirationDate	The expiration date is the date that determines the limit of consumption or use of a product/coupon. [...] It is often referred to as 'use by date' or 'maximum durability date'.
clothingCut	Supplemental information to indicate the clothing cut or silhouette make of the garment. [...]
consumerSafetyInformation	Information on consumer safety regarding the product.
fatInMilkContent	The percentage of fat contained in milk content of the product.
ingredientName	Free text field describing an ingredient or ingredient group. [...]

Tabelle 2.3: Ausschnitt aus den domänenspezifischen Produktattributen des GS1-Standards [33]

Beispielsweise entspricht die Eigenschaft `Handle` einer ID, der `Title` dem Produktnamen und `Body (HTML)` der Produktbeschreibung. Shopify verwendet analog zu dem schema.org-Standard allgemein gehaltene Eigenschaften für Produkte. Auch hier ist davon auszugehen, dass eine möglichst große Anzahl verschiedener Produkte oder Dienstleistungen über das Schema abgebildet werden sollen. Für spezielle Attribute bleibt die Möglichkeit, über Metafelder zu arbeiten (vgl. `additionalProperty` bei schema.org).

Ebenso ist der Standard von Shopify mit 48 Eigenschaften im Vergleich eher klein. Die enthaltenen Attribute fokussieren stärker als zuvor den direkten Betrieb des Online-Shops, während die Qualität der Daten in Bezug auf die genaue Beschreibung des Produkts nachrangig ist. Das wird durch die Anzahl der Attribute, die sich

Eigenschaft	Beschreibung
Handle	Handles sind eindeutige Namen für jedes Produkt. Sie können Buchstaben, Bindestriche und Zahlen enthalten, aber keine Leerzeichen, Akzente oder andere Sonderzeichen, auch keine Punkte. Handles werden in der URL für jedes Produkt verwendet. [...]
Title	Der Titel des Produkts.
Body (HTML)	Die Beschreibung des Produkts im HTML-Format. Hierbei kann es sich auch um reinen Text ohne Formatierung handeln.
Vendor	Der Name des Anbieters für das Produkt.
Standard product type	Ein Etikett, das den Produkttyp beschreibt. Dieses Etikett muss aus der vordefinierten, standardisierten Liste der Produkttypen stammen.
Variant Grams	Das Gewicht des Produkts oder der Variante in Gramm. [...]

Tabelle 2.4: Ausschnitt von Attributen aus dem Shopify-Import-Format [9]

auf Suchmaschinenoptimierung bzw. Google-Ranking beziehen, deutlich. Rund ein Drittel aller Attribute (15 von 48) beziehen sich auf Suchmaschinen.

Aufgrund der tabellarischen Struktur des vorgegebenen CSV-Formats sind keine Bezüge und Zusammenhänge einzelner Attribute beschrieben. Ein Nutzer muss Zusammenhänge selbst entdecken bzw. herstellen. Des Weiteren sind keine Vorgaben bezüglich Datentyp oder Wertebereich vorhanden.

2.4 Kategorisierung von Produktdaten

Der folgende Abschnitt wurde bereits zu großen Teilen in Schmidts et al. [S5] veröffentlicht.

Die Attribute, mit denen die Produkteigenschaften beschrieben werden, lassen sich in verschiedene Kategorien einordnen. Ausschlaggebend dafür sind die Typen der Attributinstanzen. Die Kategorien finden sich in allen vorgestellten Standards wieder.

2.4.1 Identifikationsnummern

IDs werden verwendet, um Produkte mindestens innerhalb eines Katalogs eindeutig zu identifizieren. Häufig werden die IDs firmenintern genutzt. Im Falle des Shopify-Standards bestimmt das Attribut `Handle` zusätzlich einen Teil der URL, unter dem das Produkt aufgerufen werden kann. Neben den firmeninternen IDs existieren globale IDs, wie z. B. die Global Trade Item Number (GTIN). Hierüber können Produkte über Firmengrenzen hinweg eindeutig identifiziert werden.

2.4.2 Numerische Werte

Attributinstanzen dieser Kategorie bestehen ausschließlich aus Zahlen. Dabei ist unerheblich, ob es sich um eine ganze Zahl oder eine Fließkommazahl handelt. Die Bedeutung numerischer Werte leitet sich in der Regel direkt aus dem Attributnamen ab. Ein typisches Beispiel ist das Attribut `Variant Grams` aus dem Shopify-Standard. Hier muss eine ganze Zahl angegeben werden, die dem Produktgewicht in Gramm entspricht. Weitere Beispiele für diese Kategorie sind Preise und Lagertemperaturen.

Wenn die Einheit zur Interpretation der Werte nicht im Attributnamen angegeben ist, wird sie entweder in einem zusätzlichen Attribut oder gemeinsam mit dem numerischen Wert angegeben. Im letzteren Fall wäre der numerische Wert nicht mehr dieser Kategorie zuzuordnen, sondern der Kategorie *unstrukturierter Text (UT)*.

2.4.3 Wertelisten

Attribute, die in diese Datenkategorie fallen, enthalten nur Worte aus einer vorgegebenen Liste. Die Liste wird entweder durch den genutzten Standard (beispielsweise `Standard product type`, `gpcCategoryDescription`, Länderkennungen nach ISO 3166 oder Zahlungsmethoden[4]) oder durch die Produktdomäne (z. B. Allergene) festgelegt. Auch Markennamen fallen häufig in diese Kategorie, mit der Einschränkung, dass sich die Werteliste nicht aus einem definierten Standard ergibt, sondern aus allen Anbietern eines Marktes. In diesem Fall kann die Werteliste unvollständig sein.

[4] https://www.gs1.org/voc/PaymentMethod

2.4.4 Unstrukturierter Text

Attribute dieser Kategorie enthalten entweder eine Liste von Wörtern oder ganze Sätze bis hin zu Absätzen mit mehreren Sätzen. Typische Beispiele für diese Klasse sind Produktnamen oder Beschreibungen. Die Kategorie ist jedoch nicht auf diese beiden Attribute beschränkt, sondern umfasst alle Attribute, die Freitext enthalten können und somit nicht in die Kategorie Werteliste fallen (vgl. `clothingCut`, `consumerSafetyInformation` aus Tabelle 2.3). Häufig werden auch stichpunktartige Aufzählungen anstelle von vollständigen und grammatikalisch korrekten Sätzen genutzt. Zur Vereinfachung können IDs und numerische Werte bei der Katalogintegration auch als Text bzw. Wertliste aufgefasst werden, um Fehler durch falsche Kategorisierung, zusätzlicher oder fehlender Symbole, Zeichen und Einheiten zu vermeiden.

2.4.5 Uniform Resource Locators

Einige Attribute beinhalten URLs, die auf weitere Inhalte verweisen. Dazu zählen Abbildungen des Produkts (z. B. *Image Scr* bei Shopify) oder PDF-Dateien mit zusätzlichen Informationen (z. B. *referencedFileURL* bei GS1), die ein Endkunde herunterladen kann. Bei URLs sind die einzigen beiden Anhaltspunkte für eine Zuordnung der Attributname und eine mögliche Dateiendung. URLs werden im ETL-Prozess entweder übernommen oder verworfen. Zusätzlich können URLs in Text eingebettet werden. In diesem Fall wird das Attribut allerdings der Kategorie *unstrukturierter Text* zugeordnet.

2.5 Herausforderungen im Umgang mit Produktdaten

Obwohl gemeinsam genutzte Standards große wirtschaftliche Vorteile bieten würden, benutzen vor allem Online-Marktplätze eigene Schemata. Dies gilt sowohl für den Antikörpermarkt, bei dem zwar offene Datenbanken wie z.B. *Uniprot*[5] mit offenen Ontologien existieren, als auch für den Einrichtungsmarkt, obwohl ein ISO-Standard existiert [10]. Selbst gut dokumentierte und verbreitete Standards wie der GS1-Standard werden nicht überall genutzt, wo dies grundsätzlich möglich wäre. Darüber hinaus zeigt eine Studie von GS1, dass die Daten, die den Betreibern von

[5] `https://www.uniprot.org/`

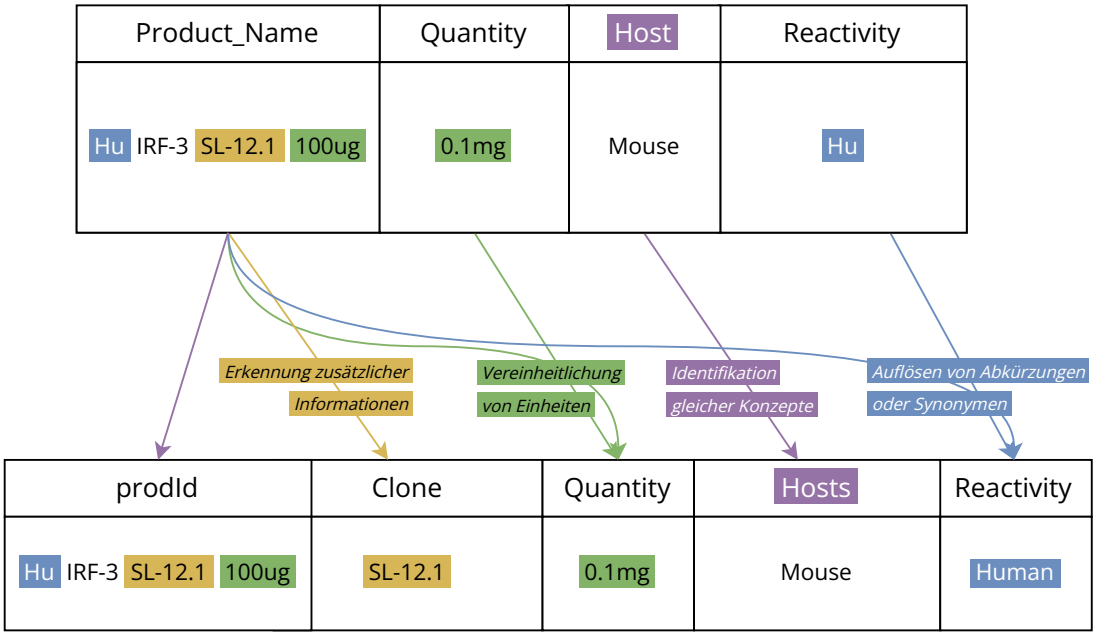


Abbildung 2.8: Die Abbildung zeigt typische Herausforderungen, die bei der Integration von Produktkatalogen auftreten können. Dazu zählen unter anderem die Erkennung von zusätzlichen, sonst nirgends angegebenen Informationen, die Vereinheitlichung von Einheiten, Identifizierung zusammengehöriger Konzepte oder das Auflösen von Synonymen.

Online-Shops zur Verfügung gestellt werden, bei 96% der Zulieferer inkonsistent sind und 48% der Produktdaten Fehler enthalten [36].

Die folgenden Abschnitte befassen sich mit den speziellen Herausforderungen, die bei der Integration von Produktkatalogen aufgrund fehlender oder nicht verwendeter Standards und Inkonsistenzen entstehen.

2.5.1 Identifizierung gleicher Konzepte

Die Identifizierung übereinstimmender Konzepte in Eingangs- und Zielschema ist die Kernaufgabe des Schema-Abgleichs. Stimmen Attributnamen aus beiden Schemata überein, handelt es sich in der Regel um eine $1:1$ Kardinalität. Stimmen die Schemata nicht überein, wie in Abbildung 2.8 zu sehen ist, ist die Identifizierung solcher Konzepte jedoch aus den folgenden Gründen schwierig:

1. Durch die manuelle Bearbeitung von Attributnamen oder -instanzen können im ETL-Prozess Probleme, wie Rechtschreibfehler und Buchstabendreher oder

durch die Verwendung des Plurals anstelle des Singulars (`Host` wird zu `Hosts`), auftreten. Zusätzlich kann die Nutzung verschiedener Standards zur Zeichenkodierung oder Sonderzeichen, die für Menschen die gleiche Bedeutung haben aber unterschiedlich kodiert werden, zu Problemen führen. So existieren beispielsweise für das &-Zeichen zwei weitere Unicode Kodierungen, die lediglich die Zeichengröße und -breite verändern.

2. Die Benennung von Attributen erfolgt nach unterschiedlichen Mustern. Darunter fällt die Verwendung von Abkürzungen und der Umgang mit Wortübergängen. Statt der in Texten üblichen Leerzeichen zwischen Worten, können Techniken wie *CamelCase*, *snake_case* oder *kebab-case*[6] verwendet werden. Diese Techniken wurden aus Programmiersprachen übernommen, damit Worte ohne Leerzeichen zusammengesetzt werden können. Der Einsatz dieser Techniken findet sich auch in vorhandenen Standards wieder (z.B. `productName` bei GS1 in Tabelle 2.3), obliegt aber den Vorlieben des Namen Gebenden, sodass in Produktdaten je nach Zulieferer auch unterschiedliche Techniken zum Einsatz kommen können.

3. Zusätzlich zu etablierten Standards mangelt es an übereinstimmenden, firmenübergreifenden, konzeptionellem Verständnis. Dadurch entstehen immer wieder Situationen, in denen Zulieferer und Marktplatzbetreiber zwar die gleichen Attributnamen verwenden, aber diese fachlich anders interpretieren. Dadurch können die Attributinstanzen in Eingangs- und Zielkatalog übereinstimmen, aber deren Attributnamen nicht. Folglich reicht eine reine Betrachtung von Attributnamen nicht aus, um übereinstimmende Konzepte zu identifizieren.

2.5.2 Erkennung zusätzlicher Informationen

Einige Attributinstanzen enthalten Informationen eines semantischen Konzepts, das im Zielschema als Attribut modelliert ist. Fehlt ein solches Attribut im gröberen Eingangschema, muss erkannt werden, dass diese Information trotzdem enthalten ist und dem richtigen Attribut zugeordnet werden ($1 : M$). In Abbildung 2.8 wird beispielsweise der Klon aus dem Produktnamen extrahiert, da in dem Eingabeschema für diese Information kein eigenes Attribut modelliert ist. Für gewöhnlich können

[6] Weitere Informationen zu den Schreibweisen finden sich unter `https://en.wikipedia.org/wiki/Letter_case`

ETL-Prozesse diese zusätzlichen Konzepte extrahieren, z.B. über eindeutige reguläre Ausdrücke, wenn ihnen diese bekannt sind. Andernfalls muss ein Mitarbeiter diese Konzepte identifizieren, extrahieren und schließlich manuell dem richtigen Attribut zuordnen. In der Praxis tritt letzteres meist aufgrund von Problemen bei den Schemazuordnungen auf, da nicht bekannt ist, welche Attributinstanzen Konzepte aus dem Zielschema enthalten. Wie bereits beschrieben, erfordert die manuelle Bearbeitung dieser Aufgabe sehr viel Zeit und ist fehleranfällig.

2.5.3 Vereinheitlichung

Die Vereinheitlichung von Informationen erfolgt in der Regel in einem späteren Transformationsschritt. Das Ziel ist, die Informationen im Online-Shop in gleicher Form zu präsentieren. Dazu müssen beispielsweise Abkürzungen aufgelöst werden, die herstellerspezifisch sein können (vgl. Abbildung 2.8, `Hu` wird `Human`).

Ein weiteres Beispiel ist die Vereinheitlichung von Einheiten. In Abbildung 2.8 werden in zwei verschiedenen Attributen Einheiten (`ug`, `mg`) genutzt. In der Transformation muss entschieden werden, welche Einheit zu verwenden ist, ob es sich um die gleiche Information handelt und ob die Daten ggf. umgewandelt werden müssen, um dem Standard des Marktplatzes zu entsprechen. Auch hier können Schwierigkeiten mit Schreibweisen auftreten.

2.5.4 Wortreichtum der Attributinstanzen

Abhängig vom Zulieferer fallen Produktattribute, wie in Abschnitt 2.4 beschrieben, in die Kategorien IDs, Wertelisten, numerische Werte, unstrukturierter Text oder URLs. Während IDs entweder nicht oder unverändert übernommen werden, kann die Bestimmung der relevanten Zuordnungen für Attributinstanzen der Typen *Werteliste* und *unstrukturierter Text* herausfordernd sein, wenn sich die Attributnamen der Schemata unterscheiden.

Wertelisten Entspricht eine Attributinstanz einem Element aus einer standardisierten Werteliste, lässt sich die Zuordnung leicht ermitteln, da der Zielattributname für die standardisierten Elemente bereits bekannt ist. In diesem Fall kann der Eingangsattributname ignoriert werden und die Zuordnung direkt erfolgen. Dies ist jedoch nicht der Regelfall. Häufig weichen die genutzten Attributinstanzen, wie auch die Schreibweisen der Attributnamen, voneinander ab. Beispielsweise wenn

eine Attributinstanz ein Allergen enthält, aber die Schreibweise von der standardisierten Werteliste nicht enthalten ist oder wenn mehrere Attributinstanzen Elemente der gleichen Werteliste enthalten, z. B. Länderkennzeichen für das Herstellungsland, den Zielmarkt oder die Lagerorte. Hier muss der Kontext der Attributinstanz bestimmt werden, um letztendlich die Zuordnung ermitteln zu können.

Unstrukturierter Text Attribute, die UT enthalten, können bei der Integration entweder $1:1$ Kardinalitäten, wie beispielsweise Produktbeschreibungen oder Zusatzinformationen für den Endkunden, oder $1:M$ Kardinalitäten abbilden. Letzteres trifft insbesondere dann zu, wenn bei der Transformation grob granulare Schemata in fein granulare Schemata überführt werden müssen. Ein Beispiel hierfür ist die Überführung des Shopify-Standards in den GS1-Standard. In diesem Fall könnten in der Produktbeschreibung (`Body HTML`) Informationen (Mindesthaltbarkeitsdatum, Zutaten, o.ä.) enthalten sein, die im GS1-Standard in separaten Attributen abgebildet werden. Auch das Attribut `Product_Name` in Abbildung 2.8 fällt in die UT-Kategorie.

2.5.5 Fehlende (Daten-)Strukturen

Im Gegensatz zu stark strukturierten Formaten wie OWL oder XML können in tabellarischen Formaten wie XLSX oder CSV nicht ohne weiteres Informationen bezüglich des Datentyps und den Beziehungen zwischen den Attributen hinterlegt werden. Dadurch können die Attributinstanzen häufig nur als Zeichenkette interpretiert werden. Darüber hinaus werden in der Tabellenstruktur keine Beziehungen zwischen den Spalten der Tabelle definiert. Beziehungen zwischen den Spalten lassen sich im günstigsten Fall für einen Menschen aus dem Kontext kombiniert mit allen Attributnamen der Tabelle herleiten.

Durch die tabellarischen Formate lassen sich in der Katalogintegration weder Informationen zu Datentyp noch Attributbeziehungen nutzen, um Zuordnungen zwischen Eingangskatalog und Zielschema zu identifizieren. Daher bleibt nur das Interpretieren der Zeichenketten aus Attributnamen und -Instanzen.

2.6 Zusammenfassung

In diesem Kapitel wurden zunächst die theoretischen Hintergründe zur Datenintegration dargestellt, bevor die Definition der Katalogintegration als Spezialfall des

Schema-Matchings erfolgte. Danach wurden einige von Konsortien oder Unternehmen etablierte Standards vorgestellt. Anschließend wurden die Besonderheiten der Katalogintegration in Bezug auf das Schema-Matching am Beispiel eines Antikörper-Marktplatzes dargelegt.

Des Weiteren wurden die unterschiedlichen Kategorien eingeführt, in die sich Produktattribute unterteilen lassen. Dazu zählen IDs, Wertelisten, unstrukturierter Text, numerische Werte und URLs. Insbesondere durch Wertelisten mit mehreren Interpretationsmöglichkeiten und unstrukturierten Text entstehen weitere Herausforderungen, wie beispielsweise die Erkennung zusätzlicher Informationen, die in unstrukturiertem Text enthalten sind.

Kapitel 3

Grundlagen des maschinellen Lernens

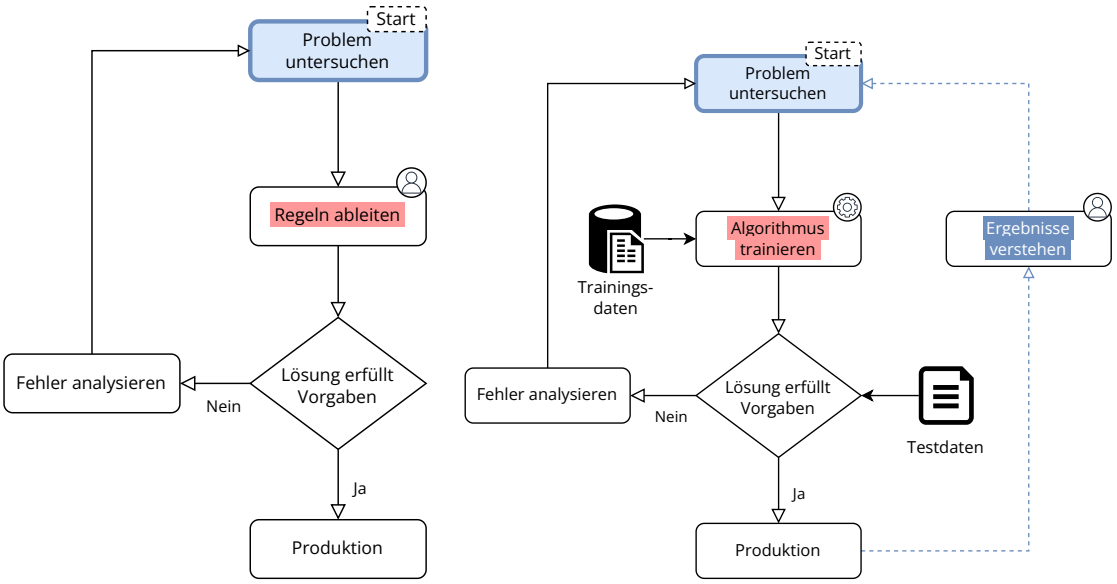
Maschinelles Lernen (ML) ist aus der Idee entstanden, Menschen das anstrengende Implementieren von Regeln zu erleichtern und Computer so zu programmieren, dass diese Probleme möglichst eigenständig und ohne zusätzlichen Aufwand lösen können [37].

Eben dieser Aufgabenstellung widmet sich die Leitfrage dieser Arbeit in Bezug auf die Katalogintegration. Daher werden in diesem Kapitel die Grundlagen von ML erklärt. Zunächst wird auf die grundlegenden Unterschiede in der Problemlösung zwischen einem klassischen Vorgehen und einem ML-basierten Vorgehen eingegangen. Im Anschluss wird ein Überblick über ML-Systeme, deren Entwicklung, sowie typische Herausforderungen beim Einsatz von ML-Systemen gegeben. Abschließend werden verschiedene Klassifikationsverfahren sowie deren möglicher Einsatz der Schemazuordnung und Bewertungsansätze für deren Ergebnisse vorgestellt.

3.1 Problemlösung durch maschinelles Lernen

Im klassischen datengetriebenen Vorgehen müssen Entwickler typischerweise ein Problem anhand von Beispieldaten untersuchen und Muster darin erkennen. Danach können sie Regeln ableiten und in einen Algorithmus zu implementieren, mit dem das Problem gelöst werden kann.

Soll beispielsweise ein Spamfilter klassisch entwickelt werden, müssen die Entwickler Spam-Nachrichten untersuchen und Gemeinsamkeiten finden, die zur Erkennung von Spam-Nachrichten führen. Dabei kann es sich um das Vorkommen bestimmter Wörter und Phrasen handeln oder um komplexere Regeln, die mit regulären Ausdrücken abgebildet werden können.



(a) Klassisches Vorgehen *(b) Problemlösung durch ML*

Abbildung 3.1: Vergleich des grundsätzlichen Vorgehens bei der Entwicklung von klassischen Lösungen im Vergleich zum Entwicklungsprozess bei Anwendungen, die auf maschinellem Lernen basieren.

Sobald diese Muster durch einen Entwickler gefunden wurden, müssen sie in einen Algorithmus übertragen werden, der die Nachrichten entsprechend markiert. Wenn dieser Algorithmus in einer Testphase die Anforderungen erfüllt, kann der Spamfilter in Produktion gehen. Ein solches Vorgehen ist in Abbildung 3.1a zu sehen.

Mit dieser klassischen Vorgehensweise entstehen bei der Entwicklung des Spamfilters jedoch Probleme:

- Spam wird anhand einer Kombination von Regeln erkannt.
- Sind die Regeln zu einfach, führen sie zu vielen fehlerhaften Kategorisierungen.
- Je mehr Regeln im System enthalten sind, desto schwieriger wird die Wartung.

Trotzdem ist das klassische Vorgehen auch heute noch üblich.

Im Gegensatz dazu steht das Vorgehen über maschinelles Lernen, das in Abbildung 3.1b abgebildet ist. Hier beginnt der Entwicklungsprozess zwar auch mit der

Untersuchung des Problems, danach wird jedoch aus den vorliegenden Daten eine Teilmenge als Trainingsdatensatz gewählt, mit dessen Hilfe ein ML-Algorithmus trainiert wird. Im Falle eines Spamfilters wird versucht, dem System anhand von positiven und negativen Beispielen beizubringen, wie eine Spam-Nachricht aussieht, ohne manuell Erkennungsmuster festzulegen. Das System lernt automatisch, welche Phrasen oder Worte mit Spam-Nachrichten in Verbindung zu bringen sind, und kann basierend darauf Entscheidungen fällen.

Um herauszufinden, ob das daraus entstehende ML-System die Vorgaben erfüllt, wird ein Testdatensatz genutzt, der ebenfalls aus den zugrundeliegenden Daten gebildet wird. Folglich benötigt dieses System keine festgelegten Regeln und ist dadurch leicht zu warten. Wird das System zu ungenau, können weitere Daten hinzugefügt und der ML-Algorithmus neu trainiert werden. Dieser Prozess lässt sich automatisieren, sodass beispielsweise auch Veränderungen in der verwendeten Sprache in Spam-Nachrichten über die Zeit erfasst werden können.

Darüber hinaus können ML-Ansätze Menschen helfen, das eigentliche Problem besser zu verstehen, indem sie die Vorhersagen des Systems untersuchen und verstehen, warum diese getroffen wurden. Die so gewonnenen Erkenntnisse lassen sich nutzen, um klassische oder ML-Systeme weiter zu verbessern. Die Anwendung von ML-Techniken auf große Datenmengen zur Erkennung von nicht offensichtlichen Mustern wird *Data Mining (DM)* genannt [38].

3.2 Arten von ML-Systemen

ML-Systeme können in verschiedenen Varianten umgesetzt werden. In den folgenden Abschnitten werden häufig genutzte Varianten vorgestellt.

3.2.1 Supervised ML-Systeme

Bei *supervised* ML-Systemen müssen Daten zunächst mit *Labeln* versehen (annotiert) werden, die der erwarteten Lösung entsprechen. Diese Daten werden auch als *Gold-Standard* bezeichnet. Ein Algorithmus lernt dann von den annotierten Daten. Typische Aufgaben solcher Systeme sind z. B. binäre Klassifikationen (Spam, kein Spam), oder die Vorhersage eines numerischen Wertes (Regression) anhand von bestimmten Merkmalen (Kaufpreis eines Autos anhand von Kilometerleistung, Ausstattung etc.). Typische Algorithmen sind

- Support Vektor Maschinen (SVMs),

- Entscheidungsbäume und
- Neuronale Netze.

3.2.2 Unsupervised ML-Systeme

Im Gegensatz zu supervised ML-Systemen brauchen unsupervised ML-Systeme keine aufbereiteten Daten. Sie sollen Strukturen bzw. Eigenschaften direkt aus den zugrunde liegenden Daten lernen. Daher liegt das Anwendungsgebiet im Clustering oder der Dimensionsreduktion zur Visualisierung von Daten. Ein weiteres Anwendungsgebiet ist die Verarbeitung natürlicher Sprache. Unsupervised ML-Systeme sind hier von herausragender Bedeutung, da die Bedeutung von Wörtern ohne vorherige Annotation gelernt werden kann. So sind ML-Systeme in der Lage einzelne Worte oder Sätze in einen Vektor zu transformieren, der wiederum als Eingangsvektor für weitere ML-Verfahren genutzt werden kann. Diese *Wortvektoren* (siehe Kapitel 4) können über Verfahren zur Dimensionsreduktion (Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE)) visuell dargestellt werden.

3.2.3 Generalisierungsstrategien für ML

Bei der Auswahl des richtigen Ansatzes für ein ML-System ist die Generalisierungsstrategie entscheidend. In den meisten Anwendungsbereichen ist das Ziel von ML, nach dem Training mit bekannten Daten eine Vorhersage auf neue, unbekannte Daten zu treffen (Inferenz). Im Wesentlichen sind zwei Strategien zu unterscheiden:

- **Instanzbasiert:** Beim instanzbasierten Lernen lernt ein ML-Algorithmus anhand von Beispieldaten im Trainingsdatensatz, indem unbekannte Daten direkt mit den Beispieldaten verglichen werden. Dazu muss ein Ähnlichkeitsmaß auf den Daten definiert werden. Ein instanzbasiertes System würde zuerst die enthaltenen Worte in einer E-Mail in Wortvektoren transformieren und anschließend das Ähnlichkeitsmaß mit den Trainingsdaten abgleichen. Die Vorhersage ergibt sich aus der größten Ähnlichkeit.
- **Modellbasiert:** Ein einfaches Beispiel einer modellbasierten Strategie ist die lineare Regression $m(x) = \theta_0 + \theta_1 \cdot x$. Das Modell m besitzt zwei Parameter (θ_0, θ_1), die anhand der Trainingsdaten optimiert werden müssen, um Vorhersagen zu ermöglichen. Die Bestimmung der Parameter erfolgt über die Optimierung

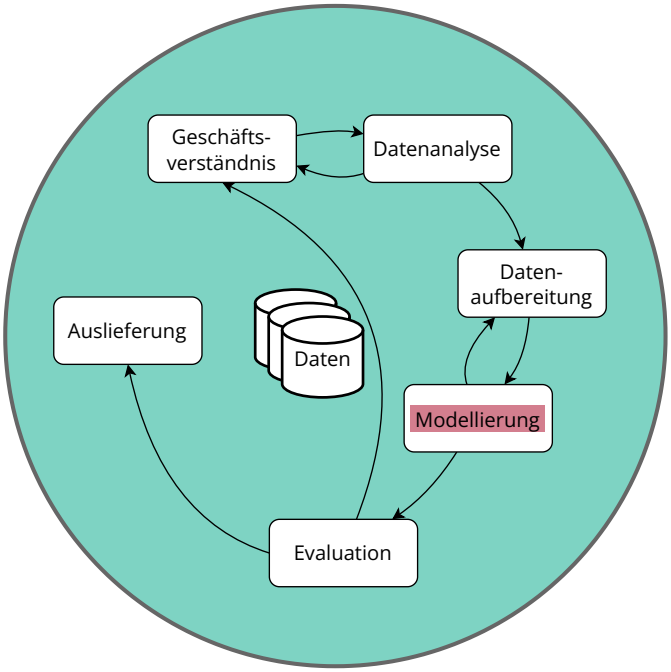


Abbildung 3.2: Industrieübergreifenden Standardprozess für Data Mining (CRISP-DM) [39]

von einer vorgegebenen Kosten- (z. B. Abstand der Punkte zur einer linearen Regressionsgeraden) bzw. Nutzenfunktionen.

3.3 Entwicklung von ML-Systemen

Das Vorgehen bei der Entwicklung datengetriebener Systeme, zu denen auch ML-Systeme zählen, unterscheidet sich von klassischen (und agilen) Vorgehensmodellen zur Softwareentwicklung. Abbildung 3.2 zeigt das Vorgehen nach dem industrieübergreifenden Standardprozess für Data Mining (CRISP-DM) [39] ohne Erweiterungen. Grundsätzlich kann CRISP-DM auch auf die Entwicklung von ML-Systemen angewandt werden. In diesem grundlegenden Vorgehensmodell besteht die Entwicklung eines Data Mining (DM) Systems aus sechs Phasen:

- **Geschäftsverständnis:** In dieser Phase sollen die Projektziele aus wirtschaftlicher und fachlicher Sicht verstanden werden, um diese Ziele als DM Problem zu formulieren.

- **Datenanalyse:** Bei der Datenanalyse werden zunächst Datenquellen beschrieben und ein initialer Datensatz erstellt. In diese Phase fallen alle Aktivitäten, die dazu dienen, die Daten besser kennenzulernen, wie das Finden von Datenqualitätsproblemen oder das Gewinnen erster Erkenntnisse. Aus einem besseren Verständnis der Rohdaten folgt gleichzeitig ein verbessertes Verständnis der eigentlichen Geschäftsanforderungen.

- **Datenaufbereitung:** Die Datenaufbereitung enthält alle Aufgaben, die auch ETL-Prozesse enthalten. Dazu zählen unter anderem die Bereinigung der Daten von Qualitätsproblemen, das Konstruieren neuer kombinierter Eigenschaften, das Annotieren der Daten oder weitere Transformationen.

- **Modellierung:** Anschließend können Modelle gebildet werden, die auf die aufbereiteten Daten angewandt werden. Bei der Entwicklung von ML-Modellen wird der Datensatz (häufig auch als Korpus bezeichnet) in der Regel in drei kleinere Datensätze aufgeteilt: den Trainingsdatensatz, der ca. 80 % der Daten enthält, einen Validierungsdatensatz (ca. 10 %), und einen Testdatensatz (ca. 10 %). Anschließend wird ein Modell anhand des Trainingsdatensatzes trainiert. Nach dem Training wird das entstandene Modell über verschiedene Metriken auf dem Testdatensatz evaluiert.

- **Evaluation:** In der Evaluation erfolgt die Bewertung des entwickelten Modells anhand der gesetzten Geschäftsziele, nicht der ML-Metriken. Bevor ein Modell ausgeliefert wird, muss geprüft werden, ob alle Anforderungen erfüllt sind, oder ob Anwendungsfälle vergessen wurden. Sollten alle Anforderungen erfüllt sein, kann das Modell ausgeliefert werden.

- **Auslieferung:** Nach der Erstellung des Modells muss es nutzbar gemacht werden. Dazu sind häufig zusätzliche Arbeitsschritte notwendig. Je nach Zweck des Modells reicht die weitere Arbeit von der Erstellung eines Reports über die Entwicklung eines Dashboards bis hin zur Einbettung des Modells in einen Webservice. Die Einbettung in einen Webservice erfordert einen vollständig reproduzierbaren Prozess, bei dem neben dem Modell auch alle Schritte der Datenaufbereitung implementiert sind. Erst nach der Auslieferung entsteht ein tatsächlicher, geschäftlicher Nutzen.

Der größte Unterschied zwischen DM-Prozessen und ML-Systemen liegt in der Modellierung und in der anschließenden Verwendung der Modelle. Ein DM-Prozess

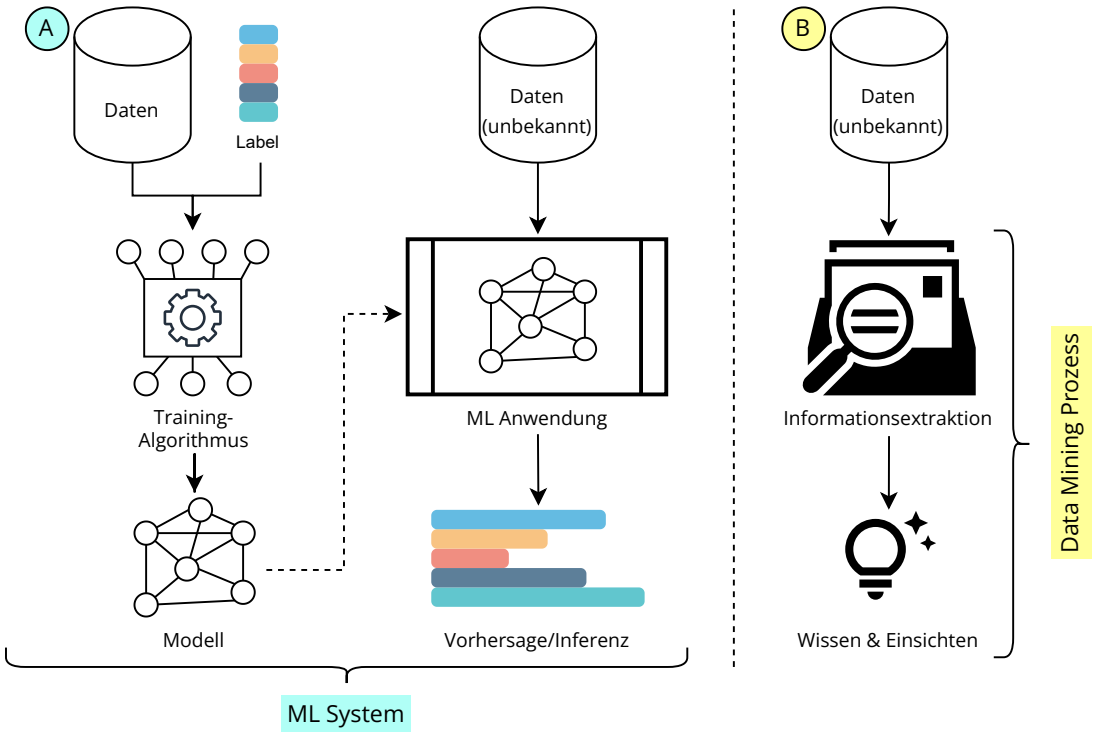


Abbildung 3.3: Phasen bei der Entwicklung eines ML-Systems (A) im Vergleich zur Entwicklung in einem Data-Mining-Prozess (B).

hat das Ziel, Wissen und Einsichten aus bislang nicht analysierten Daten zu generieren (vgl. Abbildung 3.3 (B)). Dazu müssen, wie zuvor beschrieben, Daten analysiert werden, um Informationen extrahieren zu können. Nachdem die Extraktion abgeschlossen ist, müssen die Ergebnisse aufbereitet werden, um letztendlich neue Einsichten zu gewinnen. Ein ML-System wird nach der Entwicklung für weitere Vorhersagen eingesetzt, während ein DM-Prozess sich auf die Informationsgewinnung konzentriert.

Bei ML-Systemen muss die Entwicklung in zwei Phasen aufgeteilt werden (vgl. Abbildung 3.3 (A)). In der ersten Phase wird ein Training-Algorithmus ausgewählt, der auf bekannte Daten angewendet wird. Für Supervised-Algorithmen werden die Daten zusätzlich mit Labeln versehen. Das Ergebnis dieses Trainings ist das Modell. In der zweiten Phase wird dieses Modell in eine Anwendung eingebettet, um auf unbekannten Daten vorhersagen zu treffen. Ein Modell mit seinen Vorhersagen kann auch als Teil eines DM-Prozesses eingesetzt werden.

Industrieübergreifender Standardprozess für DM (CRISP-DM) berücksichtigt nicht die besonderen Anforderungen von ML-Systemen, sobald diese außerhalb eines DM-Prozesses, der nach der Auslieferung endet, eingesetzt werden. ML-Systeme müssen, einmal im produktiven Einsatz, überwacht und – wie sämtliche produktive Software – regelmäßig gewartet werden. Diese zusätzliche Phase wird von CRISP-ML(Q) [38] adressiert. Da die im nächsten Abschnitt beschriebenen Herausforderungen nicht nur in der Entwicklung von ML-Systemen auftreten können, sondern auch zu einem beliebigen Zeitpunkt während das Modell genutzt wird, ist CRISP-ML(Q) bei der Entwicklung von ML-Systemen vorzuziehen.

3.4 Typische Herausforderungen

Während der beschriebenen Entwicklungsphasen und -zyklen müssen in jedem ML-Projekt verschiedene Herausforderungen gelöst werden. Diese können sich sowohl auf die verwendeten Algorithmen als auch auf die vorhandenen Daten beziehen.

3.4.1 Unzureichende Trainingsdaten

Liegt ein Problem mit den Trainingsdaten eines ML-Systems vor, ist die Ursache häufig in den folgenden Kategorien zu finden:

- **Unzureichende Menge:** Um ein ML-System zu trainieren, müssen Daten in ausreichender Menge vorliegen. Die Anzahl der Beispiele im Trainingsdatensatz beeinflusst die Wahl des Algorithmus. Banko und Brill konnten zeigen, dass die Wahl des Algorithmus ab einer gewissen Anzahl nur einen geringen Einfluss auf die Vorhersagequalität besitzt [40]. Zu ähnlichen Schlüssen kommen Halevy et al. [41]. Trotzdem sind gerade in der Industrie kleine und mittelgroße Datensätze, die diese kritische Masse nicht überschreiten, der Regelfall. Bei diesen Datensätzen spielt daher die Entwicklung und Anwendung geeigneter Algorithmen eine große Rolle. Insbesondere bei der Verarbeitung natürlicher Sprache kann zwar auf eine große Menge frei verfügbarer Daten bzw. Artikel zugegriffen werden (z.B. Wikipediaartikel), allerdings müssen die Algorithmen im Einsatz bei Unternehmen auf Fachsprache funktionieren, die sich erheblich von der verwendeten Sprache in offenen Datensätzen unterscheidet.

- **Mangelnde Qualität:** Liegen Daten in ausreichender Menge vor, kann dennoch deren Qualität zu schlechten Ergebnissen führen. Im Falle von Produktdaten sind unter anderem Kodierungsunterschiede, Rechtschreibfehler und unterschiedliche Standards zu nennen. Je mehr dieser Probleme in den Daten auftauchen, desto schlechter ist die zu erwartende Vorhersagequalität. Typische Strategien zur Vermeidung dieses Problems sind manuelle Korrekturen oder das Erkennen von Ausreißern im Datensatz.

- **Mangelnde Repräsentativität:** Damit ein Modell für den produktiven Einsatz geeignet ist, müssen die Trainingsdaten den Anwendungsfall repräsentieren. Nur so kann ein Modell gut generalisieren. Ist ein Datensatz zu klein, kann dieser durch Zufall nicht repräsentativ sein (*sampling noise*). Dies kann beispielsweise der Fall sein, wenn als Trainingsdatensatz einfach die letzten 100 empfangenen E-Mails eines Postfachs gewählt werden und darin zufällig keine Spam-Nachrichten enthalten sind. In diesem Fall kann ein System niemals lernen, wie eine Spam-Nachricht aussieht. Eine weitere Fehlerquelle ist der *sampling bias*. Dieser tritt z.B. ein, wenn der Datensatz zwar Spam-Nachrichten enthält, aber ausschließlich Werbung für Potenzmittel. Somit enthält der Datensatz nicht die gesamte Bandbreite an möglichen Spam-Nachrichten, sondern ist auf eine bestimmte Art Nachricht beschränkt. Ein repräsentativer Datensatz sollte daher möglichst viele auftretende Varianten von Spam- und Nicht-Spam-Nachrichten enthalten, aber gleichzeitig keine der Klassen bevorzugen, damit ein ML-System die Unterschiede zwischen diesen beiden Klassen lernen kann. Ein solcher Datensatz wird als balanciert bezeichnet. Ist der Datensatz unbalanciert, also eine Klasse überproportional im Datensatz enthalten, besitzen Vorhersagen von ML-Systemen häufig eine Tendenz zu dieser Klasse.

3.4.2 Irrelevante Merkmale

Neben unzureichenden Daten kann das Lernen von irrelevanten Merkmalen dazu führen, dass ein ML-System schlecht generalisiert. Die Wahl der zu lernenden Merkmale ist daher ein kritischer und häufig zeitintensiver Prozess.

3.4.3 Überanpassung

Eine Überanpassung tritt dann ein, wenn ein ML-System auf den Trainingsdaten hervorragend arbeitet, aber trotzdem schlecht generalisiert. In diesem Fall wurden

Muster in den Trainingsdaten entdeckt, beispielsweise im Grundrauschen der gelernten Merkmale vorhanden sind. Alternativ kann auch der Trainingsdatensatz zu klein sein, sodass ein Modell lediglich die vorhandenen Daten auswendig lernt. Neben Ursachen in den Daten kann eine schlechte Modellierung zur Überanpassung führen. Ein Beispiel hierfür ist die Verwendung eines hochgradigen Polynoms anstelle eines linearen Modells für eine Regression. Je mehr Parameter das Polynom besitzt, desto eher können alle Punkte aus den Trainingsdaten genau getroffen werden. Das abgeleitete Modell generalisiert in der Folge jedoch schlecht. Mögliche Ansätze, um eine Überanpassung zu verhindern sind:

- Das Modell vereinfachen (weniger Parameter),
- Parameter des Modells an Bedingungen knüpfen und so deren Wertebereich einschränken (*Regularisieren*),
- mehr Trainingsdaten nutzen
- oder Qualitätsprobleme und Rauschen in den vorhandenen Daten reduzieren.

3.4.4 Unteranpassung

Im Gegensatz zur Überanpassung lernt ein Modell bei einer Unteranpassung zu wenig über die Muster in den Trainingsdaten, um zu generalisieren. Häufig tritt eine Unteranpassung ein, wenn ein Modell zu einfach ist. Eine lineare Regression ist zum Beispiel häufig nicht in der Lage die komplexe Realität abzubilden. Um eine Unteranpassung zu vermeiden, bleiben hauptsächlich folgende Optionen:

- Ein Verfahren mit mehr Parametern wählen,
- bessere Merkmale auswählen, die das Modell lernen soll
- oder weniger Regularisieren, falls bereits Wertebereiche eingeschränkt wurden.

Erfahrungsgemäß tritt eine Unteranpassung hauptsächlich auf, wenn die Trainingsdaten qualitativ zu schlecht sind, um Muster zu erkennen, oder ein Modell bereits zu stark eingeschränkt wurde.

3.5 Schemazuordnung durch maschinelles Lernen

Neben klassischen, regelbasierten Ansätzen zur Schemazuordnung können auch ML-basierte Ansätze gewählt werden. Die folgenden Abschnitte behandeln grundlegende ML-Verfahren zur Umsetzung der Schemazuordnung über ML, sowie die Bewertung der Ergebnisse der ML-Verfahren.

Um Schemazuordnungen zu lernen, müssen zunächst Daten gesammelt werden, aus denen Merkmale extrahiert werden können, mit denen ein ML-System trainiert werden kann. Diese Daten müssen neben den ursprünglichen Eingangsdaten auch die korrekten Zuordnungen zwischen Eingangs- und Zielschema enthalten. Aus der Sicht eines ML-Systems handelt es sich in der Regel um eine Klassifikation, da ein oder mehrere Attribute des Eingangsschemas einem oder mehreren Attributen des Zielschemas zugeordnet werden müssen.

Abbildung 3.4 zeigt den grundsätzlichen Ablauf eines ML-gestützten Matching-Systems nach dem erfolgten Training für das Attribut `name` aus dem Eingangsschema. Aus diesem Attribut lassen sich im Beispiel verschiedene Merkmale aus dem Attributnamen extrahieren. Dazu zählen unter anderem der Datentyp, in diesem Fall `Text`, die Länge des Attributnamens, die Struktur des Attributs oder sonstige, evtl. auftretende Muster. Transformiert in einen *Feature-Vektor* können diese Merkmale als Eingabe für das ML-System genutzt werden, das auf dieser Grundlage die wahrscheinlichste Zielklasse vorhersagt. Zu der Vorhersage zählt auch ein entsprechender Konfidenzwert, der angibt, wie sicher sich das ML-System bei dieser Vorhersage ist. Je nach Datenmenge, -aufbereitung usw. kommen verschiedene Verfahren in Betracht, die innerhalb des ML-Sys-tems zur Klassifikation genutzt werden können. Diese werden im Folgenden beschrieben.

3.5.1 Support Vector Machines

Support Vector Machines (SVMs) fallen in die Klasse des *Supervised*-Lernens. Sie können unter anderem für Klassifikationen oder Regressionen genutzt werden [42]. Eine einfache SVM kann zwischen genau zwei Klassen unterscheiden, indem das lineare Modell aus Gleichung (3.1) [43] verwendet wird.

$$y(x) = w^T \phi(x) + b \tag{3.1}$$

Dabei bezeichnet w die Gewichte, $\phi(x)$ die Transformation in den Merkmalsraum und b den Bias. Der Trainingsdatensatz für die SVM besteht aus N Eingangsdaten

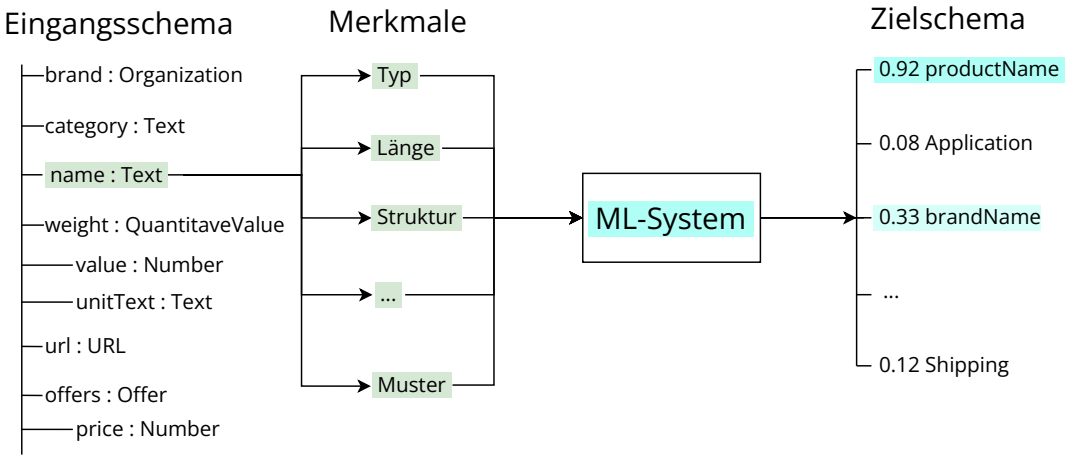


Abbildung 3.4: Schemazuordnung mithilfe von ML-Systemen. Die Attributnamen des Zielschemas werden mit der jeweiligen Konfidenz des ML-Systems dargestellt.

$X = \{x_1, \ldots, x_N\}$ mit den zugehörigen Zielklassen t_1, ..., t_N mit $t_n \in \{-1, 1\}$ aus den Rohdaten. Neue Eingangsdaten, die nicht im Trainingsdatensatz enthalten sind, können dann über $sign(y(x))$ klassifiziert werden. Ist das Ergebnis der Entscheidungsfunktion $y(x)$ positiv (≥ 0), wird die eine Klasse angenommen, ansonsten (< 0) die andere.

Die Entscheidungsgrenze einer einfachen SVM verläuft in diesem Fall mit maximalem Abstand von allen Instanzen der beiden Klassen und somit in der Mitte. Diese Art der Klassifikation wird als *harte* oder *Maximum-Margin-Klassifikation* bezeichnet. Sie funktioniert nur, wenn die beiden Klassen strikt linear separierbar sind. Daher ist diese Art der Klassifikation anfällig für Ausreißer.

Die Berechnung der Entscheidungsgrenze erfolgt über die Lösung der Gleichung

$$\arg\max_{w,b} \left\{ \frac{1}{||w||} \min_n [t_n y(x_n)] \right\}. \tag{3.2}$$

Durch die Umformulierung von Gleichung (3.1) über den Kernel-Trick und quadratische Programmierung lässt sich die Entscheidungsfunktion weiter zu Gleichung (3.3) umformen [43]. Dies ermöglicht auch die Unterscheidung nicht linear separierbarer Klassen. Dabei werden die Eingangswerte X in einen höherdimensionalen Raum transformiert, in dem diese ursprünglich nicht linear separierbaren Werte einfacher zu teilen sind.

$$y(x) = \sum_{n=1}^{N} t_n \alpha_n K(x, x_n) + b \tag{3.3}$$

Linear:	$K(a,b)$	=	$a^T \cdot b$
Polynomial:	$K(a,b)$	=	$(\gamma a^T \cdot b + r)^d$
radiale Basisfunktion (RBF):	$K(a,b)$	=	$exp(-\gamma \|\|a-b\|\|^2)$
Sigmoid:	$K(a,b)$	=	$tanh(\gamma a^T \cdot b + r)$

Tabelle 3.1: Häufig genutzte Kernel für SVM [43]

In Gleichung (3.3) steht α_n für einen Lagrange-Multiplikator für den zugehörigen Punkt aus den Trainingsdaten mit $\alpha_n \geq 0$ und $\sum t_n \alpha_n = 0$. Nur die Beispiele aus den Trainingsdaten (x_n), die in der Nähe der Entscheidungsgrenze liegen, besitzen ein $\alpha > 0$. Häufig genutzte Kernel K finden sich in Tabelle 3.1.

Dennoch sind SVMs ohne weitere Hyperparameter nicht in der Lage, überlappende Klassen zu erlauben. Mit kleinen Ergänzungen ist jedoch auch eine *Soft-Margin-Klassifikation* möglich, bei der Klassen sich bis zu einem gewissen Grad überlappen dürfen. Dadurch werden SVMs resistenter gegenüber Ausreißern [44].

Systeme, die SVMs bereits zur Klassifikation im Rahmen des Schema-Matchings genutzt haben, finden sich in Kapitel 9. Grundsätzlich bieten sich die folgenden Ansätze an:

Binäre-Klassifikation Die Instanzen der Trainingsdaten werden in genau zwei Kategorien eingeteilt. Entweder eine Zuordnung existiert oder nicht. Bezugnehmend auf Abbildung 3.4 bedeutet dies, dass die Merkmale des Attributs `name` aus dem Eingangsschema nur im Falle von `productName` in die Klasse „Zuordnung existiert“ fällt. Dabei müssen als Eingangsdaten in die SVM Eingangs- und Zielattribut gewählt werden, sodass sämtliche Attribute permutiert werden müssen, bis die Schemazuordnung abgeschlossen ist. Der Vorteile gegenüber der mehrklassigen Klassifikation ist, dass nur ein einziges Modell trainiert werden muss.

Mehrklassen-Klassifikation Bei der mehrklassigen Klassifikation existieren verschiedene Strategien für SVMs. Die am häufigsten genutzten sind „one-vs-rest“ und „one-vs-one“.

Bei der „one-vs-rest“ Strategie werden so viele SVMs trainiert, wie Zielklassen K existieren. Das Endergebnis wird durch die Heuristik $y(x) = \max_k y_k(x)$ bestimmt, wobei k die k-te Klasse bezeichnet [43]. Diese Heuristik löst inkonsistente Ergebnisse der einzelnen SVMs auf. Beim Training wird jede SVM mit der ihr zugehörigen

Zuordnung als Positiv- und den anderen Zuordnungen als Negativbeispiel trainiert. Dadurch sind die Klassen im jeweiligen Trainingsprozess unbalanciert, selbst wenn sie im Trainingsdatensatz gleich verteilt sind. Im Gegensatz zur binären Klassifikation ist hier jede SVM auf genau eine Zielklasse spezialisiert.

Die „one-vs-one"-Strategie bildet eine Zwei-Klassen SVM pro Zielklassenpaar. Somit müssen $K(K-1)/2$ verschiedene SVMs trainiert werden. Das Ergebnis der Klassifikation ist die Klasse, die am häufigsten vorhergesagt wurde. Dieses Vorgehen wird auch als *Voting* bezeichnet. Trotzdem kann dieses Vorgehen zu Mehrdeutigkeiten führen (z.B. bei Gleichstand), die einer weiteren Auflösung bedürfen. Ein weiteres Problem bei der „one-vs-one"-Strategie ist die signifikant erhöhte Komplexität und Berechnungsdauer bei vielen Zielklassen. Dies ist insbesondere dann der Fall, wenn das Zielschema eine hohe Anzahl von Attributen besitzt.

3.5.2 Neuronale Netze

Künstliche neuronale Netze (NN) sind dem Aufbau menschlicher Nervenverbindungen nachempfunden und fallen in die Klasse des *Supervised*-Lernens. Ein NN besteht aus einzelnen Knoten, den Neuronen, deren Verbindungen untereinander gewichtet werden. Gleichung (3.4) stellt ein einfaches Neuron als *Linear Threshold Unit (LTU)* dar. Dabei werden alle Eingangswerte $x_1 \dots x_n$ mit den zugehörigen Gewichten $w_1 \dots w_n$ aufsummiert und das Ergebnis über eine Aktivierungsfunktion f berechnet [43].

$$y(x) = f(\sum_{i=1}^{n} w_i x_i) = f(w^T x) \tag{3.4}$$

Die Aktivierungsfunktion für eine LTU ist eine Treppenfunktion, wie die *Heavyside-Funktion* aus Gleichung (3.5). Für $a \geq 0$ wird die LTU aktiviert.

$$f(a) = \begin{cases} 1, & a \geq 0 \\ 0, & a < 0 \end{cases} \tag{3.5}$$

Werden nun mehrere Neuronen in verschiedene Ebenen eines Netzwerks gruppiert, lassen sich komplexe NNs bilden. Sie bestehen in der Regel aus einer Ebene für Eingangswerte (Input-Layer), einen Output-Layer für die Ergebnisse und beliebig viele Hidden-Layer, die Input- und Output-Layer miteinander verbinden.

Eine einfache Form eines NN ist das Perzeptron. Es besteht aus einem einzigen Output-Layer von LTUs. Die Anzahl der verwendeten LTUs hängt von der Aufga-

be des Perzeptron ab. Bereits eine einzelne LTU reicht aus, um eine Zwei-Klassen-Klassifikation ($t \in 0, 1$) zu ermöglichen. Für eine Mehrklassen-Klassifikation muss der Output-Layer eine LTU pro Zielklasse besitzen. Als Input-Layer werden sogenannte *passthrough* Neuronen genutzt, die die Eingangswerte lediglich an den nächsten Layer (Output-Layer) weiterreichen. Zusätzlich wird der Input-Layer um ein Bias-Neuron, dessen Ausgangswert immer eins ist, erweitert. Das Bias-Neuron ermöglicht die Verschiebung der Aktivierungsfunktion nach rechts oder links. Vereinfacht entspricht es der y-Verschiebung b in einer linearen Funktion $f(x) = wx + b$. Als Eingangswerte für den Input-Layer dienen die extrahierten Merkmalsvektoren ($\phi(x)$). An dieser Stelle ist bereits ein Vorteil von NNs gegenüber SVMs zu erkennen. Im Gegensatz zu SVMs kann ein NN so konstruiert werden, dass die Mehrklassen-Klassifikation unterstützt wird, ohne mehrere NNs zu trainieren [43].

Allerdings kann ein Perzeptron ebenso wie eine SVM mit linearem Kernel nur linear klassifizieren. Um auch komplexere Problemstellungen lösen zu können, lässt sich das Perzeptron-Modell erweitern. Das sog. Multi-Layer Perzeptron (MLP) besteht zusätzlich aus einem Hidden-Layer von LTUs und einem Bias-Neuron, der zwischen Input- und Output-Layer liegt. Die Berechnung der Gewichte erfolgt im Training über den Backpropagation-Algorithmus [45].

Vereinfacht beschrieben, wird für jede Instanz aus dem Trainingsdatensatz zunächst eine Vorhersage durch das Netzwerk berechnet, danach der Fehler gemessen und im Anschluss rückwärts jede Verbindung des vorangegangenen Layers betrachtet, um zu ermitteln, welche Verbindung wie viel zum Gesamtfehler beigetragen hat. Abschließend werden die Gewichte der Verbindungen so angepasst, dass der Gesamtfehler reduziert wird, z.B. über das Gradient Descent Verfahren [43].

Eine weitere notwendige Veränderung gegenüber dem einfachen Perzeptron ist der Austausch der Aktivierungsfunktion. Da die Heavyside-Funktion keinen Gradienten (Steigung = 0) besitzt, kann das Gradient Descent Verfahren nicht zur Optimierung der Gewichte genutzt werden. Aus diesem Grund werden in der Praxis Aktivierungsfunktionen, wie die logistische Funktion $\sigma(z) = 1/(1 + \exp(-z))$ oder $\text{ReLU}(z) = \max(0, z)$ (rectified linear unit (ReLU)) genutzt.

Im Schema- und Ontologie-Matching werden MLP bzw. NN hauptsächlich zur Klassifikation (eine einzelne Zielklasse) oder Kategorisierung (erlaubt mehrere Zielklassen gleichzeitig) verwendet, um Zuordnungen zwischen Eingangs- und Zielschema zu ermitteln. Eine Auswahl von Ansätzen zum Schema-Matching mithilfe von NNs findet sich in Kapitel 9.

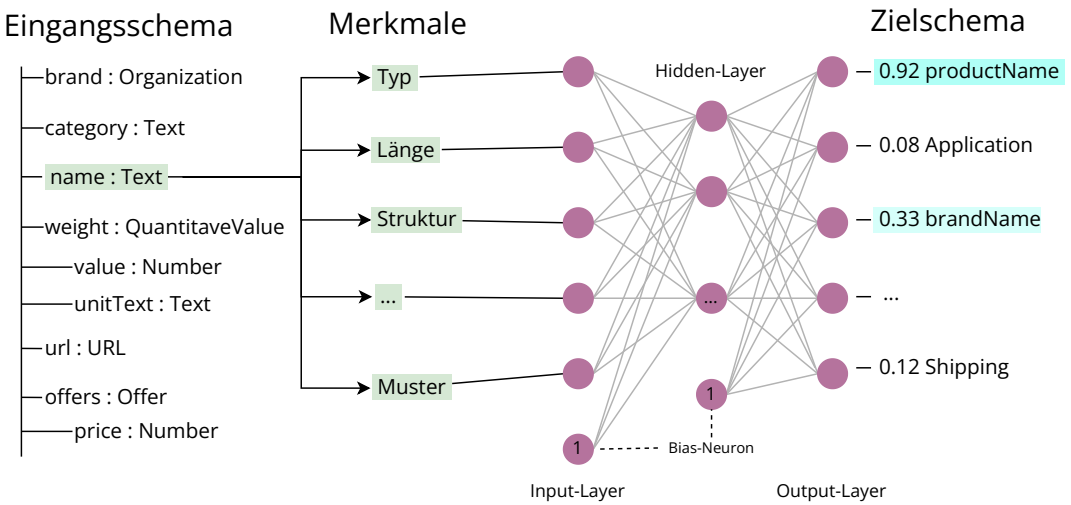


Abbildung 3.5: Schemazuordnung mithilfe eines MLP. Die Merkmale des Attributs `text` ***werden als Eingangswerte genutzt, während die Attribute aus dem Zielschema den Neuronen im Output-Layer entsprechen.***

Abbildung 3.5 stellt dar, wie ein MLP als Klassifikator genutzt werden kann, um Zuordnungen zwischen einem Eingangs- und einem Zielschema zu ermitteln. Dabei werden die Merkmale eines Attributes aus dem Eingangsschema (*name*) als Eingabewerte für den Input-Layer des MLP genutzt. Im Output-Layer steht jeweils ein Knoten für ein Attribut des Zielschemas, sodass direkt eine Mehrklassen-Klassifikation erfolgen kann. Im Gegensatz zur SVM genügt das Training eines einzelnen NN für eine Mehrklassen-Klassifikation. Die Ebenen des NN sind vollständig miteinander vernetzt und werden, wo notwendig, durch Bias-Neuronen ergänzt. Um eine vollständige Schemazuordnung durchzuführen, muss die Klassifikation für alle Attribute aus dem Eingangschema erfolgen.

3.5.3 Entscheidungsbäume

Bei einer Klassifikation durch Entscheidungsbäume werden anhand der Merkmale Regeln abgeleitet, die sequenziell durchlaufen werden und letztendlich zu einer Entscheidung führen. Im Gegensatz zu den zuvor vorgestellten Klassifikationsverfahren sind die Entscheidungen von Entscheidungsbäumen direkt interpretierbar. In der Trainingsphase wird der Classification And Regression Tree (CART) Algorithmus genutzt. Dabei wird der Trainingsdatensatz in zwei Teilmengen anhand des Merkmals k und einem Grenzwert t_k geteilt.

Das Paar aus Merkmal und Grenzwert (k, t_k) wird so gewählt, dass die Teilmengen gewichtet nach ihrer Größe möglichst *rein* sind. Die Reinheit wird beispielsweise über den Gini-Koeffizienten in Gleichung (3.6) definiert. Eine Teilmenge i wird als rein bezeichnet, wenn $G_i = 0$. Dies bedeutet, dass eine Teilmenge vollständig zu einer Klasse c gehört. Zur Berechnung werden die Anteile ($p_{i,c}$) aller Datenpunkte der Teilmenge, die in die Klasse c fallen, genutzt. Ist eine Teilmenge rein, sind alle Anteile $p_{i,c} = 0$, abgesehen von genau einem c. Der Gini-Koeffizient lässt sich dann über Gleichung (3.6) für jede Teilmenge i berechnen, wobei n die Anzahl der Klassen ist.

$$G_i = 1 - \sum_{c=1}^{n} p_{i,c}^2 \tag{3.6}$$

Der CART Algorithmus optimiert das Aufteilen in die einzelnen Teilmengen über die Kostenfunktion 3.7. Wurde erfolgreich ein Minimum gefunden, werden die Teilmengen erneut wie beschreiben aufgeteilt, bis die maximal eingestellte Tiefe des Baums erreicht wurde. Dieser Parameter ist bei Entscheidungsbäumen einer der Hyperparameter, der optimiert werden kann.

$$J(k, t_k) = \frac{m_{\text{links}}}{m} G_{\text{links}} + \frac{m_{\text{rechts}}}{m} G_{\text{rechts}} \,, \quad \text{wobei} \begin{cases} G_{\text{links/rechts}} \text{ für die (Gini-)Reinheit einer Teilmenge steht.} \\ m_{\text{links/rechts}} \text{ die Anzahl der Elemente einer Teilmenge ist.} \end{cases} \tag{3.7}$$

Ein Nachteil des Gini-Koeffizienten kann bei unbalancierten Datensätzen auftreten, da die Optimierung über den Gini-Koeffizienten dazu tendiert, große Klassen in einen einzigen Teilbaum zu isolieren.

Auch Entscheidungsbäume wurden bereits in verschiedenen Matching-Systemen genutzt (vgl. Kapitel 9). Abbildung 3.6 zeigt die Anwendung eines Entscheidungsbaumes nach erfolgtem Training. Zur Klassifikation der Zuordnung zwischen Eingangsschema und Zielschema werden die im Training gelernten Regeln eingesetzt. In Abbildung 3.6 wird für das Attribut *name* das zugehörige Attribut aus dem Zielschema gesucht. Dafür wird der vereinfachte Entscheidungsbaum durchlaufen. Analog zur Klassifikation mit NN muss der Entscheidungsbaum für jedes Attribut aus dem Eingangsschema durchlaufen werden, um alle Zuordnungen zu bestimmen. Es müssen jedoch nicht mehrere Bäume trainiert werden, damit ein Entscheidungsbaum eine Mehrklassen-Klassifikation durchführen kann. Wie bei den NN reicht ein einzelnes trainiertes Modell zur Klassifikation mehrerer Zielattribute aus.

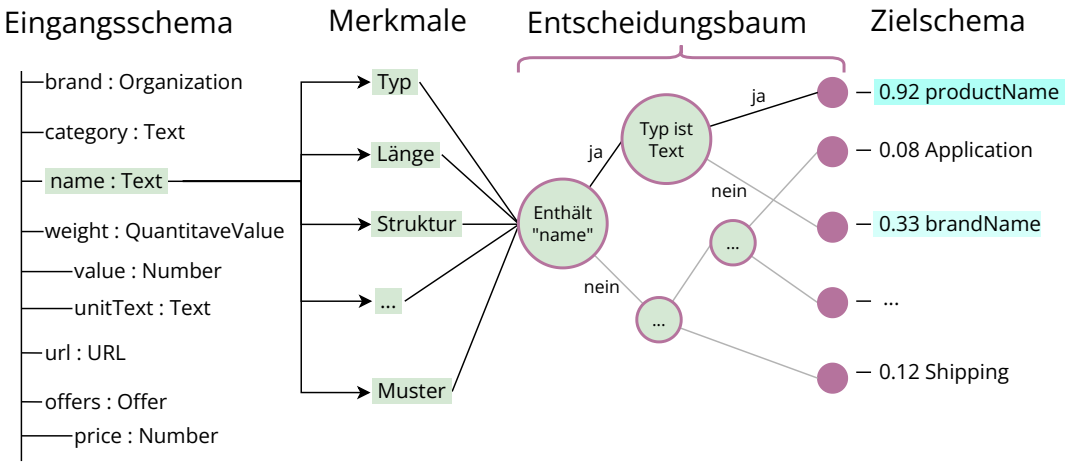


Abbildung 3.6: Schemazuordnung mithilfe eines Entscheidungsbaumes. Die Merkmale des Attributs `name` ***werden als Eingangswerte genutzt, um über gelernte Regeln die passende Zuordnung zu finden. Die gelernten Regeln beziehen sich eindeutig auf ein Merkmal. Somit sind Entscheidungen für Menschen jederzeit nachvollziehbar.***

3.5.4 Ensemble-Methoden

Ensemble-Methoden nutzen das Prinzip der Weisheit der Vielen. Hinter der Idee dieses Ansatzes steht die Annahme, dass Gruppen klüger sind als Individuen. Bezogen auf ML bedeutet dies, dass die Vorhersagen verschiedener ML-Verfahren aggregiert werden, um zu einer gemeinsamen Antwort zu gelangen. Diese Gruppe von Verfahren wird auch als Ensemble bezeichnet und liefert häufig bessere Ergebnisse als ein einzelnes Modell. In diesem Abschnitt werden die Methoden *Voting*, *Bagging* und *Random Forest* (eine Gruppe von Entscheidungsbäumen) vorgestellt.

Voting Die einfachste Ensemble-Methode entspricht einem Mehrheitsentscheid. Dabei werden mehrere Klassifikatoren genutzt, die auf dem gleichen Datensatz trainiert wurden. Abbildung 3.7 zeigt ein solches Szenario, in dem eine SVM, ein NN und ein Entscheidungsbaum trainiert wurden. Soll dann eine Vorhersage für einen unbekannten Datenpunkt durchgeführt werden, können die Ergebnisse der einzelnen Verfahren kombiniert werden.

In Abbildung 3.7 sagen zwei Verfahren *productName* und das NN *brandName* vorher. Die Vorhersage des Ensembles ist die Vorhersage mit den meisten Stimmen (Ergebnissen). Dieses Verfahren kann deshalb auch als *Hard Voting* bezeichnet wer-

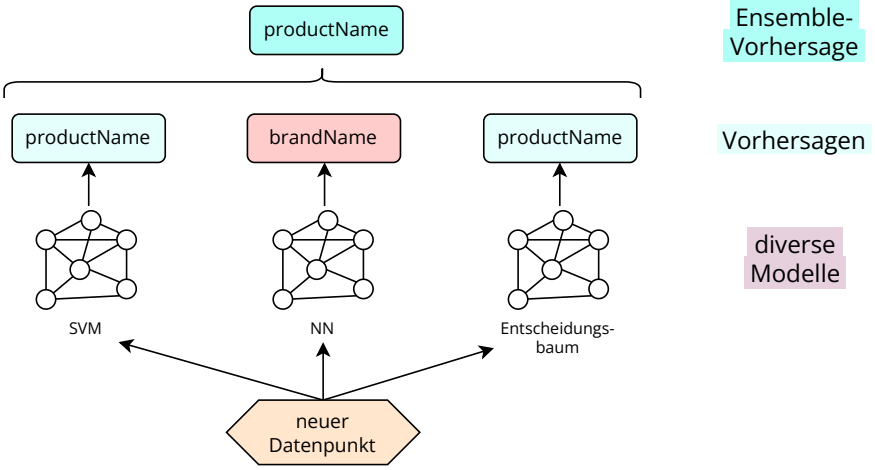


Abbildung 3.7: Hard voting von verschiedenen Vorhersagen

den, da eine einfache Mehrheit ausreicht und Konfidenzen der einzelnen Modelle nicht berücksichtigt werden. Werden auch die Konfidenzen berücksichtigt, wird der Mehrheitsentscheid als *Soft Voting* bezeichnet. Das Ergebnis der Vorhersage ist in diesem Fall die Klasse mit dem höchsten Mittelwert der Konfidenzen.

Grundsätzlich können beliebig viele einzelne Modelle im Ensemble genutzt werden. Es sollte allerdings darauf geachtet werden, dass die genutzten Modelle möglichst unabhängig voneinander sind, damit fehlerhafte Vorhersagen unkorreliert sind und nicht alle Modelle ähnliche Fehler machen. Dies kann beispielsweise erreicht werden, indem sehr unterschiedliche Klassifikationsverfahren genutzt werden.

Bagging Während beim einfachen *Voting* verschiedene ML-Verfahren auf dem gleichen Trainingsdatensatz angelernt wurden, verfolgt *Bagging* den gegenteiligen Ansatz: Das gleiche ML-Verfahren wird auf verschiedene Datensätze angewandt.

Dazu werden aus dem Trainingsdatensatz zufällig Beispiele bzw. Instanzen gezogen, wobei Beispiele mehrfach gezogen werden können, bis die neu erzeugten Datensätzen die gewünschte Anzahl von Beispielen enthalten. Ist der ursprüngliche Trainingsdatensatz im Vergleich zu den neu Erzeugten groß genug, kann davon ausgegangen werden, dass die Verteilung der Datensätze annähernd identisch ist (sog. *Bootstrapping*). Somit ähneln sich auch deren Vorhersagen.

Unter diesen Voraussetzungen lässt sich ein Ensemble von Modellen trainieren, das eine geringere Varianz besitzt, als ein einzelnes Modell. Die Vorhersage des

Ensembles wird abhängig vom genutzten Verfahren durch Hard- oder Soft-Voting ermittelt.

Ein großer Vorteil des Bagging ist die Skalierbarkeit des Trainings, da Modelle unabhängig voneinander trainiert und ausgewertet werden können. Dadurch wird die Parallelisierung erheblich vereinfacht. Nachteilig wirkt sich auf das Bootstrapping aus, dass Beispiele aus dem ursprünglichen Trainingsdatensatz existieren, die nie gezogen wurden. Somit hat kein Verfahren diese Beispiele je gesehen. Gleichzeitig kann dies aber auch als Vorteil genutzt werden, da kein explizit definierter Testdatensatz genutzt werden muss, sondern auf diese unbekannten Beispiele zurückgegriffen werden kann, um das Ensemble zu evaluieren.

Random Forests Als *Random Forest* wird ein Ensemble von Entscheidungsbäumen bezeichnet, das mithilfe von Bagging trainiert wurde. Allerdings werden die zu lernenden Regeln im Entscheidungsbaum nicht übergreifend optimiert. Die Regeln werden anhand einer zufällig gewählten Teilmenge aller Merkmale abgeleitet, die sich allerdings für jeden einzelnen Baum unterscheiden kann. Dadurch wird gleichzeitig die Diversität der Bäume erhöht, was zu besseren Ergebnissen im Voting führt.

3.6 Metriken zur Bewertung von Matching-Systemen

Um die ermittelten Zuordnungen von Matching-Systemen zu bewerten, müssen die Ergebnisse der automatisch abgeleiteten Zuordnungen anhand von objektiven Kriterien mit Referenzzuordnungen verglichen werden. Diese oft manuell festgelegten Referenzzuordnungen werden auch als *Gold-Standard* bezeichnet. Da das Ermitteln von Zuordnungen sehr eng mit einer Klassifikationsaufgabe von ML-Systemen verwandt ist, können Metriken, die eigentlich zur Beurteilung von Ergebnissen einer Klassifikation entwickelt wurden, auch für Schema-Matching-Systeme verwendet werden [46].

Die folgenden Abschnitte stellen häufig genutzte Metriken zur Bewertung von Klassifikationsergebnissen vor, die im weiteren Verlauf der Ausarbeitung genutzt werden.

	korrekte Zuordnung ist *productName*	korrekte Zuordnung ist nicht *productName*
Attribut wurde als *productName* klassifiziert	r_p	f_p
Attribut wurde nicht als *productName* klassifiziert	f_n	r_n

Tabelle 3.2: Kategorisierung der Ergebnisse eines Schema-Matching-Systems am Beispiel einer Zuordnung

3.6.1 Mögliche Ergebnisse einer Klassifikation

Bei einer binären Klassifikation können die Ergebnisse des Klassifizierers in vier Kategorien eingeteilt werden (vgl. Tabelle 3.2).

- **Richtig positiv (r_p):** Der Klassifizierer hat *richtig* vorhergesagt, dass die Eingangsdaten in eine Klasse (*positiv*) fallen.
- **Falsch positiv (f_p):** Der Klassifizierer hat *falsch* vorhergesagt, dass die Eingangsdaten in eine Klasse (*positiv*) fallen.
- **Richtig negativ (r_n):** Der Klassifizierer hat *richtig* vorhergesagt, dass die Eingangsdaten *nicht* in eine Klasse (*negativ*) fallen.
- **Falsch negativ (f_n):** Der Klassifizierer hat *falsch* vorhergesagt, dass die Eingangsdaten *nicht* in eine Klasse (*negativ*) fallen.

Zur weiteren Beurteilung von ML- und Matching-Systemen können diese Kategorien genutzt werden, um weitere Metriken abzuleiten und die Qualität eines Systems zu beurteilen.

3.6.2 Beurteilung von Klassifikationsergebnissen

Um die jeweiligen Metriken zu berechnen, müssen die Elemente pro Kategorie gezählt werden. Die Metriken sind so gewählt, dass daraus ein Wert im Intervall $[0, 1]$ berechnet wird, wobei 1 für einen zuverlässigen Algorithmus im Sinne der Metrik steht. Bei ML-Systemen kann dazu der Testdatensatz verwendet werden.

Das System muss dann für jede Instanz aus dem Datensatz eine Vorhersage treffen, die mit dem *Gold-Standard* verglichen wird. Die Vorhersage kann dann in eine

der vier Kategorien eingeteilt werden. Sollen sowohl ML-basierte Systeme als auch klassische Systeme miteinander verglichen werden, ist es sinnvoll, diesen Vergleich nur mit dem Testdatensatz durchzuführen.

Accuracy (Acc) Eine perfekte Klassifikation erzeugt weder falsch positive noch falsch negative Vorhersagen. Gleichung (3.8) zeigt die Genauigkeit oder Accuracy. Die Accuracy ist der Anteil von richtigen Vorhersagen bezogen auf die Gesamtheit aller Vorhersagen.

$$Acc = \frac{r_p + r_n}{r_p + f_p + r_n + f_n} \tag{3.8}$$

Diese Metrik ist insbesondere dann aussagekräftig, wenn der Datensatz ausbalanciert, also jede Klasse annähernd gleich häufig im Datensatz enthalten ist. Bei einer binären Klassifikation sind folglich auch ungefähr gleich viele richtig positive und richtig negative Beispiele im Datensatz enthalten (ein Beispiel gehört zu einer Klasse oder nicht).

Sind die Klassen unbalanciert, kann ein ML-Algorithmus die Klasse mit mehr Beispielen besser lernen. Je schlechter die Balance der Klassen ist, desto eher lernt der Algorithmus nur diese Klasse vorherzusagen. Dadurch wird die Accuracy-Metrik ebenfalls in diese Richtung verzerrt, wenn die häufiger vorkommende Klasse gut vorhergesagt wird.

Datensätze mit Bezug zu praktischen Problemen sind nahezu immer unbalanciert. Im Folgenden werden daher Metriken vorgestellt, die weniger anfällig für unbalancierte Datensätze und Vorhersagen sind.

Precision (P) Die Precision (Gleichung (3.9)) beschreibt, wie zuverlässig die vorhergesagten Ergebnisse in Bezug auf ihre Korrektheit sind.

$$P = \frac{r_p}{r_p + f_p} \tag{3.9}$$

Im Sinne der Precision-Metrik sind Vorhersagen gut, wenn keine falsch positiven Vorhersagen gemacht wurden ($P = 1$). $P = 0$ würde bedeuten, dass ein System ausschließlich falsche Vorhersagen getroffen hat.

Recall (R) Der Recall beschreibt die Vollständigkeit der Vorhersagen, wie in Gleichung (3.10) dargestellt. Idealerweise kann ein System Vorhersagen für alle Elemente des Datensatzes treffen. In diesem Fall würden keine falsch negativen Ergebnisse

entstehen, sondern $R = 1$. $P = 0$ würde bedeuten, dass alle Instanzen einer Klasse falsch zugeordnet wurden.

$$R = \frac{r_p}{r_p + f_n} \tag{3.10}$$

Der Recall wird häufig auch als Sensitivität oder *True Positiv Rate* bezeichnet, da er die korrekt klassifizierten Instanzen einer Klasse, im Verhältnis zur Gesamtzahl der Instanzen dieser Klasse beschreibt.

Der Nachteil bei der Betrachtung von P und R ist, dass sie im Gegensatz zur Accuracy für sich alleine genommen keine Aussagekraft haben. So kann beispielsweise ein System, dass genau ein Element einer Klasse richtig klassifiziert $P = 1$ erreichen, wenn es alle anderen Elemente in eine andere Klasse zugeordnet hat, da durch P keine falsch negativen Ergebnisse berücksichtigt werden. Daher muss eine weitere Metrik eingeführt werden, die P und R gleichermaßen berücksichtigt, um alleinstehend eine Aussagekraft zu besitzen.

F_β-Score Ein einfacher Ansatz P und R in eine kombinierte Metrik zu überführen, ist den Mittelwert in Gleichung (3.11) zu bilden. Dabei würden alle erzielten Werte das Gesamtergebnis gleichmäßig beeinflussen.

$$\frac{P + R}{2} \tag{3.11}$$

Es ist allerdings erstrebenswert, dass besonders niedrige P und R Werte den Wert der Metrik stärker beeinflussen. Eine Lösung hierfür ist das harmonische Mittel von P und R (F_1-Score, Gleichung (3.12)). Hierbei fallen niedrige Werte der einzelnen Metriken stärker ins Gewicht. Der F_1-Score eignet sich insbesondere zur Beurteilung von Systemen, bei denen Precision und Recall gleich wichtig sind.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} = \frac{r_p}{r_p + \frac{f_n + f_p}{2}} \tag{3.12}$$

Je nach Anwendungsfall kann es allerdings nützlich sein, P und R unterschiedliche zu gewichten. Im Bereich der Dunkelverarbeitung ist beispielsweise eine hohe Precision nützlicher, da die Vorhersagen als richtig betrachtet werden können und somit Prozesse anhand der Vorhersage weitgehend automatisierbar sind.

$$F_\beta = (1 + \beta)^2 \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \tag{3.13}$$

	Vorhersage		
Goldstandard	productName	Application	Shipping
productName	7	2	1
Application	3	8	0
Shipping	1	1	11

Tabelle 3.3: Konfusionsmatrix zwischen drei Attributen als Klassen.

Über F_β-Score in Gleichung (3.13) lässt sich diese Gewichtung erreichen. Häufig wird $\beta = 0.5$ verwendet, wenn P stärker gewichtet werden soll als R, indem der Einfluss von falsch negativen Ergebnissen reduziert wird, oder $\beta = 2$ im umgekehrten Fall.

Konfusionsmatrix Die Darstellung der Klassifikationsergebnisse in Tabelle 3.2 wird auch als Konfusionsmatrix bezeichnet. In Tabelle 3.2 wird jedoch nur die binäre Klassifikation abgebildet. Die Konfusionsmatrix lässt sich einfach auf die Mehrklassen-Klassifikation erweitern. Angenommen, es wird nicht mehr die binäre Klassifikation zwischen `productName` oder `nicht productName` verwendet, sondern es soll zwischen den Klassen `productName`, `Application` und `Shipping` unterschieden werden. Dann kann eine Konfusionsmatrix aussehen wie in Tabelle 3.3.

In der Konfusionsmatrix beschreibt jede Zeile wie oft eine Klasse richtig bzw. falsch ermittelt wurde. In Tabelle 3.3 wurden Attribute der Klasse `productName` siebenmal tatsächlich der Klasse `productName` zugeordnet und dreimal, hier $2 + 1$ nicht. Über die Konfusionsmatrix lassen sich die zuvor vorgestellten Metriken ebenfalls auf eine Mehrklassen-Klassifikation erweitern. Dazu müssen lediglich die Werte für r_p, f_p, r_n und f_n anhand der Matrix wie folgt bestimmt werden:

- **Richtig positiv (r_p):** Die Werte auf der Diagonalen der jeweiligen Klasse entsprechen den richtig positiven Beispielen ($r_p = 7$ in Tabelle 3.3 für die Klasse `productName`).
- **Falsch positiv (f_p):** f_p wird über die Summe aller Werte in einer Spalte, ausgenommen der Diagonalen berechnet. Für die Klasse `productName` ist $f_p = 4$.
- **Richtig negativ (r_n):** Als richtig negativ werden in der Konfusionsmatrix alle Einträge auf der Diagonalen mit Ausnahme der aktuellen Zeile zusammengezählt. Im Beispiel in Tabelle 3.3 ist $r_n = 19$ für die Klasse `productName`.

- **Falsch negativ (f_n):** Der Wert, der falsch negativ vorhergesagten, bildet sich dementsprechend aus der Summe einer Zeile ohne die Diagonale. Für die Klasse `productName` ergibt sich der Wert $f_n = 3$.

3.6.3 Beurteilung von Vorschlägen und Kategorisierungen

Die bisher vorgestellten Metriken eignen sich insbesondere für die Klassifikation mit einer einzelnen Zielklasse. Daher können in der Schemazuordnung bis jetzt nur Zuordnungen mit $1:1$ und $N:1$ Kardinalität evaluiert werden. Dies ist für die primäre Zuordnung ausreichend. Sollen allerdings auch $1:M$ Zuordnungen abgebildet werden, müssen zusätzliche Metriken verwendet werden.

Zudem kann die Reihenfolge der Vorhersagen, geordnet nach ihrer Konfidenz, je nach Anwendungsfall relevant sein. Etwa, um beurteilen zu können, wie gut sich ML-Systeme für Empfehlungen eignen, um Angestellte bei manuellen Zuordnungen effizient zu unterstützen.

In den folgenden Abschnitten werden Metriken für diese Anwendungen vorgestellt.

Mean Reciprocal Rank Der Mean Reciprocal Rank (MRR) stammt aus dem Bereich der Informationsgewinnung aus Textdokumenten. Diese Metrik folgt der Idee, das Ergebnis eines Empfehlungssystems danach zu bewerten an welcher Stelle, dem Rang, die erste relevante Empfehlung steht. Ein einfaches Beispiel ist die Bewertung von Suchmaschinen, die relevante Suchergebnisse möglichst weit oben in der Liste aller Suchergebnisse platzieren sollen. Für den MRR wird dann der Mittelwert über alle Vorschläge des Systems gebildet [47].

Diese Metrik lässt sich auf Klassifikationen übertragen, wenn vorhergesagte $1:1$ Zuordnungen als Vorschlag interpretiert werden. Dann kann bewertet werden, wie gut die primäre Zuordnung erkannt wurde. Dazu müssen die ermittelten Zuordnungen anhand ihrer Konfidenz geordnet werden, wobei die höchste Konfidenz als beste Empfehlung gewertet wird.

Verdeutlicht wird die Funktionsweise des MRR für Schemazuordnungen in Tabelle 3.4. Werden mehrere mögliche primäre Zuordnungen für ein beliebiges Eingangsattribut vorgeschlagen, wird die Position bzw. der Rang der richtigen Zuordnung unter den Vorschlägen bewertet. Die richtige Zuordnung wird im Empfehlungssystem von einem Menschen ausgewählt oder entspricht der Zielklasse im Testdaten-

Vorgeschlagene Zuordnungen	richtige Zuordnung	Rang	reziproker Rang
productName	productName	1	1
Application_Advice, *Application*, Description	Application	2	0.5
Storage_Shipping, Reactivity, *Shipping*	Shipping	3	0.33

$MRR = 0.61$

Tabelle 3.4: Beispielhafte Ermittlung des MRR

satz. Der MRR ist dann der Mittelwert aller reziproken Ränge der vorhergesagten Zuordnungen (vgl. Gleichung (3.14)). Im Falle eines idealen Systems ist $MRR = 1$.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rang}_i} \tag{3.14}$$

In der Gleichung entspricht $|Q|$ der Anzahl Zeilen in Tabelle 3.4. Somit ist im Beispiel $|Q| = 3$. Jede Zeile steht für eine einzelne Anfrage an das Empfehlungssystem. Im ML-Kontext entsprechen die Anzahl der Zeilen der Anzahl von Instanzen im auszuwertenden Datensatz, welches üblicherweise der Testdatensatz ist. Im Beispiel werden die konkreten Instanzen aus dem Datensatz zur Vereinfachung vernachlässigt.

Label Ranking Average Precision Eine Verallgemeinerung des MRR ist LRAP. Die LRAP-Metrik misst, welcher Anteil vorhergesagter Kategorien über einer Kategorie aus dem Goldstandard liegt. Dazu werden die Konfidenzen ausgewertet, um den Rang einer Kategorie, das Label, zu bestimmen. Je genauer die Reihenfolge der Label aus dem Goldstandard abgebildet wird, desto höher wird der Wert der Metrik, wobei 1 der beste Wert ist. Enthält die Zuordnung im Goldstandard genau ein Label, gilt $LRAP = MRR$.

Formell definiert wird die LRAP-Metrik in Gleichung (3.15) nach Tsoumakas et al. [48]. Angenommen n ist die Anzahl der Beispiele und m ist die Anzahl der Labels, dann kann in der Gleichung jede Zeile einer binären Indikatormatrix $y \in \{0, 1\}^{n \times m}$ verwendet werden, um die Zuordnungen aus dem Goldstandard als binäre Kategorien darzustellen. Ein Label nimmt den Wert 0 an, wenn die Zuordnung im Goldstan-

dard fehlt und 1, wenn es vorhanden ist. Unter Hinzunahme der Vorhersagen des Modells über die Konfidenzwerte $\hat{f} \in \mathbb{R}^{n \times m}$ der Kategorien lässt sich die LRAP-Metrik definieren als

$$LRAP(y, \hat{f}) = \frac{1}{m} \sum_{i=0}^{n-1} \frac{1}{||y_i||_0} \sum_{j:y_{ij}=1} \frac{|\mathcal{L}_{ij}|}{\text{rang}_{ij}} \tag{3.15}$$

wobei $\mathcal{L}_{ij} = \left\{k : y_{ik} = 1, \hat{f}_{ik} \geq \hat{f}_{ij}\right\}$, $\text{rank}_{ij} = \left\{k : \hat{f}_{ik} \geq \hat{f}_{ij}\right\}$, $|\cdot|$ die Mächtigkeit der Menge bestimmt und $||\cdot||_0$ die ℓ_0-Norm bezeichnet, die die Anzahl von Elementen ungleich null in einem Vektor ermittelt.

Im Gegensatz zu den bereits vorgestellten Metriken bietet LRAP die Möglichkeit, die Vorhersage von weiteren Zuordnungen neben der primären Zuordnung in die Bewertung mit einzubeziehen. Dies ist insbesondere im Schema-Matching und bei der Katalogintegration wichtig, da mithilfe der üblichen Metriken keine Aussagen über Qualität der Vorhersagen bei $1 : M$ Beziehungen möglich ist.

AUC Die Area Under the Receiver Operating Characteristic Curve (AUC) [49] ist eine Metrik zur Bewertung von binären Klassifikationsvorhersagen. Die Receiver Operating Characteristic (ROC)-Kurve ist ein Diagramm, das sich aus zwei Parametern zusammensetzt. Dabei werden die Rate der richtig positiven (TPR, Gleichung (3.16)) Vorhersagen und die Rate der falsch positiven (FPR, Gleichung (3.17)) Vorhersagen einer binären Klassifikation bestimmt.

$$TPR = \frac{TP}{TP + FN} \tag{3.16}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.17}$$

Werden TPR und FPR als Diagramm dargestellt, befindet sich die FPR auf der x-Achse und TPR auf der y-Achse. Die ROC-Kurve stellt eine Wahrscheinlichkeitsverteilung dar.

Die AUC wird definiert über die Fläche unter der ROC-Kurve. Sie ist maximal 1, wenn alle Vorhersagen richtig sind, und 0, wenn alle Vorhersagen falsch sind. Ist die AUC gleich 0.5, ist die ROC eine Gerade mit der Steigung 1. In diesem Fall rät das ML-Modell lediglich. Es kann nicht zwischen den beiden Klassen unterscheiden.

AUC lässt sich auf verschiedenen Wegen für die Mehrklassen-Klassifikation und Kategorisierung erweitern. Eine Variante ist das Nutzen einer *one-vs-one*-Strategie, analog zum Vorgehen bei SVMs (vgl. Abschnitt 3.5.1). Dabei werden die AUC-Werte jeder Klasse paarweise mit jeder anderen Klasse verglichen. Um die Ergebnisse

auf unbalancierte Datensätze aussagekräftiger zu machen, kann die a-priori Wahrscheinlichkeit, die relative Häufigkeit der einzelnen Klassen ($\frac{m_j}{m}$), mit einbezogen werden. Für die Berechnung der AUC für c verschiedene Klassen in Gleichung (3.18) muss folglich die AUC von $c(c-1)$ binären Klassifizierern ermittelt werden. [49]

$$AUC_{\text{ovo}} = \frac{1}{c(c-1)} \sum_{j=1}^{c} \sum_{k \neq j}^{c} \frac{m_j}{m} AUC(j,k) \tag{3.18}$$

3.6.4 Subjektive Zufriedenheit

Aus Sicht einer anwendungsorientierten Problemlösung ist es sinnvoll, die Zufriedenheit der Nutzer zu messen. Die Zufriedenheit ist ein höchst subjektives Maß, dessen Bestimmung schwierig ist. Allerdings ist in einigen Fällen die Verknüpfung der Zufriedenheit mit objektiven Metriken möglich. Denkbare Kriterien für die Zufriedenheit sind

- Genauigkeit der Ergebnisse bzw. Vorschläge (Precision, MRR),
- Vollständigkeit der Ergebnisse (Recall),
- Ressourcenaufwand, wie z.B. Speicher oder Rechenleistung,
- Verständlichkeit der Ergebnisse (z.B. Nachvollziehbarkeit),
- Eingabeaufwand,
- Geschwindigkeit und
- Aufwandsreduktion im Gesamtprozess.

3.7 Zusammenfassung

In diesem Kapitel wurden die Grundlagen des maschinellen Lernens eingeführt, die im weiteren Verlauf der Arbeit genutzt werden, um verschiedene Lösungen für eine ML-basierte Katalogintegration zu entwickeln. So müssen die hier vorgestellten Herausforderungen, wie unzureichende Trainingsdaten oder eine Überanpassung im Trainingsprozess, in der Entwicklung und Evaluation berücksichtigt werden.

Zusätzlich wurden die ML-Algorithmen vorgestellt, die unter anderem in Kapitel 6 verwendet werden, um Zuordnungen anhand von Attributnamen zu lernen, sowie sämtliche relevanten Metriken, wie Precision, Recall und F_1Score, zur Bewertung

der Verfahren. Bei der Evaluation von ALR in Kapitel 8, das sowohl aus Attributnamen und -Instanzen lernt, werden die vorgestellten Metriken zur Bewertung von Vorschlägen und Kategorisierungen genutzt.

Kapitel 4

Textmerkmale für Matching-Systeme

Texte und Worte müssen zunächst in einen Vektorraum transformiert werden, um sie als Feature-Vektor für ML-Verfahren nutzen zu können. Dafür müssen Texte zunächst in einzelne Segmente unterteilt werden und diese Segmente wiederum in Einzelteile, die Token, zerlegt werden. In diesem Kapitel werden zunächst die Verfahren und Begriffe der Textsegmentierung eingeführt. Anschließend werden Ähnlichkeitsmetriken beschrieben, die häufig in regelbasierten Systemen zur Bestimmung der Ähnlichkeit von Attributnamen verwendet werden. Danach werden Techniken vorgestellt, die es ermöglichen, Attributnamen und -instanzen in Vektorräume zu transformieren. Alle in diesem Kapitel vorgestellten Strategien lassen sich auf beliebige Texte anwenden. Sie sind unabhängig von ihrem konkreten Einsatz bei der Schemazuordnung. Sie werden im späteren Verlauf lediglich angewandt, daher wird hier nur ein Eindruck der Verfahren vermittelt.

4.1 Textsegmentierung

Ein früher Schritt in der Textanalyse ist die sog. Textsegmentierung. Dabei werden Sätze in einzelne Token zerlegt und Token ggf. in einzelne Buchstaben [11]. Da Attributinstanzen von Produktkatalogen zwar aus Text bestehen können, aber darin oft nur Aufzählungen oder Stichpunkte anstelle von ganzen Sätzen enthalten sind, wird auf die Erkennung von Sätzen nicht weiter eingegangen.

Token Token sind Worte oder Satzzeichen in einem Text bzw. Satz. Die Erkennung von einzelnen Token wird auch als *Wortsegmentierung* bezeichnet. Häufig werden zusammenhängende Texte anhand von komplexen Algorithmen[1] in einzelne Token

[1] z. B. spaCy tokenizer

aufgeteilt. Die einfachste Möglichkeit ist allerdings das Trennen von Texten anhand von Leerzeichen bzw. Trennzeichen.

Die Menge aller Token in einem Datensatz wird mitunter als *Vokabular* bezeichnet. Bei ML-Systemen wird das Vokabular immer aus dem Trainingsdatensatz gebildet. Dadurch können in der späteren Anwendung Fehler auftreten, wenn Token nicht im Vokabular enthalten sind. Diese Fehler werden auch als *Out Of Vocabulary (OOV) Fehler* bezeichnet. Weitere Informationen zur Textsegmentierung finden sich in [50]. Sind beispielsweise einzelne Token aus dem Trainingsdatensatz nicht im Testdatensatz vorhanden, kann eine Attributinstanz als Aneinanderreihung von Token nicht korrekt in einen Vektorraum transformiert werden. Besteht die Attributinstanz nur aus einem einzelnen Token, entsteht bei der Transformation ein Nullvektor ohne jegliche Information.

N-Gramm Ein N-Gramm ist das Ergebnis der Zerlegung eines Textes in einzelne Token, wobei N aufeinanderfolgende Token als N-Gramm zusammengefasst werden. Neben Token können Texte auch in einzelne Zeichen zerlegt werden. Die wichtigsten N-Gramme sind das Uni-, das Bi- und das Trigramm, also ein, zwei oder drei aufeinanderfolgende Token bzw. Fragmente. Wird beispielsweise der Text

```
Kann Spuren von Nüssen enthalten
```

in Token-Bigramme zerlegt, folgen daraus die Bigramme

```
{Kann Spuren, Spuren von, von Nüssen, Nüssen enthalten}.
```

Ein Unigramm enthält dementsprechend die Menge aller Token und ein Trigramm alle Kombinationen aus drei aufeinanderfolgenden Token. Für die Anwendung in der Katalogintegration sind zusätzlich N-Gramme aus Tokenfragmenten interessant. Diese werden im weiteren Verlauf auch als *Subtoken-N-Gramme* bezeichnet. Dazu wird ein Text zuerst segmentiert und die daraus resultierenden Token in weitere Buchstabenfragmente aus N-Grammen aufgeteilt. Am vorherigen Beispiel würde eine Zerlegung in Subtoken-Trigramme wie folgt aussehen:

```
{Kan, ann, Spu, pur, ren,...}
```

Diese Form der Zelegung ist insbesondere hilfreich, um Nullvektoren als Folge von OOV Fehler zu vermeiden. Dadurch können ML-Systemen bessere Eingangsdaten geliefert werden.

Je nach konkreter Implementierung können die Trigramme in den Beispielen von den Token, die mit Frameworks erzeugt werden abweichen, da in Frameworks häufig Füllzeichen für Start und Ende eines Wortes eingesetzt werden. Scikit-Learn[2] erzeugt aus dem Wort `Preis` zum Beispiel die Trigramme

{␣␣P, ␣Pr, Pre, rei, eis, is␣, s␣␣}.

Füllzeichen dienen einzig dem Zweck, dass alle N-Gramme dieselbe Länge besitzen. Je nach Implementierung können andere Füllzeichen verwendet werden. Dies sind häufig # oder umschließen der Worte mit <>. Je nach Anwendung können Token auch als Synonym zu N-Grammen verwendet werden, da jedes N-Gramm ein eigenes Wort im Vokabular darstellt.

4.2 Normalisierung

Eine Normalisierung bzw. Vereinheitlichung von Zeichenfolgen kann die Ergebnisse weiterführender Verfahrensschritte, wie die Anwendung eines ML-Algorithmus, verbessern [51]. Häufig genutzte Normalisierungen im Bereich der Textanalyse sind:

- **Vereinheitlichung der Groß-/Kleinschreibung:** Oftmals werden Sätze oder Token in Kleinschreibung umgewandelt, um spezielle Schreibweisen und Abkürzungen zu vereinfachen. Beispielsweise wird k`Wh` in k`wh` umgewandelt. An einigen Stellen können so uneinheitliche Schreibweisen und Rechtschreibfehler vermieden werden. Dennoch sollte beachtet werden, dass je nach Sprache und Anwendungsfall nützliche Informationen verloren gehen. Insbesondere bei Attributnamen von Produktkatalogen wird *CamelCase* eingesetzt um zwei oder mehr Wörter in einem zu verbinden (siehe Tabelle 2.2 auf Seite 39), was zusätzliche Schritte zur Normalisierung erfordert, wenn alle Informationen erhalten bleiben sollen.

- **Umschreibung diakritischer Zeichen:** Unter der Umschreibung diakritischer Zeichen wird die Auflösung von an Buchstaben angebrachten Zeichen, wie Punkten oder Strichen[3], die die Aussprache oder Bedeutung beeinflussen, verstanden. Bei der Auflösung wird zum Beispiel Jülich in `Juelich` umgeschrieben.

[2] https://scikit-learn.org/stable/

[3] Liste diakritischer Zeichen: `https://de.wikipedia.org/wiki/Diakritisches_Zeichen`, Abgerufen 01.04.2022

- **Entfernung von Leer- und Satzzeichen:** Die Entfernung von Leer- und Satzzeichen erfolgt üblicherweise implizit bei der Tokenisierung. Dabei werden zusätzliche Leerzeichen, Tabulatorzeichen, Zeilenumbrüche (\n) und Satzzeichen von Worten getrennt. In der weiteren Verarbeitung kann entschieden werden, ob und welche Satzzeichen berücksichtigt werden sollen oder ob sie entfernt werden sollen.

- **Normalisierung von Verbindungszeichen:** Anstelle von CamelCase werden bei der Benennung von Attributen auch Binde- und Unterstriche genutzt. Bei der Normalisierung werden diese Zeichen entfernt oder nur eine Variante genutzt.

- **Einheitlicher Umgang mit Ziffern:** Treten Ziffern in textuellen Daten auf, muss ein einheitlicher Umgang gewählt werden. Bezogen auf Produktdaten treten Ziffern meistens in Identifikationsnummern oder Mengenangaben mit Einheiten auf. Bei der Tokenisierung muss entschieden werden, ob die Ziffern gemeinsam mit der Einheit ein einzelnes Token bilden oder ob zwei Token gebildet werden. Für Ziffern ohne Einheit muss in der Vorverarbeitung entschieden werden, ob die Ziffern eine inhaltliche Bedeutung besitzen und Informationen beisteuern, oder ob sie ohne Bedenken entfernt werden können.

- **Umwandlung in Lemma und/oder Wortstämme:** Bei der Umwandlung in ein Lemma wird ein gebeugtes Wort bzw. Token in seine Grundform umgeschrieben [11]. Dadurch wird beispielsweise aus dem Verb *gekauft* die Grundform kaufen. Im Gegensatz dazu wird der Wortstamm über Regeln ermittelt, ohne den Kontext oder die Wortart zu berücksichtigen. Je nach Algorithmus bzw. Regelwerk können sich die Wortstämme unterscheiden. Allerdings muss nicht erst aufwendig ermittelt werden, welche Wortart zugrunde liegt. Weitere Details finden sich bei Jivani [52].

4.3 Ermitteln von Unterschieden in Zeichenfolgen

Das Finden von zusammengehörigen Wörtern oder Zeichenketten in unterschiedlichen Datensätzen wird auch als *String-Matching* bezeichnet. Im Rahmen des Schema-Matching und der Katalogintegration lassen sich Zuordnungen zwischen Eingangs- und Zielschema über den Vergleich von Zeichenfolgen finden. Die Ähnlichkeitsbestimmung kann sowohl auf den Attributnamen als auch den Attributinstanzen er-

folgen. Des Weiteren kann die Ermittlung von Ähnlichkeiten Teil von komplexeren Algorithmen sein, die Zuordnungen zu bestimmen, beispielsweise *Similarity Flooding* [53].

Ziel ist, über eine Ähnlichkeitsmetrik Zeichenketten zu finden, die sich auf dieselbe Entität aus der echten Welt oder auf dasselbe zugrunde liegende Konzept beziehen. Das *String-Matching* löst das folgende Problem: Aus zwei Datensätzen A und B, die Strings enthalten, sollen alle Paare (a, b) mit $a \in A$ und $b \in B$ gefunden werden, die zusammengehören. Ein höherer Ähnlichkeitswert lässt auf eine größere Zusammengehörigkeit schließen. [54]

Definition 4.3.1: Ähnlichkeitsfunktion

$$\begin{aligned} sim(a, b) \rightarrow & \quad [0..1], \\ sim(a, a) = & \quad 1, \\ sim(a, b) = & \quad sim(b, a) \end{aligned}$$

Definition 4.3.1 zeigt die Eigenschaften einer Ähnlichkeitsfunktion für die zwei Strings a und b. Eine Ähnlichkeitsfunktion bildet die Ähnlichkeit der Zeichenketten auf das Intervall $[0, 1]$ ab. Je höher der Wert ist, desto größer ist die Ähnlichkeit. Bei identischen Zeichenketten gilt somit $sim(a, b) = 1$. Zusätzlich ist eine Ähnlichkeitsfunktion symmetrisch, sodass die Reihenfolge des Vergleichs für die Ähnlichkeit irrelevant ist.

In der praktischen Anwendung wird eine korrekte Zuordnung nicht nur von der konkreten Ähnlichkeitsfunktion beeinflusst, sondern auch von unterschiedlichen Schreibweisen, Synonymen und Abkürzungen. Hinzu kommen typischerweise Fehler in den Daten, wie zum Beispiel Buchstabendreher, Rechtschreibfehler oder unterschiedliche Zeichenkodierung. Wie beschrieben, können verschiedene Normalisierungen angewendet werden, um diese Einflüsse möglichst gering zu halten (vgl. Abschnitt 4.2).

Aspekte der Skalierbarkeit spielen vor allem dann eine Rolle, wenn Ähnlichkeitsfunktionen auf eine größere Menge Zeichenketten angewandt werden, um Übereinstimmungen zu finden, genannt *Data-Matching*. In diesem Fall müsste die Ähnlichkeitsfunktion $sim(a, b)$ auf alle Paare des kartesischen Produkts zweier Datensätze angewandt werden.

Da die Anzahl der Aufrufe von Ähnlichkeitsfunktionen quadratisch mit der Anzahl der zu vergleichenden Instanzen wächst, ist dieser Ansatz ohne entsprechende Op-

timierungen in der Praxis nicht einsetzbar. Dennoch gibt es Lösungen, die z.B. Ähnlichkeitsfunktionen nur auf vielversprechende Kandidaten anwenden [54].

In dieser Arbeit werden Ähnlichkeitsfunktionen hauptsächlich zur Zuordnung einer überschaubaren Anzahl von Attributnamen verwendet. Daher wird an dieser Stelle nicht weiter auf Performance Optimierungen eingegangen. Im Folgenden werden die verwendeten Ähnlichkeitsfunktionen, gruppiert nach vergleichenden Eigenschaften, vorgestellt. Die Gruppierung in dieser Arbeit folgt der Einteilung von Bär et al. [55] in lexikalische, phonetische und semantische Ähnlichkeitsmetriken.

4.3.1 Lexikalische Ähnlichkeit

Lexikalische Ähnlichkeitsfunktionen betrachten Zeichenabfolgen ohne den Kontext, in den diese eingebettet wurden. Sie verwenden häufig Sequenzen von Buchstaben oder die Menge enthaltener Buchstaben, um die Ähnlichkeit zu bestimmen.

Levenshtein-Ähnlichkeit

Eine bekannte Metrik ist die Levenshtein-Ähnlichkeit, die auf der Editier-Distanz basiert. Häufig wird sie daher auch als Levenshtein-Distanz bezeichnet. Im Unterschied zu Ähnlichkeitsfunktionen ($sim(a, a) = 1$) liefern Distanzfunktionen einen niedrigen Wert, bei ähnlichen Zeichenabfolgen. Für den Distanzwert gilt dementsprechend $d(a, b) = 0$ bei exakter Übereinstimmung.

Die Editier-Distanz kann durch folgende Gleichung bestimmt werden (vgl. [54]).

$$d_{LEV}(i, j) = min \begin{cases} d(i-1, j-1) & \text{wenn } x_i = y_i \text{ (kopieren)} \\ d(i-1, j-1) + 1 & \text{wenn } x_i \neq y_i \text{ (ersetzen)} \\ d(i-1, j) + 1 & (x_i \text{ löschen}) \\ d(i, j-1) + 1 & (y_i \text{ einfügen}) \end{cases} \tag{4.1}$$

Dabei sind $a = a_1 a_2 \dots a_n$ und $b = b_1 b_2 \dots b_m$ Zeichenketten mit den Zeichen a_i bzw. b_j und $d(i, j)$ beschreibt die Distanz zwischen Zeichen a_i und b_j. Die Distanz zwischen a und b wird letztendlich über $d_{LEV}(n, m)$ berechnet.

Bei der Transformation von a nach b können vier grundsätzliche Operationen durchgeführt werden: Kopieren, Ersetzen, Löschen und Einfügen.

Kopieren Bei der *Kopieren*-Operation werden alle Zeichen $a_1 a_2 \dots a_{i-1}$ in $b_1 b_2 \dots b_{j-1}$ durchlaufen. Gilt für das jeweils aktuelle Zeichen $a_i = b_j$, wird das Zeichen *kopiert*.

Diese Operation bedeutet in der Praxis keine Veränderung der Zeichenkette, was dazu führt, dass die Levenshtein-Distanz bei identischen Zeichenketten null ist, da nur *Kopieren* durchgeführt wird (vgl. Gleichung 4.1).

Ersetzen Die *Ersetzen*-Operation führt zu einer Erhöhung der Distanz zwischen zwei Zeichenketten. Vor dem *Ersetzen* werden die Zeichen $a_1a_2a \ldots a_{i-1}$ in $b_1b_2 \ldots b_{j-1}$ umgewandelt. Ist dann $a_i \neq b_j$, wird a_i durch b_j ersetzt.

Löschen Beim *Löschen* wird zuerst das aktuelle Zeichen a_i gelöscht und danach die Abfolge $a_1a_2 \ldots a_{i-1}$ in $b_1b_2 \ldots b_j$ umgewandelt.

Einfügen Das *Einfügen* entspricht genau der gegenteiligen Operation des Löschens. Dabei wird zuerst $a_1a_2 \ldots a_i$ in $b_1b_2 \ldots b_{j-1}$ umgewandelt und danach b_j eingefügt.

Diese rekursive Definition der Levenshtein-Distanz führt zu einem großen Rechenaufwand, am Ende das Minimum aller Summen der einzelnen Teildistanzen bestimmt werden kann.

Basierend auf der Levenshtein-Distanz lässt sich eine Ähnlichkeitsfunktion definieren, die alle Anforderungen aus Definition 4.3.1 erfüllt. In Definition 4.3.2 wird dazu die Distanz der Zeichenabfolgen in Relation zur maximalen Länge der Zeichenfolgen gesetzt.

Definition 4.3.2: Levenshtein-Ähnlichkeit

$$sim_{LEV}(a,b) = 1 - \frac{d_{LEV}(a,b)}{max(length(a), length(b))}$$

Ähnlichkeit der längsten gemeinsamen Teilfolge

Eine Alternative zur Levenshtein-Distanz ist die Distanzberechnung über die längste gemeinsame Teilfolge (longest common subsequence (LCS)). Im Gegensatz zur Levenshtein-Distanz sind nur die Operationen *Einfügen* und *Löschen* erlaubt, wobei jede durchgeführte Operation, wie zuvor, die Kosten von 1 trägt. Somit entspricht die LCS-Distanz der Levenshtein-Distanz ohne *Ersetzen*. Es wird die längste Teilfolge von identischen Zeichen (Paaren) in beiden Strings unter Beachtung der Reihenfolge ermittelt. Die Distanz entspricht dann genau der Anzahl der Zeichen, für die kein passendes Paar mehr gefunden werden konnte [56].

Definition 4.3.3: LCS-Ähnlichkeit

$$sim_{LCS}(a,b) = 1 - \frac{d_{LCS}(a,b)}{max(length(a), length(b))}$$

Die LCS-Ähnlichkeit unterscheidet sich von der Levenshtein-Ähnlichkeit nur durch die Distanzfunktion im Zähler. Neben den beiden bisher vorgestellten Ähnlichkeitsmetriken existieren zahlreiche Erweiterungen, wie z.B. *Needleman-Wunch* [57] und *Affine Gap* [58], die insbesondere das *Einfügen* von Zeichen oder sogar ganzen Worten, z. B. Zweitnamen, zwischen umschließenden Worten mit geringeren Straftermen versehen. In einigen Fällen führt dieses Vorgehen zu einer höheren Ähnlichkeit als Levenshtein oder LCS. Diese Ähnlichkeitsfunktionen werden nicht näher betrachtet, da sie für den Anwendungsfall – das Finden zusammengehöriger Attributnamen – keine Vorteile bringen, da Attributnamen in Produktkatalogen häufig aus ein bis drei Worten zusammengesetzt werden.

Die *Smith-Waterman-Ähnlichkeit* [59] ist ebenso eine Erweiterung. Sie berücksichtigt zusätzlich lokale Ähnlichkeiten. Im Idealfall werden unterschiedliche Präfixe und Suffixe ignoriert. Ebenso werden Unterbrechungen der längsten gemeinsamen Sequenz toleriert.

Jaro-Winkler-Ähnlichkeit

Die *Jaro-Ähnlichkeit* wurde entwickelt, um insbesondere kurze Zeichenketten, wie beispielsweise Namen, miteinander zu vergleichen. Diese Ähnlichkeit ist für Matching-Systeme interessant, da Attributnamen häufig aus kurzen Zeichenketten bestehen.

$$sim_{jaro}(a,b) = \frac{1}{3} \cdot \left(\frac{c}{|a|} + \frac{c}{|b|} + \frac{c - \frac{t}{2}}{c}\right) \tag{4.2}$$

Berechnet wird die Ähnlichkeit über Gleichung (4.2) [54]. Dabei bezeichnet

- $|\cdot|$ die Länge des Strings,
- c die Anzahl der Zeichen, die im Abstand von höchstens $\frac{\min\{|a|,|b|\}}{2}$ in beiden Zeichenketten vorkommen, sodass nahe beieinanderliegende Zeichen stärker berücksichtigt werden,
- und t die Anzahl der positionsweise unterschiedlichen Zeichen ($a_i \neq b_i$) im Abstand von höchstens $\frac{\min\{|a|,|b|\}}{2}$.

Durch diese Definition werden beispielsweise Buchstabendreher weniger stark gewichtet als bei der Editier-Distanz. Die *Jaro-Winkler*-Ähnlichkeit berücksichtigt Sonderfälle. So wird angenommen, dass Strings, die über dasselbe Präfix verfügen, ähnlich sind, auch wenn sim_{jaro} niedrig ist. Dazu werden zwei Parameter eingeführt: P_l beschreibt die größte gemeinsame Präfix-Länge und P_w die gewählte Gewichtung des Präfixes. Daraus ergibt sich Definition 4.3.4 [60].

Definition 4.3.4: Jaro-Winkler

$$sim_{jw}(a,b) = (1 - P_l \cdot P_w) \cdot sim_{jaro}(a,b) + P_l \cdot P_w$$

Zeichenmengenbasierte Metriken

Die bisher vorgestellten Metriken hängen stark von der Reihenfolge von Worten oder Zeichen innerhalb einer Zeichenkette ab. Neben der Betrachtung von Zeichenketten als festgelegte Abfolge von Zeichen, ist auch eine mengenbasierte Ähnlichkeitsberechnung möglich. Hierbei wird eine Abfolge von Zeichen als Menge bzw. Vereinigung von Teilmengen angesehen, um die Ähnlichkeit zu ermitteln.

Hierzu wird eine Abfolge von Zeichen in Token oder N-Gramme umgewandelt. Dadurch spielt neben der Metrik selbst ebenfalls die Art und Weise der Tokenisierung eine entscheidende Rolle.

Jaccard-Ähnlichkeit Die einfachste mengenbasierte Metrik ist die Token-Überlappung. Dabei wird aus einer Menge an Token T_a und T_b für die beiden Zeichenketten a und b die Anzahl der überlappenden, also in beiden Zeichenfolgen vorhandenen, Token gezählt, sodass gilt:

$$O(a,b) = |T_a \cap T_b| \tag{4.3}$$

Aus Gleichung (4.3) lässt sich direkt die zugehörige Ähnlichkeitsfunktion, die Jaccard-Ähnlichkeit (vgl. Definition 4.3.5) ableiten. Sie berechnet den Anteil der überlappenden Token an der Vereinigung aller Token aus beiden Zeichenketten [61].

Definition 4.3.5: Jaccard-Ähnlichkeit

$$sim_{JAC} = \frac{|T_a \cap T_b|}{|T_a \cup T_b|}$$

Kosinus-Ähnlichkeit Die Kosinus-Ähnlichkeit baut wie die Jaccard-Ähnlichkeit auf den beiden Tokenmengen T_a und T_b auf. Wie in Definition 4.3.6 zu sehen ist, sind die Zähler identisch. Somit ergibt sich der Unterschied der Ähnlichkeiten aus dem Nenner. Definition 4.3.6 wird auch Ochiai-Koeffizient [62] genannt.

Definition 4.3.6: Kosinus-Ähnlichkeit für Zeichenmengen

$$sim_{cos}(T_a, T_b) = \frac{|T_a \cap T_b|}{\sqrt{|T_a| \cdot |T_b|}}$$

Einschränkungen mengenbasierter Metriken Die hier vorgestellten mengenbasierten Metriken haben insbesondere bei längeren Texten den Nachteil, dass das bloße Vorhandensein eines Wortes beziehungsweise Token ausreicht, um die Ähnlichkeit zu erhöhen. Folgendes Beispiel veranschaulicht die Auswirkungen unterschiedlicher Tokenisierungs-Strategien: Sei `a="abcdef"` und `b="fedcba"`,

- dann ist bei Nutzung von Unigrammen $sim_{cos}(a, b) = 1$ und $sim_{jac}(a, b) = 1$,
- und bei Bigrammen $sim_{cos}(a, b) = 1$ und $sim_{jac}(a, b) = 0$.

4.3.2 Phonetische Ähnlichkeit

Während Ansätze zur Beschreibung der lexikalischen Ähnlichkeit sich nur auf das Erscheinungsbild beziehungsweise die visuellen Eigenschaften der Zeichenfolgen beschränken, fokussieren sich phonetische Ähnlichkeiten auf die gehörte Aussprache des Wortes.

Diese Herangehensweise ist insbesondere beim Zuordnen von Namen sehr erfolgreich und wurde explizit dafür entwickelt [63]. Die Namen Schmidt, Schmitt, Schmid, Schmit, Schmitz, als zweithäufigster deutscher Familienname, klingen ausgesprochen nahezu gleich. Sie sind also phonetisch ähnlich, auch wenn die Schreibweisen sich je nach Region unterscheiden.[4]

Der Soundex-Algorithmus (Definition 4.3.7) ordnet einer Zeichenkette einen vierstelligen Buchstabencode zu. Stimmt dieser Code mit dem Code einer anderen Zeichenkette überein, sind die beiden Zeichenketten phonetisch identisch. Angewandt auf die Schreibweisen von Schmidt ergibt sich S253.

[4] Namensvariationen aus https://de.wikipedia.org/wiki/Schmidt

Kodierung:	0	1	2	3	4	5	6
Buchstabe:	a e i o u y h w	b p f v	c g j k q s x z	d t	l	m n	r

Tabelle 4.1: Phonetische Kodierung des Soundex-Algorithmus für Englisch.

Definition 4.3.7: Der Soundex-Algorithmus [63]

1. Ersetze alle, bis auf den Anfangsbuchstaben, durch die phonetischen Kodierung aus Tabelle 4.1.
2. Entferne aneinander angrenzende Duplikate.
3. Entferne alle Nullen (Vokale).
4. Gib die ersten vier Zeichen der resultierenden Zeichenkette zurück (Anfangsbuchstabe und drei Zahlen)

Durch den einfachen Algorithmus kommt es allerdings häufig zu Fehlern, bei denen unähnlichen Strings gleiche Kodierungen zugeordnet werden. Weiterhin funktioniert der Soundex-Algorithmus hauptsächlich für die Englische Sprache. Für andere Sprachen, wie Deutsch, muss die Kodierungstabelle geändert werden, da andere Betonungsregeln gelten und Umlaute fehlen. Zusätzlich können mehr als vier Stellen für die Kodierung genutzt werden (siehe auch Kölner Phonetik [64]).

Alle Kodierungen haben gemeinsam, dass nur binär entschieden werden kann, ob die Kodierung übereinstimmt und Zeichenketten somit phonetisch ähnlich sind oder nicht. Eine genauere Ermittlung der phonetischen Ähnlichkeit, wie bei bisher vorgestellten Metriken, bietet der Soundex-Algorithmus nicht. In der Praxis findet der Soundex-Algorithmus dennoch relevante bzw. ähnliche Zeichenketten, die von lexikalischen Ansätzen übersehen werden.

Die Editex-Distanz [63] kombiniert den Soundex-Alogrithmus mit der Levenshtein-Distanz, um die bisherigen Einschränkungen zu umgehen. Für die Editex-Distanz wird die Kodierung des Soundex-Algorithmus so verändert, dass Buchstaben in mehreren phonetischen Gruppen enthalten sein dürfen, da sich die Aussprache je nach umgebenden Buchstaben ändern kann (vgl. Tabelle 4.2).

Zusätzlich wird eine Editier-Distanz nach dem Vorbild der lexikalischen Editier-Distanz definiert. Dazu werden die Funktionen $r(a, b)$ aus Gleichung (4.4) und $d(a, b)$

Gruppierung:	0	1	2	3	4	5	6	7	8	9
Buchstabe:	a e i o u y	b p	c k q	d t	l r	m n	g j	f p v	s x z	c s z

Tabelle 4.2: Phonetische Gruppen der Editex-Distanz.

aus Gleichung (4.5) eingeführt, wobei $a \in s$ und $b \in t$ Buchstaben aus den Zeichenketten s und t sind. r und d sind jeweils null, wenn die Buchstaben a und b identisch sind.

$$r(a,b) = \begin{cases} 0 & \text{wenn } a = b \\ 1 & \text{wenn } a \text{ und } b \text{ aus der gleichen Gruppe stammen} \\ 2 & \text{sonst} \end{cases} \tag{4.4}$$

Wenn die Buchstaben a und b in die gleiche Gruppe fallen (siehe Tabelle 4.2), nimmt r den Wert 1 an und andernfalls 2. Die Funktion d unterscheidet sich von r insoweit, als das sie 1 zurückgibt, wenn a der Buchstabe h oder `w` ist und gleichzeitig a ungleich b. Auf diese Weise sollen stumme Laute berücksichtigt werden, die nicht in Tabelle 4.2 enthalten sind.

$$d(a,b) = \begin{cases} 0 & \text{wenn } a = b \\ 1 & \text{wenn } (a = \texttt{'h'} \text{ oder } a = \texttt{'w'}) \text{ und } a \neq b \\ 2 & \text{sonst} \end{cases} \tag{4.5}$$

Über diese beiden Funktionen lässt sich die Editex-Distanz rekursiv definieren (siehe Definition 4.3.8). Je ähnlicher zwei Worte phonetisch sind, desto geringer ist die Distanz. Somit ist eine besser Abstufung der Ähnlichkeit möglich als im Soundex-Algorithmus, der nur zwischen ähnlich und nicht ähnlich unterscheiden kann.

Definition 4.3.8: Editex-Distanz nach Zobel und Dart [63]

$$
\begin{aligned}
edit(0,0) &= 0 \\
edit(i,0) &= edit(i-1,0) + d(a_{i-1}, a_i) \\
edit(0,j) &= edit(0,j-1) + d(b_{j-1}, b_j) \\
edit(i,j) &= \min \begin{cases} edit(i-1,0) + d(a_{i-1}, a_i), \\ edit(0,j-1) + d(b_{j-1}, b_j), \\ edit(0,j-1) + d(b_{j-1}, b_j \end{cases}
\end{aligned}
$$

4.3.3 Semantische Ähnlichkeit

Die semantische Ähnlichkeit zwischen Worten lässt sich entweder über geeignete Wortvektoren, die in Abschnitt Abschnitt 4.4 beschrieben werden, oder über die Entfernung der beiden Worte in lexikalisch-semantischen Nachschlagewerken, wie zum Beispiel WordNet[5] [65], bestimmen.

WordNet ist eine große lexikalische Datenbank für die englische Sprache. Sie enthält Substantive, Verben, Adjektive und Adverbien, die in Gruppen von kognitiven Synonymen (Synsets) zusammengefasst werden. Jede einzelne Gruppe drückt dabei ein bestimmtes Konzept aus. Die Synonyme sind durch begrifflich-semantische und lexikalische Beziehungen miteinander verknüpft. Aus den Verknüpfungen entsteht ein Netz von sinnvoll miteinander verbundenen Wörtern und Konzepten. Die Struktur und die freie Verfügbarkeit von WordNet haben es zu einem nützlichen und häufig genutzten Werkzeug für die Computerlinguistik und die Verarbeitung natürlicher Sprache werden lassen.

Auf den ersten Blick ähnelt WordNet einem einfachen Thesaurus, da es Wörter auf der Grundlage ihrer Bedeutungen zusammenfasst. WordNet verknüpft jedoch nicht nur Wortformen, sondern auch bestimmte Bedeutungen von Wörtern miteinander. Folglich sind Wörter, die im Netz in unmittelbarer Nähe zueinander stehen, semantisch eindeutig zuzuordnen. Die wichtigste Beziehung zwischen den Wörtern im WordNet sind Synonyme. Sie werden in ungeordneten Mengen, sog. Synsets, gruppiert.

Jedes der 117 000 Synsets von WordNet ist mit anderen Synsets durch eine kleine Anzahl von Beziehungen verknüpft. Zusätzlich enthält ein Synset eine kurze Definition und in den meisten Fällen einen oder mehrere kurze Sätze, die die Verwendung

[5] https://wordnet.princeton.edu/

der Synsetmitglieder veranschaulichen. Wortformen mit mehreren unterschiedlichen Bedeutungen werden in ebenso vielen unterschiedlichen Synsets dargestellt. Somit ist jedes Form-Bedeutungs-Paar im WordNet einzigartig. Die am häufigsten kodierte Beziehung zwischen Synsets ist die übergeordnete Beziehung, auch Hyperonymie oder Hyponymie genannt.

Sie verbindet allgemeinere Synsets wie Möbel, Möbelstück mit immer spezifischeren, wie Bett und Etagenbett. Die Hyponymie-Beziehung ist transitiv: Wenn ein Sessel eine Art von Stuhl ist, und wenn ein Stuhl eine Art von Möbel ist, dann ist ein Sessel eine Art von Möbel. Zusätzlich kann WordNet zwischen Typen (allgemeine Substantive) und Instanzen (bestimmte Personen, Länder und geografische Einheiten) unterscheiden. Instanzen sind immer Blattknoten (Endknoten) in ihren Hierarchien. Ein ähnlicher Aufbau gilt auch für Verben und Adjektive. [65]

Der Aufbau von WordNet lässt sich folglich als Graph mit Knoten und Kanten oder als Ontologie interpretieren. Auf dieser Basis kann die Pfadähnlichkeit zwischen Worten definiert werden. Die genutzten Ähnlichkeitsmaße basieren üblicherweise auf der Berechnung der Länge des kürzesten Pfades zwischen Konzepten in einer Ontologie, in diesem Falle WordNet. Ein solches Verfahren wurde beispielsweise von Wu und Palmer [66] beschrieben.

Hierbei wird die Ähnlichkeit zwischen zwei Konzepten über die folgende Gleichung bestimmt:

$$Con_{sim}(C_1, C_2) = \frac{2 \cdot N_3}{N_1 + N_2 + 2 \cdot N_3} \tag{4.6}$$

Dabei N_1 ist die Anzahl der Knoten auf dem Weg von dem ersten Konzept C_1 bis zur ersten gemeinsamen übergeordneten Beziehung (C_3), die mit dem zweiten Konzept C_2 geteilt wird. N_2 ist dementsprechend die Anzahl der Knoten auf dem Weg von C_2 nach C_3. N_3 ist die Anzahl der Knoten auf dem Pfad von C_3 bis zur Wurzel. Eine Übersicht weiterer Ansätze auf Basis von WordNet bietet [67].

4.4 Wortvektoren

Um Worte, Sätze oder Phrasen über Algorithmen auszuwerten, können sie in einen Vektorraum überführt werden. Diese sog. Wortvektoren können entweder direkt als Feature-Vektor für ein ML-System genutzt werden oder über die Kosinus-Ähnlichkeit von Vektoren (vgl. Abschnitt 4.4.5) als Grundlage von Ähnlichkeitsberechnungen dienen. In den folgenden Abschnitten werden mehrere Transformationsverfahren, die Text in Feature-Vektoren umwandeln, vorgestellt.

4.4.1 One-Hot-Kodierung

Die OH-Kodierung ist die einfachste Form Token in einen Vektorraum zu transformieren. Bei gegebenem Vokabular V wird einem Token ein binärer Vektor zugeordnet: Er ist 1, an der Position (Index), wo das Token im Vokabular auftaucht und an allen anderen Positionen 0. Damit entspricht die Länge eines solchen Wortvektors der Länge des Vokabulars.

Um einen Satz oder einen Attributnamen in einen OH-Vektor zu transformieren, muss zunächst ein Vokabular, z.B. aus dem Trainingsdatensatz, gebildet werden. Im Anschluss muss der Satz in Token aufgeteilt werden. Immer wenn ein Token aus dem Vokabular im Satz vorkommt, wird der Wert des Vektors am zugehörigen Index auf 1 gesetzt.

Diese OH-kodierten Vektoren können direkt als Eingangsvektoren für einen ML-Algorithmus dienen. Anstatt einer gleichen Gewichtung der einzelnen Token über eine Ähnlichkeitsmetrik, können Token bei Klassifikationsaufgaben durch den ML-Algorithmus ein höheres Gewicht erhalten.

4.4.2 TF-IDF

Die One-Hot-Kodierung berücksichtigt, wie die mengenbasierten Metriken, nicht, welchen Informationsgehalt ein Token oder N-Gramm besitzt. Dem lässt sich entgegenwirken, indem Token nach Bedeutung oder Aussagekraft gewichtet werden. Ein solches Verfahren ist term frequency-inverse document frequency (TF-IDF). Dieses stammt ursprünglich aus dem Bereich der Schlagwortsuche und der Informationsgewinnung. Die intuitive Bedeutung der Metrik ist, dass zwei Strings ähnlich zueinander sind, wenn sie gemeinsame charakteristische Terme oder Token enthalten.

TF-IDF beurteilt die Relevanz eines Token im Kontext eines Dokuments. Die Relevanz wird ermittelt, indem die Häufigkeit des Vorkommens in einem Dokument ins Verhältnis zur Häufigkeit im gesamten Korpus gesetzt wird (vgl. Definition 4.4.1 nach Schütze et al. [68]). Darüber kann jedem Token in einem Dokument ein Wert zugewiesen werden. In der Praxis werden die resultierenden Werte häufig noch durch die ℓ_2-Norm normalisiert, sodass jedes Token im Vokabular einen Wert in $[0, 1]$ besitzt, wobei 1 den höchsten Relevanzwert darstellt.

Definition 4.4.1: term frequency-inverse document frequency (TF-IDF)

Gegeben seien ein Korpus C mit Dokumenten d ($d \in C$) und ein Term t als String, dann wird das Maß wie folgt definiert:

$$
\begin{aligned}
tf(t,d) &= t\#d && \text{(Häufigkeit von } t \text{ in } d \text{ bzw. term frequency)} \\
idf(t) &= log\left(\frac{|C|}{|\{d \in C; t \in d\}|}\right) && \text{(inverse document frequency)} \\
tfidf(d,t) &= tf(t,d) \cdot idf(t)
\end{aligned}
$$

Die ermittelten Werte pro Token können analog zur OH-Kodierung direkt als Eingangswerte für einen ML-Algorithmus genutzt werden, um die weitere Gewichtung im Training zu bestimmen. Dabei ist die Länge des Eingangsvektors wie zuvor gleich der Größe des Vokabulars. Neben Token kann TF-IDF auf alle Formen von N-Grammen angewendet werden.

4.4.3 Word2Vec

Die bisher vorgestellten Ansätze ermöglichen zwar die Transformation von Worten in Vektorräume, allerdings mit entscheidenden Nachteilen: Die Dimension des Vektors ist an die des Vokabulars gebunden, wobei nur wenige Elemente eines Wort- bzw. Satzvektors Informationen (Elemente ungleich 0) enthalten. Des Weiteren wird die Bedeutung der Token nicht durch die Vektoren erfasst.

Um Worte so in einen Vektorraum zu transformieren, dass die Bedeutung in den Vektoren abgebildet wird, kann ein alternatives Problem formuliert werden. Abbildung 4.1 zeigt den Continuous Bag Of Words (CBOW) Ansatz. Die Idee hinter diesem Ansatz ist, dass ein eigenständiges NN trainiert wird, anhand der Kontextwörter (gelb) das Wort im Zentrum (z grün, `Fett`) vorherzusagen. In der Abbildung wird eine Kontextgröße K von zwei Worten genutzt [69].

Initial wird jedem Wort im Vokabular ein Vektor mit festgelegter Dimension, in der Praxis häufig 100 oder 300, zugeordnet. Dieser Vektor wird initial mit zufälligen Werten gefüllt, wie beispielsweise normalverteilte Zufallszahlen. Die Vektoren der Kontextwörter ($w(t-2),\ w(t-1),\ w(t+1),\ w(t+2)$) werden dann aufsummiert, wobei in der Praxis in der Regel der Durchschnitt gebildet wird. Anschließend wird

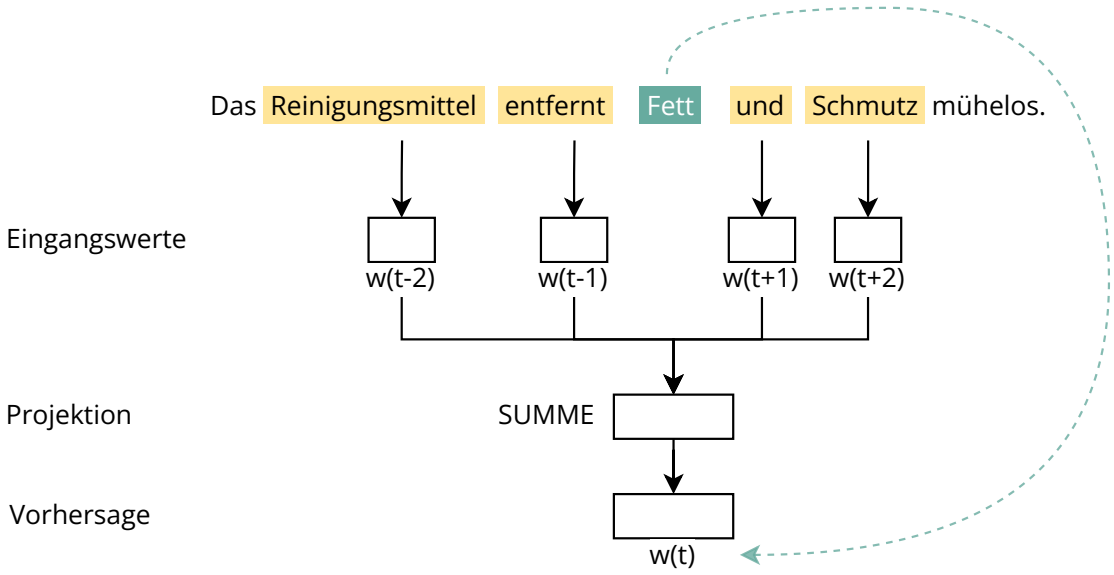


Abbildung 4.1: CBOW-Architektur nach Mikolov et al. [69]

über die Softmax-Funktion in Gleichung (4.7) die Wahrscheinlichkeit für das Wort im Zentrum z ermittelt und im Trainingsprozess letztendlich optimiert.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{für } i = 1, 2, \ldots, K \tag{4.7}$$

Währenddessen werden die Vektoren der Kontextwörter in jedem Schritt angepasst. Nach erfolgtem Training steht dadurch eine Nachschlagetabelle zur Verfügung, aus der ein Vektor für ein einzelnes Wort abgelesen werden kann.

Dieses Optimierungsproblem kann auch umgedreht werden. Der sog. Skip-Gram Ansatz, versucht aus einem Wort im Zentrum die Kontextworte vorherzusagen [69]. Beide Varianten ordnen Worten Vektoren zu, die semantische Ähnlichkeiten abbilden können. Hier zeigt sich ein Unterschied zu WordNet. Während WordNet manuell erstellt wurde, ermöglichen Verfahren wie Word2Vec ebenfalls die Bedeutung von Worten zu erfassen. Hier müssen lediglich ausreichend große Mengen Text prozessiert werden.

In der Praxis existieren viele Nachschlagetabellen mit Wortvektoren, die auf großen, öffentlichen Datensätzen trainiert wurden. Es müssen jedoch einige Einschränkungen berücksichtigt werden: Ist ein Wort nicht im Vokabular enthalten, tritt ein OOV-Fehler auf, was die weitere Nutzung der Vektoren als Eingangswert in andere ML-Modelle erschwert. Daher wird häufig das Token `<UNK>` für unbekannte Worte

eingefügt. Dies ist insbesondere bei der Katalogintegration ein Hindernis (vgl. Abschnitt 2.5), da Worte sehr häufig nicht in bereits vorhandenen Nachschlagetabellen enthalten sind und die vorhandene Datenmenge häufig ungeeignet ist, eigene Wortvektoren zu trainieren.

4.4.4 FastText

Wie beschrieben führen sowohl durch Skip-Gram als auch durch CBOW erzeugte Vektoren zu häufigen OOV-Fehlern. Um dem entgegenzuwirken, können Strukturen innerhalb von Wörtern genutzt werden. Dies gilt insbesondere, um eine Vektordarstellung von seltenen oder falsch geschriebenen Wörtern ermitteln zu können.

Eine mögliche Lösung ist die Verwendung eines *bag of n-grams* anstelle eines *bag of words* [70]. Dabei wird jedes einzelne Wort über die Menge seiner Subtoken-N-Gramme und sich selbst repräsentiert. Zusätzlich wird jedes Wort von den Zeichen < und > eingerahmt, um Präfixe und Suffixe von anderen Teilfolgen unterscheiden zu können. Veranschaulicht am Beispiel von *Eistee*, aufgeteilt in Trigramme, folgt aus dieser Vorgehensweise die Aufteilung in `<Eistee>`, `<Ei`, `Eis`, `ist`, `ste`, `tee`, `ee>`. Hierbei ist zu bemerken, dass das Trigramm `ist` eine andere Darstellung erhält als das Wort *ist*.

Angenommen, es existiert ein Vokabular mit der Größe G bestehend aus Subtoken-N-Gramme. Dann kann ein Wort w über seine Subtoken-N-Gramme $\mathcal{G}_w$ als echte Teilmenge von G dargestellt werden. In der Bewertungsfunktion (vgl. Gleichung (4.8)) wird der Vektor z_g genutzt, um ein einzelnes Subtoken-N-Gramm g abzubilden. v_c entspricht dem Vektor für das Kontextwort c. [70]

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c \tag{4.8}$$

Im Training wird die Bewertungsfunktion $s(w, c)$ optimiert.

In der FastText[6] genannten Implementierung von Bojanowski et al. [70] werden bei der Berechnung der Wortvektoren alle N-Gramme der Länge drei bis einschließlich sechs und zusätzlich das Wort selbst genutzt, wobei zur Berechnung weitere Tricks, wie zum Beispiel der *Hashing Trick* verwendet werden, um Einschränkungen von Speichergröße und Rechenzeit zu verringern [71, 72].

[6] https://fasttext.cc/

4.4.5 Vektorbasierte Kosinus-Ähnlichkeit

Die Kosinus-Ähnlichkeit ist definiert über das ℓ_2-normalisierte Skalarprodukt zweier Vektoren. Sind x und y Zeilenvektoren entspricht die Kosinus-Ähnlichkeit Gleichung (4.9) [68].

$$sim_{cos}(x, y) = \frac{x y^\top}{\|x\| \|y\|} \tag{4.9}$$

Die Kosinus-Ähnlichkeit beschreibt den Winkel zwischen zwei Vektoren auf einer Einheitskugel. Sind zwei Vektoren identisch, beträgt die Ähnlichkeit 1. Besonders ungleiche Vektoren sind entgegengesetzt ($sim_{cos}(x, y) = -1$). Um den gleichen Wertebereich der vorherigen Definitionen zu verwenden, muss die Kosinus-Ähnlichkeit in den Wertebereich $[0, 1]$ transformiert werden.

Da die Kosinus-Ähnlichkeit für jegliche Vektoren anwendbar ist, kann sie auf jede Art von Wortvektor angewendet werden, um die Ähnlichkeit von Worten zu bestimmen. Je nachdem welches Verfahren zur Erzeugung der Wortvektoren genutzt wird, kann die Kosinus-Ähnlichkeit lexikalische Ähnlichkeiten, zum Beispiel über die OH-Kodierung oder sogar semantische Ähnlichkeiten, beispielsweise durch FastText, beschreiben.

4.5 Zusammenfassung

In diesem Kapitel wurden Verfahren zum Umgang mit Texten und Zeichenketten in ML-Systemen beschrieben. Alle Verfahren basieren darauf, dass die Texte zunächst segmentiert, also in Token aufgeteilt werden, bevor der Text weiter normalisiert wird.

Für kurze Zeichenketten, insbesondere für Attributnamen, können Ähnlichkeitsmetriken genutzt werden, um bei der Katalogintegration geeignete Zuordnungen zu ermitteln. Dabei können die Ähnlichkeitsmetriken verschiedene Eigenschaften, wie Phonetik oder Semantik der einzelnen Worte, vergleichen. Je ähnlicher einzelne Worte oder Token einer Zeichenkette sind, desto wahrscheinlicher ist, dass eine Zuordnung vorliegt. Jede Metrik besitzt eigene Stärken und Schwächen, sodass die Verwendung mehrerer Metriken als Feature-Vektor in einem ML-System die Schwächen ausgleichen könnte.

Alternativ können Wortvektoren als Eingangsvektor für ML-Systeme genutzt werden. Dies reicht von einfachen Verfahren, wie der OH-Kodierung bis hin zu komplexeren FastText-Vektoren, die auch die Semantik zwischen Subtoken berücksichtigt.

Kapitel 5

Bestimmung der Schemazuordnungen über Attributnamen

In diesem Kapitel werden Verfahren entwickelt, die Schemazuordnungen anhand von Attributnamen bestimmen können, um FF1 zu beantworten. Durch die Einschränkungen auf die Attributnamen werden nur $1:1$ Zuordnungen berücksichtigt. Dazu werden verschiedene Ansätze zur Extraktion von Merkmalsvektoren in Kombination mit unterschiedlichen ML-Verfahren erarbeitet und im folgenden Kapitel evaluiert. Da das Zielschema bei der hier verwendeten Form der Katalogintegration fest definiert ist, wird ein Klassifikationsansatz verfolgt. Zusätzlich werden die Datengrundlage und nicht ML-basierte Referenzverfahren als Vergleichsmöglichkeit vorgestellt. In diesem Kapitel dargestellte Verfahren wurden in großen Teilen bereits in Schmidts et al. [S3] veröffentlicht.

5.1 Aufbau als Klassifikationsansatz

Sind die Zielattributnamen im Vorhinein bekannt, sind die folgenden Ansätze möglich, um die Zuordnungen durch ein ML-System ermitteln zu lassen.

- **Klassifikationsansatz:** Beim Klassifikationsansatz über die Verfahren aus Kapitel 4 Feature-Vektoren aus den Eingangsattributnamen gebildet und anschließend als Eingangswerte für ein ML-System genutzt. Dieses nutzt einen Klassfikationsalgorithmus, um ein Zielattribut zu bestimmen, wie in Abbildung 5.1 dargestellt. Bei der Klassifikation basierend auf Attributnamen wird eine exklusive Klassifikation durchgeführt, sodass jeweils ein Eingangsattribut genau einem Zielattribut gegenüber steht, wobei verschiedene Eingangsattribute das

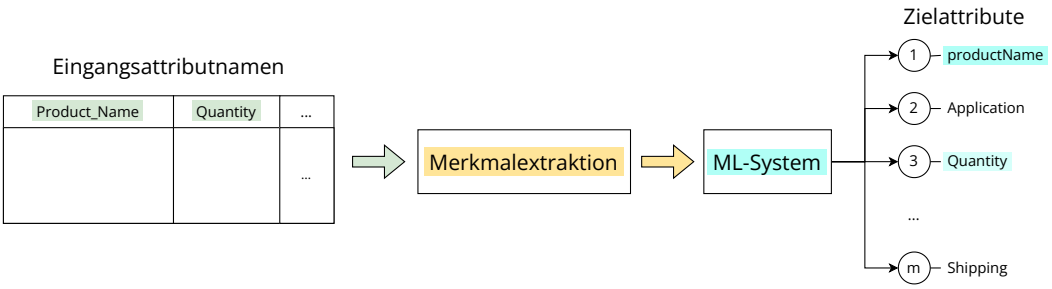


Abbildung 5.1: Grundsätzlicher Ablauf der Vorhersage von Zuordnungen bei einem Klassifikationsansatz anhand von Attributnamen

gleiche Zielattribut besitzen können ($N:1$). Für die Festlegung der Zuordnungen wird in der Regel das Zielattribut mit der höchsten Konfidenz genutzt.

- **Regressionsansatz:** Bei einem Regressionsansatz muss ein ML-System anstelle einer Klasse mit Konfidenz einen Wert vorhersagen und somit einen direkten $1:1$ Vergleich durchführen, während die extrahierten Merkmale die gleichen wie bei einer Klassifikation sein können. Beispielsweise müsste der Regressionsansatz vorhersagen, wie gut `Product_Name` und `Application` (vgl. Eingangsattribute und Zielattribute aus Abbildung 5.1) zusammengehören. Anschließend muss dieser Vergleich bzw. die Vorhersage mit jedem weiteren Zielattribut durchgeführt werden, sodass auch hier die Abwägung zwischen Schwellwert und Höchstwert getroffen werden muss, um letztendlich die Zuordnung abzuleiten. Um alle Zuordnungen zu ermitteln, müssten die Berechnungen mit jedem Eingangsattribut durchgeführt werden.

Ein großer Nachteil des Regressionsansatzes ist, dass implizit eine Ähnlichkeitsfunktion basierend auf den extrahierten Merkmalen gelernt wird, jedoch nicht die Gewichtung zwischen den Attributen des Zielschemas. Die Gewichtung muss dementsprechend, wie bei üblichen Ähnlichkeitsfunktionen (vgl. Kapitel 4), durch einen weiteren Algorithmus bestimmt werden. Im Gegensatz dazu berücksichtigt der Klassifikationsansatz bereits das Zielschema und lernt die Gewichtung mit. Sie wird in der Regel durch die Konfidenz wiedergegeben.

Aus diesem Grund werden im weiteren Verlauf verschiedene Klassifikationsansätze entwickelt. Der Aufbau ist bei allen Kombinationen identisch: Anhand eines Eingangsattributs L_i^{in} aus einem Produktkatalog P^{in} wird versucht, das zugehörige Zielattribut aus L^{out} vorherzusagen. Weitere Informationen, wie Spalteninhalte oder Metadaten, stehen an dieser Stelle nicht zur Verfügung. Daher werden im nachfol-

genden Abschnitt Möglichkeiten beschrieben, geeignete Textmerkmale aus den Attributnamen zu extrahieren.

5.2 Geeignete Vektortransformationen für Klassifikationsansätze

Für ML-Systeme können verschiedene Merkmale als Eingangsvektoren benutzt werden. In der Literatur werden für ML-basierte Verfahren häufig N-Gramme und Word-2Vec genutzt, während für regelbasierte Ansätze auf Ähnlichkeitsmetriken zurückgegriffen wird (vgl. Kapitel 9). In diesem Abschnitt wird ausgeführt, wie auch aus Ähnlichkeitsmetriken Feature-Vektoren gebildet werden können. Darüber hinaus werden die weiteren Ansätze zur Vektorisierung der Attributnamen aus Kapitel 4 aufgegriffen und im Kontext der Katalogintegration diskutiert.

5.2.1 Ähnlichkeitsmetriken als Feature-Vektor

Ähnlichkeitsmetriken bestimmen, wie in Abschnitt 4.3 beschrieben, die Ähnlichkeit von zwei Zeichenketten anhand verschiedener Eigenschaften, wie Struktur, Phonetik oder Semantik. Dabei besitzt jede Metrik besitzt jedoch eigene Stärken und Schwächen, sodass nicht klar ist, welche Metrik für Vergleiche von Attributnamen im Rahmen der Katalogintegration am besten funktioniert.

Wie in Definition 4.3.1 beschrieben, liegt der Wertebereich von Ähnlichkeitsfunktionen im Intervall $[0, 1]$, sodass Ähnlichkeitsfunktionen sich grundsätzlich für den Einsatz als Merkmal in einem ML-System eignen. Um eine möglichst große Bandbreite unterschiedlicher Eigenschaften abdecken zu können, wird eine Menge von Ähnlichkeitsfunktionen $S = \{s_1, \ldots, s_n\}$ genutzt. Aus S ist dann ein Feature-Vektor zu bilden, indem jede Ähnlichkeitsfunktion auf die Eingangsattributnamen mit jedem Zielattribut angewandt wird. Somit ergibt sich der Feature-Vektor für L_1^{in} als Kreuzprodukt aus S und L^{out} (vgl. Gleichung (5.1)).

$$\begin{aligned} F(L_1^{in}) = \langle\ & s_1(L_1^{in}, L_1^{out}), s_2(L_1^{in}, L_1^{out}), \ldots, s_n(L_1^{in}, L_1^{out}), \\ & s_1(L_1^{in}, L_2^{out}), \ldots, s_n(L_1^{in}, L_2^{out}), \ldots, \\ & s_1(L_1^{in}, L_m^{out}), \ldots, s_n(L_1^{in}, L_m^{out})\ \rangle \end{aligned} \tag{5.1}$$

Abbildung 5.2 zeigt das beschriebene Vorgehen am Beispiel des Eingangsattributs *Product_Name* und ausgewählten Zielattributen. Indem alle Ähnlichkeisfunktionen auf *Product_Name* in Kombination mit allen Zielattributen angewandt werden,

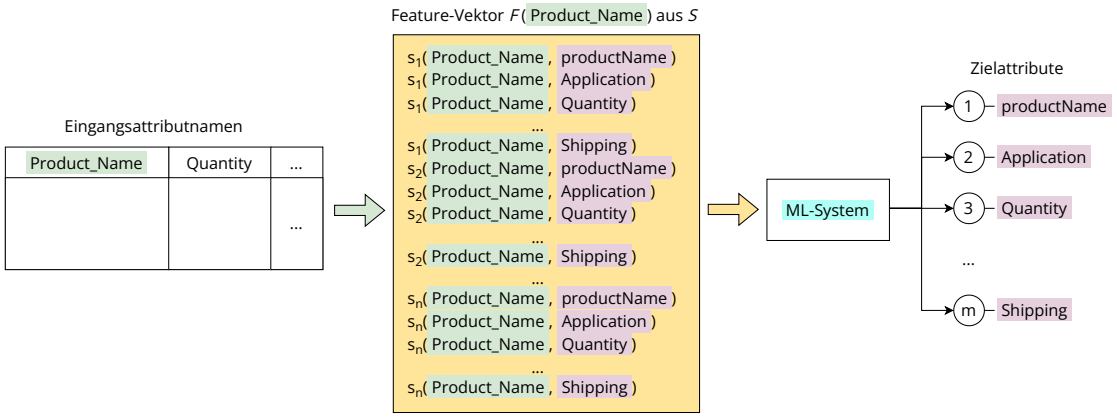


Abbildung 5.2: Bestimmung eines Feature-Vektors aus den Ähnlichkeitsmetriken $s_1, \ldots, s_n \in S$ am Beispiel des Eingangsattributs Product_Name und ausgewählten Zielattributen

ergibt sich ein Feature-Vektor der Größe $|S| \cdot |L^{out}|$ bzw. $|S| \cdot m$. Dieser Vektor kann dann als Eingangswert für ein ML-System genutzt werden, dass die Gewichtungen der Ähnlichkeitsmetriken eigenständig lernt. Somit lassen sich mehrere Metriken nutzen, um die Schwächen einzelner Metriken auszugleichen, ohne dass diese über Heuristiken oder komplexe Regeln miteinander kombiniert werden müssen.

Ein Nachteil dieses Vorgehens ist, dass Erweiterungen des Zielschemas gleichzeitig zu einer Erweiterung des Feature-Vektors führen, da für jedes neu hinzugefügte Zielattribut $|S|$ zusätzliche Ähnlichkeitsberechnungen durchgeführt werden müssen. Dadurch muss zwangsweise auch ein neues Training angestoßen werden. Das zusätzliche Training ist allerdings ohnehin erforderlich, da die neue Zielklasse den Klassifikations-Algorithmen unbekannt ist. Daher muss das Training unabhängig vom genutzten ML-Verfahren von Neuem durchgeführt werden, sobald eine neue Zielklasse hinzukommt und sie vorhergesagt werden soll. Einziger Nachteil bleibt folglich, dass die Länge des Feature-Vektors von der Anzahl der Zielattribute direkt abhängt.

5.2.2 One-Hot-Kodierung und Tf-Idf-Vektoren

Die OH-Kodierung wurde neben TF-IDF in Abschnitt 4.4 bereits als einfaches Verfahren zur Bildung eines Feature-Vektors vorgestellt. Eine mögliche Vorgehensweise ist in Abbildung 5.3 dargestellt. Um einen OH-Vektor zu erhalten, der als Eingangswert

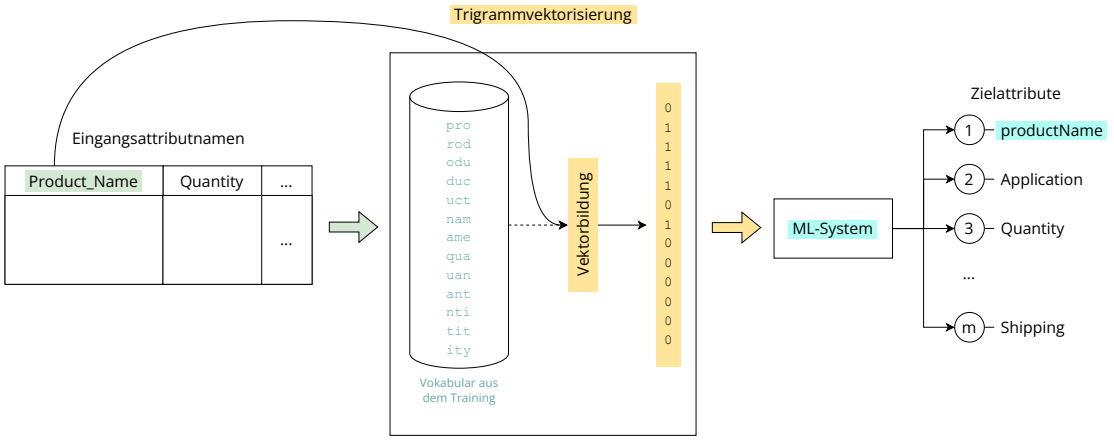


Abbildung 5.3: Anwendung der OH-Kodierung auf den Eingangsattributnamen Product_Name. Um einen OH-Vektor zu bilden, wird der Attributname in Subtoken-Trigramme zerlegt. Danach wird ermittelt, ob die so entstandenen Trigramme im Vokabular enthalten sind. Der so entstehende der Vektor für Product_Name ist 0, wenn ein Trigramm nicht enthalten ist, und 1, falls das Vokabular das Trigramm enthält.

für ein ML-System genutzt werden kann, müssen zunächst die Eingangsattribute – hier `Product_Name` und `Quantity` – in Token zerlegt werden.

Danach können diese Token, wie in der Abbildung zu sehen, beispielsweise weiter in Subtoken-Trigramme zerlegt werden. Sie bilden das Vokabular über das letztendlich der OH-kodierte Vektor erzeugt wird. Dessen Länge entspricht der Anzahl von Trigrammen im Vokabular. Mit den so entstandenen Vektoren als Eingangswerte lässt sich in der Trainingsphase ein Modell aus den bekannten Zuordnungen von Attributnamen trainieren und in der Test- bzw. Produktivphase eine Vorhersage durchführen.

Anstelle der OH-Kodierung kann auch TF-IDF verwendet werden. Am Beispiel in Abbildung 5.3 würde sich in diesem Fall nur die Form der Vektorisierung und damit der entstandene Vektor ändern, sodass dieser die TF-IDF-Werte der Trigramme aus dem Vokabular des Trainingsdatensatzes enthält.

Beide Arten der Vektorbildung führen zu Vektoren der Länge des Vokabulars. Im Gegensatz zum Feature-Vektor bestehend aus Ähnlichkeitsmetriken, ist der entstehende Vektor selbst nicht von den Zielattributen abhängig. Wird ein neues Zielattribut zum ML-System hinzugefügt, muss das Training erneut durchgeführt werden, damit auch das neue Attribut als Klasse vorhergesagt werden kann. Welche Kom-

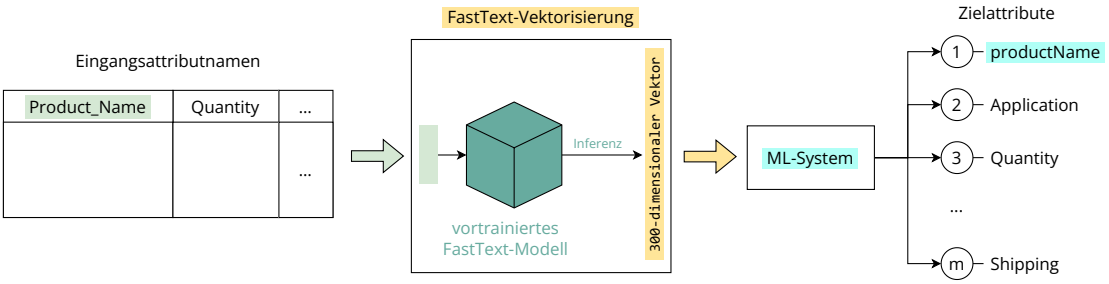


Abbildung 5.4: Nutzen von FastText zur Bestimmung eines Feature-Vectors. Dazu wird das Eingangsattribut als Eingangswert für ein vortrainiertes FastText-Modell genutzt. Das Modell berechnet daraus den 300-dimensionalen Vektor, der als Eingangswert für ein ML-System zur Katalogintegration genutzt werden kann.

binationen aus Token-Bildung, Vektorbildung und ML-Algorithmus im Kontext von Produktkatalogen effektiv sind, wird in Kapitel 6 ermittelt.

5.2.3 FastText

Mithilfe von bereits auf großen Datenmengen trainierten FastText-Modellen, können Wortvektoren von konstanter Größe gebildet werden. In der Regel besitzen FastText-Vektoren aus vortrainierten Modellen 300-Dimensionen[1]. Dafür wird, wie in Abbildung 5.4 dargestellt, der in einen Vektor umzuwandelnde Eingangsattributname als Eingangswert für ein bereits trainiertes FastText-Modell genutzt. Dieses Modell zerlegt das Eingangsattribut (vgl. Abschnitt 4.4), sodass über FastText ein Vektor vorhergesagt werden kann (Inferenz). Diese Vektoren lassen sich genau wie die Vektoren aus den vorherigen Abschnitten als Eingangswerte für ein ML-System nutzen.

Grundsätzlich spiegelt die Ähnlichkeit zwischen FastText-Vektoren zwar die semantische Nähe der zugehörigen Worte wider. Da die vortrainierten Modelle aber mithilfe öffentlicher Daten, wie Nachrichtentexten oder Wikipedia erstellt wurden, kann nicht davon ausgegangen werden, dass die Vektoren den Kontext von Produktdaten berücksichtigen oder das Modell mit besonderen Benennungsstrategien (vgl. Abschnitt 2.5) umgehen kann. Die benötigte Menge von Text für das Training eines FastText-Modells verhindert jedoch ein eigenes Training, da die Attributnamen aus

[1] https://fasttext.cc/docs/en/english-vectors.html

den Katalogen nicht in üblichen Texten oder in den Produktbeschreibungen auftauchen.

5.3 Datengrundlage

Dieser Abschnitt gibt einen Überblick über die Daten, die in Kapitel 6 zur Evaluation verschiedener Kombinationen von Vektorbildung und ML-Verfahren genutzt werden. Die Daten wurden im Rahmen der Forschungskooperation (vgl. Abschnitt 1.2) von antibodies-online gesammelt und zur Verfügung gestellt. Da in diesem Kapitel ausschließlich die Anwendbarkeit von ML auf die Attributnamen-basierte Katalogintegration untersucht wird, besteht der Datensatz aus einem Thesaurus als Goldstandard, der die Eingangsattributnamen verschiedener Zulieferer mit den zugehörigen manuell zugewiesenen Zielattributen des Marktplatzes von antibodies-online beinhaltet.

Insgesamt umfasst der Datensatz 780 Eingangsattributnamen, davon 551 für die 26 Zielattribute mit mehr als 10 verschiedenen Eingangsattributnamen (vgl. Abbildung 5.5). Dabei wird die Groß- und Kleinschreibung vollständig berücksichtigt, sodass die Eingangsattribute `Size`, `SIZE` und `size` zum Beispiel jeweils eigenständige Zuordnungen zu `Quantity` wären. Der Thesaurus enthält ausschließlich Zuordnungen aus echten Integrationsvorgängen. Nicht enthalten sind Informationen, die darüber hinausgehen, wie beispielsweise Informationen zu Datentypen oder ähnlichem.

Folglich sind im Datensatz auch die typischen En- und Dekodierungsprobleme enthalten, beispielsweise bei `Qµantity`, wo µ anstelle von u genutzt wird, falsche Einheiten in Kombination mit Kodierungsproblemen (ug statt µg), ähnliche Eingangsattributnamen für fachlich verschiedene Bezeichner, wie `Antibody_Type1` und `Antibody_Type2`, unterschiedliche Benennungsstrategien (Leerzeichen getrennt, Unterstrich getrennt, CamelCase), fehlende Leerzeichen, sowie Fragmente aus vorherigen Formaten (`<HTML bullet list>`). Tabelle 5.1 enthält eine Auswahl solcher, auch für Menschen eher schwer zu erkennenden Zuordnungen.

Tabelle 5.2 zeigt die im Datensatz enthaltenen Eingangsattribute für das Zielattribut `Hosts`. Hierbei ist klar zu erkennen, dass Schlüsselworte (`Source, Host, Species`) als Teil der Attributnamen existieren, wenn eine Zuordnung zu dem Zielattribut `Hosts` erfolgen sollte. Gleichzeitig wird anhand der Tabelle ersichtlich, dass über die bereits genannten Herausforderungen hinausgehende existieren. Dazu gehört die Verwendung einer anderen Sprache, hier `Wirt` statt `Host`, und ohne

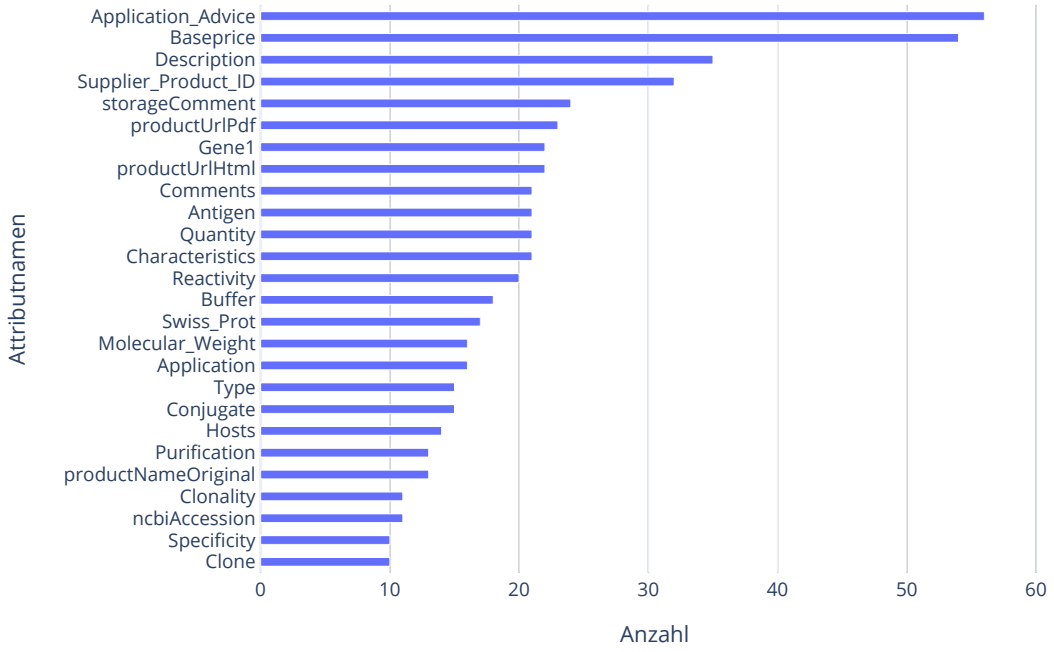


Abbildung 5.5: Zielattribute mit mindestens zehn verschiedenen manuell zugeordneten Eingangsattributnamen

Fachwissen nicht zuzuordnende Eingangsattribute wie zum Beispiel `Expression System` oder eine verbale Beschreibung eines Wirtes, wie `Raised in`. Hierbei wird auch der vielfältige Umgang mit Sprache und Attributnamen deutlich.

Eine weitere Herausforderung im Umgang mit dem Datensatz besteht darin, dass verschiedene Zulieferer gleiche Eingangsattributnamen für verschiedene Zielattribute verwenden, wobei sich die richtige Zuordnung aufgrund eines fehlenden gemeinsamen Konzeptverständnisses nur aus dem Kontext ergibt (vgl. Abschnitt 2.5). So existieren für den Eingangsattributnamen `Source` beispielsweise Zuordnungen zu den Zielattributen `Hosts`, `Comments` und `Immunogen`. Gleichzeitig wird der Plural `Sources` häufig mit dem Zielattribut `Reactivity` in Verbindung gebracht.

Die Ausschnitte verdeutlichen, dass die bereits in Abschnitt 2.5 beschriebenen Herausforderungen im Umgang mit echten Daten, insbesondere bezüglich der Datenqualität, auch auf diesen Datensatz zutreffen.

Eingangsattribut	Zielattribut
Antibody_Type1	Clonality
Antibody_Type2	Type
20ug/$	Baseprice
1 mg/$	Baseprice
Official symbol	Antigen
Tissue Specificity	Desciption
Catalog	Supplier_Product_ID
Cat__No_#	Supplier_Product_ID
Product Features <HTML bullet list>	Comments
Qµantity 1	Quantity
Size (ul)	Quantity

Tabelle 5.1: Attributnamen mit verschiedenen Fehlern und typischen Problemen in der Zuordnung.

Host
Protein_Source
HOST SPECIES
Expression System
Host Animal
SOURCE SPECIES
Source
Host_Species
Raised in
Source / Host
st_attr_host
Wirt
Class/Host

Tabelle 5.2: Eingangsattributnamen für das Zielattribut `Hosts`

5.4 Gewählte Konfiguration der Verfahren

In den folgenden Abschnitten werden die konkreten Verfahren sowie die gewählten Feature-Vektoren für die ML-Verfahren vorgestellt. Zunächst wird auf Referenzansätze eingegangen, mit denen die ML-Verfahren verglichen werden können. Im Anschluss werden die gewählten Konfigurationen für die Feature-Vektoren eingeführt.

5.4.1 Referenzansätze

Um zu bewerten, ob ML-Ansätze für Unternehmen grundsätzlich Verbesserungen in der Katalogintegration bringen können, müssen die ML-Ansätze mit weiteren Ansätzen verglichen werden. Dazu werden zwei einfache Heuristiken und COMA [73] gewählt. Im letzteren Fall kann ein bestehendes, auf komplexen Regeln basierendes Schema-Matching-Tool, die *COMA Community Edition 3.0*[1], auf die Katalogintegration angewendet werden.

Diese Referenzansätze sind geeignet, um zu beurteilen, ob ML-basierte Ansätze sowohl einfachen Heuristiken überlegen sind, als auch komplexen, regelbasierten Werkzeugen, die ebenfalls Ähnlichkeitsfunktionen nutzen.

Einfacher Ansatz über Ähnlichkeiten von Attributnamen

Der einfachste Ansatz zur Ermittlung von zusammengehörigen Attributnamen, ist zu überprüfen, ob ein Eingangsattributname bereits in den Zielattributen L^{out} enthalten ist. In diesem Fall wird die maximale Ähnlichkeit von eins erreicht (vgl. Definition 4.3.1), egal welche Ähnlichkeitsfunktion für den Vergleich genutzt wird. Eine Zuordnung kann in diesem Fall ohne weiteres erfolgen.

Dieses Vorgehen würde im Unternehmenskontext nur funktionieren, wenn in der Marktplatzdomäne bereits ein (de facto) Standard etabliert ist oder alle Zulieferer bereits das Schema des Marktplatzbetreibers nutzen. Trotzdem lässt sich ein Ansatz über eine Ähnlichkeitsfunktion nutzen, sobald nicht mehr geprüft wird, ob ein Eingangsattributname bereits im Zielschema enthalten ist. Stattdessen muss das ähnlichste Zielattribut für jeden Eingangsattributnamen bestimmt werden.

Dieses Vorgehen ist in Listing 5.1 dargestellt. Dabei wird die maximale Ähnlichkeit eines Eingangsattributnamens zu den Attributnamen aus dem Zielschema über die Levenshtein-Ähnlichkeit sim_{LEV} (vgl. Abschnitt 4.3.1) berechnet. Der Rückgabewert

[1] https://sourceforge.net/projects/coma-ce/files/ letzte Version: 24.01.2013

```
    def einfache_referenz(eingangs_name, ziel_attribute):
    ähnlichkeiten = []
    for attribut in ziel_attribute:
        ähnlichkeiten.append(sim_lev(eingangs_name, attribut))
    idx = argmax(ähnlichkeiten)
    return ähnlichkeiten[idx], ziel_attribute[idx]
```

Listing 5.1: Beispielimplementierung des einfachen Ansatzes

der Funktion in Listing 5.1 besteht aus dem Ähnlichkeitswert und zugehörigen Zielattributnamen.

Dieser Referenzansatz ist aufgrund der Einfachheit nur in der Lage, korrekte Zuordnungen zu finden, wenn bereits eine gewisse Ähnlichkeit zwischen den Attributnamen von Eingangs- und Zielschema vorliegt. Grundsätzlich kann die Levenshtein-Ähnlichkeit durch beliebige Ähnlichkeitsfunktionen ausgetauscht werden. Da es sich jedoch nur um einen einfachen Ansatz handelt, kann der Einfluss spezifischer Ähnlichkeitsfunktionen vernachlässigt werden. Dementsprechend sind von diesem Referenzansatz schlechtere Ergebnisse zu erwarten als von COMA oder dem im Folgenden beschriebenen Ansatz über bekannte Zuordnungen aus einem Thesaurus. In der Evaluation sollte dieser einfache Ansatz klar am schlechtesten abschneiden.

Pragmatischer Ansatz über verfügbare Vergleichsdaten

Da Unternehmen, wie beschrieben, häufig bekannte Zuordnungen speichern, können diese als Thesaurus verwendet werden: Immer wenn eine Zuordnung erfolgen soll, kann abgelesen werden, ob für das Attribut aus dem Eingangsschema bereits eine Zuordnung existiert. Somit kann das entsprechende Zielattribut ermittelt werden.

Folglich lässt sich der einfache Ansatz um das Ausnutzen von Vergleichsdaten erweitern. So lässt sich die Ähnlichkeitsfunktion nicht nur auf die Zielattributnamen anwenden, sondern auf alle Elemente aus dem Thesaurus. Dadurch wird auch die Ähnlichkeit eines Eingangsattributnamens zu allen anderen Eingangsattributnamen mit bereits bekannten Zuordnungen bestimmt. Ist die Ähnlichkeit für ein Paar größer als die höchste Ähnlichkeit zu einem Zielattributnamen, kann die Zuordnung über einfaches Ablesen aus den Vergleichsdaten erfolgen. Eine beispielhafte Implementierung findet sich in Listing 5.2.

```
def pragmatischer_ansatz(eingangs_attribut, thesaurus, schwellwert):
    max_similarity = 0
    ziel_attribut = ""
    for synonym in thesaurus[eingangs_attribut]:
        similartiy = sim_lev(synonym, eingangs_attribut)
        if similartiy > max_similarity:
            max_similarity = similartiy
            ziel_attribut = thesaurus[synonym]
return ziel_attribut
```

Listing 5.2: Beispielimplementierung des pragmatischen Ansatzes mit Thesaurus und Ähnlichkeitsmetriken

Dieser Ansatz ist nur wenig komplexer als der zuvor beschriebene einfache Ansatz, berücksichtigt allerdings bereits im Unternehmen vorhandene Daten. Daher wird dieses Vorgehen im weiteren Verlauf als *pragmatischer Ansatz* bezeichnet. Durch das Ausnutzen bereits bekannter Zuordnungen ist er insgesamt vergleichbarer zu ML-basierten Ansätzen als der einfache Ansatz. Die Vergleichbarkeit mit COMA ist ebenfalls gegeben, da auch bei COMA ein Thesaurus hinterlegt werden kann. Um für direkte Vergleichbarkeit mit den ML-Verfahren zu sorgen, wird der Trainingsdatensatz der ML-Verfahren für den pragmatischen Ansatz als Thesaurus verwendet. Des Weiteren wird erneut sim_{LEV} als Ähnlichkeitsfunktion genutzt (vgl. Listing 5.2), wodurch der pragmatische Ansatz direkt mit dem einfachen zu vergleichen ist. Dabei ist zu erwarten, dass Zuordnungen erheblich besser erkannt werden, auch wenn die Bestimmung der Zuordnungen durch umfangreichere Berechnungen länger dauert.

COMA

COMA [73] ist, wie bereits beschrieben ein automatisiertes Werkzeug zur Schemazuordnung und -zusammenführung. Es wurde bis 2013 in unregelmäßigen Abständen weiterentwickelt [74, 20, 46, 73, 75]. COMA wurde hier trotz seines Alters als Vergleich gewählt, da keine vergleichbaren Werkzeuge öffentlich verfügbar waren.

COMA verfolgt zwei Ziele. Erstens sollen Zuordnungen zwischen zwei Schemata ermittelt werden. Zweitens soll aus zwei Schemata ein vereinheitlichtes Schema generiert werden können. Auf letzteres Ziel wird hier nicht weiter eingegangen. COMA nutzt grundsätzlich Struktureigenschaften und Datentypinformationen aus, um Zuordnungen zu bestimmen. Da die vorliegenden Produktkataloge nur aus Attributna-

men in Form von Strings bestehen, können jedoch keine Strukturen und Beziehungen zwischen den Attributnamen, wie sie sich in OWL und XML definieren lassen, genutzt werden. Weiterhin werden in COMA verschiedene Ähnlichkeitsmetriken über Heuristiken orchestriert, um die korrekte Zuordnung iterative zu ermitteln. Zusätzlich erlaubt COMA die Verwendung eines Thesaurus. Detailliertere Informationen zu COMA und der Funktionsweise finden sich in Kapitel 9.

Die Benutzung von COMA kann in der aktuellen Version *COMA Community Edition 3.0*[2] über eine grafische Oberfläche erfolgen. Für die folgende Evaluation wurde zur Vereinfachung eine Programmierschnittstelle (application programming interface (API)) für COMAntwickelt und die Algorithmik in einen Webservice eingebettet, um COMA als Blackbox zu verwenden.

Um eine gute Vergleichbarkeit mit den ML-Verfahren zu gewährleisten, wurde die Konfiguration so gewählt, dass Eingangsattributnamen der Zulieferer bei der Zuordnung berücksichtigt werden. Zusätzlich wurden in der Konfiguration Trigramme erlaubt. Genau wie bei den zuvor beschriebenen Ansätzen wird der Thesaurus aus dem Trainingsdatensatz gebildet und COMA zur Verfügung gestellt. Es ist aufgrund der mangelnden Strukturinformationen bei Produktkatalogen davon auszugehen, dass COMA nur Vergleiche über die Attributnamen beider Schemata durchführt. Dennoch sollte COMA mindestens den einfachen Ansatz übertreffen.

5.4.2 Lernende Verfahren

Die bisher vorgestellten Referenzansätze basieren auf Heuristiken, die im Unternehmenskontext dennoch gute Ergebnisse liefern können. Da zunächst ausschließlich $1:1$ und $N:1$ Beziehungen betrachtet werden, bietet sich wie bereits beschrieben die Modellierung über einen Klassifikationsansatz (siehe Abschnitt 5.1) an. Dazu können die in Abschnitt 3.5 vorgestellten Klassifikationsverfahren verwendet werden, die genau eine Zielklasse vorhersagen können. Die genutzten ML-Verfahren sind:

- Support Vector Machines
- Multilayer-Perceptron
- Random Forests

[2] https://sourceforge.net/projects/coma-ce/files/ letzte Version: 24.01.2013

Diese ML-Verfahren müssen mit unterschiedlichen Feature-Vektoren kombiniert werden, um die erfolgreichsten Kandidaten zu ermitteln. Daher wird im Folgenden der genaue Aufbau der einzelnen Feature-Vektoren beschrieben.

Ähnlichkeitsmetriken als Feature-Vektor

Wie in Abschnitt 5.2.1 beschrieben, können Ähnlichkeitsmetriken als Feature-Vektor genutzt werden, indem die Ähnlichkeiten zwischen einem Eingangsattributnamen mit sämtlichen Zielattributnamen verglichen werden. Es werden die folgenden Ähnlichkeitsmetriken aus Abschnitt 4.3 genutzt:

- **Lexikalische Ähnlichkeit:**
  - Überlappungsähnlichkeit
  - Hamming-Ähnlichkeit
  - zeichenbasierte Kosinus-Ähnlichkeit
  - Levenshtein-Ähnlichkeit
  - Jaro-Winkler
  - Needleman-Wunsch
  - Smith-Waterman
  - LCS-Ähnlichkeit
- **Phonetische Ähnlichkeit:** Editex
- **Semantische Ähnlichkeit:**
  - Paarweise Ähnlichkeit in lexikalischen, semantischen Ressourcen (hier: WordNet[3])
  - Kosinus-Ähnlichkeit über FastText-Vektoren aus dem Standardsprachmodell für Englisch[4]

Durch die Auswahl dieser Ähnlichkeitsfunktionen sollen möglichst viele verschiedene Eigenschaften abgebildet werden.

Abbildung 5.6 visualisiert einen Vektorraum, mit einzelnen gut zu unterscheidenden Clustern. Die einzelnen Punkte entstehen indem die Eingangsattribute aus Abschnitt 5.3 in den Feature-Vektor aus Ähnlichkeitsmetriken transformiert und anschließend über den t-SNE Algorithmus auf zwei Dimensionen reduziert werden.

[3] https://wordnet.princeton.edu/
[4] https://fasttext.cc/docs/en/english-vectors.html

Die Einfärbung erfolgt anhand der Zuordnungen zu Zielattributen aus dem Goldstandard.

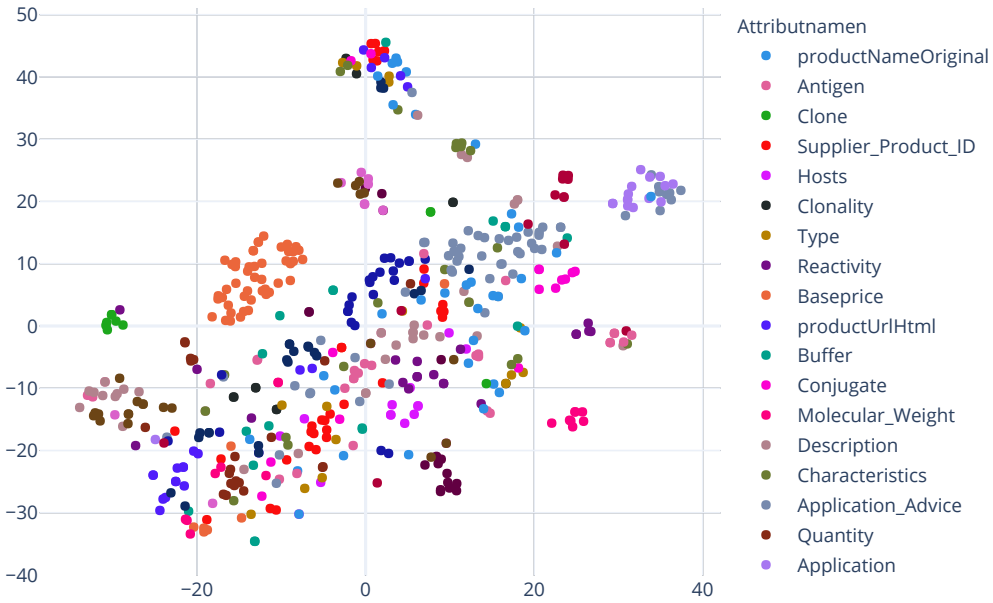


Abbildung 5.6: Visualisierung des Vektorraums durch t-SNE Dimensionsreduktion [76], der auf den Trainingsdaten von Szenario 1 aus Abschnitt 6.1 durch Ähnlichkeitsfunktionen gebildet wurde. Einzelne Punkte zeigen die Lage von Eingangsattributnamen im Vektorraum und fügen sich Teilweise zu größeren Clustern zusammen (z.B. Baseprice).

Die Abbildung verdeutlicht, dass Cluster für verschiedene Zielattribute entstehen. Dabei sind Cluster mit geringerer Varianz in den Eingangsattributnamen stärker ausgeprägt, wie beispielsweise das Cluster für das Zielattribut `Baseprice`. Andere Zielattribute, besitzen eine stärker ausgeprägte Varianz in ihren Eingangsattributnamen. Dadurch bilden sich für die Zielattribute `ProcductNameOriginal` und `Application_Advice` zum Beispiel keine eigenen Cluster, sondern sie treten an wenigen Stellen gehäuft auf, ziehen sich aber als Hintergrundrauschen durch den gesamten Vektorraum. Durch Abbildung 5.6 wird bereits ersichtlich, dass das Er-

lernen einzelner Zuordnungen für ML-Verfahren möglich ist, insbesondere für die Zielattribute, bei denen die Eingangsattributnamen bereits Cluster bilden.

One-Hot-Kodierung und Tf-Idf als Feature-Vektor

Wie in Abschnitt 5.2.2 beschrieben, eignen sich auch die OH-Kodierung und TF-IDF als Feature-Vektor zur Bestimmung von $1:1$ und $N:1$ Beziehungen. Entscheidend für das Ergebnis sind neben dem verwendeten ML-Verfahren, wie Token und N-Gramme gebildet werden. Bei einer ungeeigneten Wahl treten OOV-Fehler häufiger auf.

Die Grundlage der Token- bzw. N-Gramm-Bildung sind die Eingangsattribute aus dem Trainingsdatensatz. Aus diesen wird anschließend das Vokabular für die Vektorbildung der beiden Verfahren aufgebaut. Sowohl die Vektorbildung über die OH-Kodierung als auch durch TF-IDF sind somit auf die Domäne und den Datensatz angepasst.

In der Evaluation werden die folgenden Konfigurationen genutzt:

- **OH:** Subtoken-Trigramme
- **TF-IDF:** Uni-, Bi- und Trigramme aus Subtoken

Die Bildung des OH-Vokabulars aus Subtoken-Trigrammen zielt auf die Vermeidung von häufigen OOV-Fehlern ab. Gleichzeitig sollen möglichst wenige unnütze Textmerkmale im Feature-Vektor auftauchen, um bessere Ergebnisse bei den ML-Verfahren zu erzielen. So würden einzelne, gleichwertige Buchstaben dazu führen, dass der Feature-Vektor unnötig vergrößert wird und das ML-Verfahren mehr Gewichte berücksichtigen muss, was generell zu schlechteren Ergebnissen führen kann.

Bei der Vektorbildung über TF-IDF werden zusätzlich zu Trigrammen auch Uni- und Bigramme verwendet werden, da bei TF-IDF die Bestandteile des Vokabulars entsprechend ihrem Informationsgehalt gewichtet werden (vgl. Abschnitt 4.4.2). Dadurch können auch einzelne Buchstaben oder Sonderzeichen einen besonderen Informationsgehalt besitzen, wie beispielsweise Währungszeichen ($, €, o. ä.).

Abbildung 5.7 zeigt den Vektorraum, der auf dem Trainingsdatensatz von *Szenario 1* durch die Anwendung von TF-IDF entstanden ist. Dabei sind bereits deutliche Unterschiede zum vorherigen Vektorraum in Abbildung 5.6 zu erkennen. Die Cluster der Eingangsattribute lassen sich in den meisten Fällen klar einem Zielattribut zuordnen.

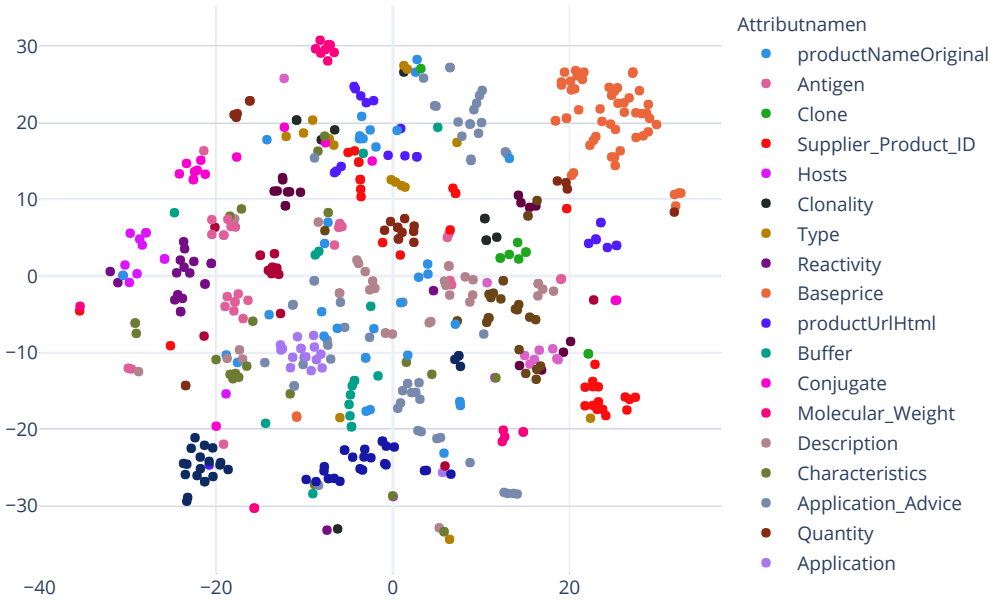


Abbildung 5.7: Visualisierung des Vektorraums durch t-SNE Dimensionsreduktion [76], der auf dem Trainingsdatensatz von Szenario 1 aus Abschnitt 6.1 durch TF-IDF gebildet wurde. Zusammenhängende Cluster sind deutlich zu erkennen.

Nur wenige Zielattribute, wie zum Beispiel *Application_Advice* besitzen kein gut erkennbares Cluster ihrer Eingangsattributnamen. In diesem Fall liegen die Eingangsattributnamen verteilt im Vektorraum. Auch bei anderen Clustern liegen vereinzelt Eingangsattributnamen außerhalb der gewünschten Cluster.

FastText

Für die folgenden Untersuchungen wurde FastText [72] als Beispiel für ein vortrainiertes Modell, das die Semantik von Token besser berücksichtigt als OH- oder TF-IDF-Vektoren, genutzt. So kann überprüft werden, inwiefern sich diese Art der Modelle eignet, um ohne weitere Anpassung Zuordnungen vorherzusagen.

An dieser Stelle wird ein FastText Modell als Grundlage gewählt, das auf der englischsprachigen Version von Wikipedia trainiert wurde.[5] Dieses Modell ermöglicht die Umwandlung von Subtoken in einen 300-dimensionalen Vektor, der direkt als Eingangswert für die drei ML-Verfahren genutzt werden kann (vgl. Abschnitt 5.2.3).

Im Gegensatz zu den vorherigen Ansätzen wird das Modell nicht an den Trainingsdatensatz angepasst, da dieses Sprachmodell wesentlich größere Mengen von Text benötigt als in Attributnamen vorhanden ist. Dadurch wird das Modell nur eingesetzt, um die Token in einen Vektor umzuwandeln. Besteht ein Text aus mehreren Token, ergibt sich bei FastText der zusammengesetzte Feature-Vektor aus dem arithmetischen Mittel der Vektoren der einzelnen Token [72].

Da FastText Modelle Subtoken berücksichtigen, wird die Anzahl der OOV-Fehler zwar verringert. Allerdings wurde das genutzte FastText Modell auf sehr allgemeinen Texten trainiert, sodass diese Fehler insbesondere dann auftauchen können, wenn in den Attributnamen ungewöhnliche Zeichenkombinationen auftauchen.

5.4.3 Auflistung der Verfahren

Abschließend findet sich in Tabelle 5.3 eine Auflistung aller Verfahren bzw. Kombinationen von Vektorisierung und ML-Verfahren, die im folgenden Kapitel evaluiert werden.

[5] wiki-news-300d-1M-subword.vec, `https://fasttext.cc/docs/en/english-vectors.html`

Einfache Ansätze
Referenzansatz
Pragmatisch

Erprobte Ansätze
COMA CE 3.0

ähnlichkeitsbasierte ML-Verfahren
RandomForest
SVM
MLP

wortvektorbasierte ML-Verfahren
TF-IDF mit SVM
TF-IDF mit MLP
OH mit SVM
OH mit MLP
FastText mit SVM
FastText mit MLP

Tabelle 5.3: Auflistung der zu valuierenden Verfahren

Kapitel 6

Evaluation der Zuordnungen über Attributnamen

In diesem Kapitel werden die vorgestellten Referenzverfahren und ML-Verfahren mit unterschiedlicher Vektorisierung aus Abschnitt 5.4 anhand der Daten aus Abschnitt 5.3 miteinander verglichen. Alle ML-Verfahren wurden für die Evaluation mit dem Framework *scikit-learn*[1] implementiert. scikit-learn ist ein Python Framework, etablierte ML-Verfahren, Metriken und einfache Vektorisierungsverfahren, wie OH und TF-IDF, bereits zur Verfügung stehen. Die Evaluation wird in vier verschiedene Szenarien aufgeteilt, um zu beurteilen, welche Verfahren besonders für den zur Verfügung stehenden Goldstandard geeignet sind. Für die Auswertung werden die üblichen Metriken Precision, Recall und der F_1-Score genutzt. Zusätzlich wird neben der reinen Bewertung aus ML-Sicht diskutiert, unter welchen Voraussetzungen sich der Einsatz von ML-Verfahren für Unternehmen lohnen kann.

Die Szenarien sind wie folgt aufgebaut:

Szenario 1 Im ersten Szenario werden die Rohdaten aus dem Goldstandard verwendet. Dadurch wird jeweils ein Eingangsattributname als einzelnes Token gewertet. OH und TF-IDF nutzen also die Eingangsattributnamen als Token, um daraus das entsprechende Vokabular ihrer Subtoken N-Gramme zu bilden. Die Ähnlichkeiten zweier Attributnamen werden dementsprechend zwischen den unveränderten Eingangs- und Zielattributnamen gebildet. unveränderten Namen gebildet

[1] `https://scikit-learn.org/s` in der Version 0.24.1. Die Hyperparameter der Modelle wurden in den Voreinstellungen belassen.

Szenario 2 Hier wird eine erweiterte Tokenisierung der Rohdaten durchgeführt. Die Token werden gebildet, indem die Attributnamen an Trennzeichen, wie _, Bindestrichen, o.ä, und anhand der CamelCase-Schreibweise aufgeteilt werden. Danach folgt die Umwandlung der Token in Kleinbuchstaben. Diese Transformation wird sowohl für die Eingangsattributnamen als auch für die Zielattributnamen durchgeführt. OH und TF-IDF bilden wie zuvor das jeweilige Vokabular aus Subtoken N-Grammen. Für die anderen Verfahren besteht der wesentliche Unterschied in der Transformation in Kleinbuchstaben.

Szenario 3 Szenario 3 erweitert Szenario 1 um die Generierung von synthetischen Daten, um den Trainingsdatensatz zu vergrößern.

Szenario 4 Szenario 4 erweitert Szenario 2 um die Generierung von synthetischen Daten, um den Trainingsdatensatz zu vergrößern.

Abschließend wird eine weitere Auswertung auf einem Testdatensatz durchgeführt, der zu einem späteren Zeitpunkt im Produktivbetrieb erhoben wurde. Dadurch lassen sich Rückschlüsse auf die Anwendbarkeit der Ergebnisse im Unternehmensalltag ziehen.

6.1 Szenario 1: Rohdaten aus dem Goldstandard

Das erste Szenario entspricht dem einfachsten Anwendungsfall für ein Unternehmen. Hier werden die Daten aus dem Goldstandard nicht weiter vorverarbeitet.

6.1.1 Trainings- und Testdaten

Damit die Lern-basierten Ansätze potenziell die Möglichkeit haben bestimmte Eigenschaften zu lernen, wird der Datensatz in diesem Szenario auf die 26 Attributnamen aus dem Zielschema mit über zehn Eingangsattributnamen beschränkt (vgl. Abbildung 5.5). Insgesamt verteilen sich 551 Eingangsattributnamen auf die 26 Zielattributnamen. Die zufällige Aufteilung der Eingangsattributnamen in Trainings- und Testdatensatz erfolgt in einem 80:20 Verhältnis, wobei 80% der Daten für das Training des Modells und 20% zur Validierung verwendet werden. Dadurch enthält der Trainingsdatensatz 440 Eingangsattributnamen und der Testdatensatz 111 Eingangsattributnamen.

Damit die Evaluierung aussagekräftig bleibt, müssen hier zusätzliche Eigenschaften beachtet werden. Wie Abbildung 5.5 zeigt, variiert die Anzahl der Zuordnungen im Datensatz zwischen 58 für `Application_Advice` und zehn für `Clone`. Diese relativen Häufigkeiten der Zuordnungen zu Zielattributnamen sollte möglichst beim Aufteilen in Trainings- und Testdatensatz erhalten bleiben (engl. „stratified split"), damit die Evaluierung aussagekräftig ist. Das heißt, von den 58 Eingangsattributnamen für `Application_Advice` sollte der Trainingsdatensatz 46 enthalten und der Testdatensatz 12.

Werden die relativen Häufigkeiten nicht eingehalten, könnte beispielsweise der Testdatensatz in einem sehr ungünstigen Fall nur aus den Eingangsattributnamen für `Application_Advice` und `Baseprice` bestehen. Im Trainingsdatensatz fänden sich dagegen nur vereinzelt Beispiele für diese Zuordnungen. Dadurch könnten die ML-Verfahren die Eigenschaften dieser Eingangsattributnamen nicht lernen und dadurch die Zuordnungen nicht bestimmen. Andersherum könnte nicht evaluiert werden, wie gut Zuordnungen zu anderen Zielattributnamen gelernt werden konnten, da keine Beispiele von deren Eingangsattributnamen im Testdatensatz enthalten sind. Die Aufteilung der Eingangsattributnamen in die Datensätze erfolgt daher zufällig, aber anhand der relativen Häufigkeiten der Zuordnungen zu den Zielattributnamen.

6.1.2 Auswertung auf dem Testdatensatz

Tabelle 6.1 zeigt die gewichteten Mittelwerte der Klassifikationsergebnisse auf dem Testdatensatz. Dabei erreicht der einfache Referenzansatz, wie erwartet, die schlechtesten Ergebnisse mit einer Precision von 0.394 und einem F_1-Score von 0.305. Auch der Recall dieses Ansatzes ist mit 0.351 mit Abstand der geringste von allen getesteten. Der pragmatische Ansatz schneidet im Vergleich deutlich besser ab. Hierbei werden mit einer Precision von 0.67, einem Recall von 0.631 und einem F_1-Score von 0.623 Ergebnisse erzielt, die sogar ähnlichkeitsbasierte Lernansätze übertreffen.

Wortvektorbasierte Lernansätze – mit Ausnahme von FastText – erreichen bessere Ergebnisse als ähnlichkeitsbasierte Ansätze, wobei TF-IDF in der Kombination mit einer SVM die höchste Precision (0.709) erreicht und bezüglich Recall und F_1-Score das Niveau des pragmatischen Ansatzes hält. Die beiden anderen wortvektorbasierten SVMs (OH und FastText) schneiden bezüglich der Precision deutlich schlechter ab als TF-IDF. Auch die anderen Kennwerte liegen teilweise unterhalb der ähnlichkeitsbasierten Ansätze.

	P	R	F_1
Einfache Ansätze			
Referenzansatz	0.394	0.351	0.305
Pragmatisch	0.670	0.631	0.623
Erprobte Ansätze			
COMA CE 3.0	0.524	0.486	0.455
ähnlichkeitsbasierte ML-Verfahren			
RandomForest	0.630	0.622	0.597
SVM	0.621	0.613	0.599
MLP	0.626	0.568	0.577
wortvektorbasierte ML-Verfahren			
TF-IDF mit SVM	**0.709**	0.631	0.624
TF-IDF mit MLP	0.680	**0.658**	0.650
OH mit SVM	0.652	0.604	0.602
OH mit MLP	0.678	**0.658**	**0.653**
FastText mit SVM	0.628	0.604	0.578
FastText mit MLP	0.639	0.613	0.611

Tabelle 6.1: Vergleich der gewichteten Mittelwerte in Szenario 1 über Precision, Recall und den F_1-Score aus dem Testdatensatz. Die jeweils besten Ergebnisse sind hervorgehoben.

Bei der Anwendung von Wortvektoren in Kombination mit einem MLP werden die höchsten Recall- und F_1-Werte erreicht (OH und TF-IDF), wobei die Precision geringfügig höher ist als beim pragmatischen Ansatz. Insgesamt schneiden die einfachen, auf den Datensatz angepassten Vektorisierungsverfahren deutlich besser ab, als das für einen allgemeinen Zweck vortrainierte FastText-Modell.

Die Lernansätze, die auf der Ähnlichkeit von Attributnamen basieren, erzielten Ergebnisse im Mittelfeld. Alle ähnlichkeitsbasierten ML-Verfahren schneiden bei der Klassifikationsaufgabe schlechter ab als der pragmatische Ansatz. Der Vergleich mit den Ergebnissen von COMA zeigt, dass alle lernenden Ansätze auch COMA überlegen sind.

6.2 Szenario 2: Goldstandard mit Tokenisierung und Kleinschreibung

Im Gegensatz zu Abschnitt 6.1 werden die Eingangsattributnamen aus dem Zielschema für Szenario 2 wie beschrieben vorverarbeitet.Um die beiden Szenarien

	P	R	F_1
Einfache Ansätze			
Referenzansatz	0.352	0.316	0.297
Pragmatisch	0.585	0.537	0.526
Verwandte Ansätze			
COMA CE 3.0	0.520	0.486	0.453
Ähnlichkeits-basierte Lernansätze			
RandomForest	0.544	0.568	0.526
SVM	0.627	0.589	0.579
MLP	0.599	0.589	0.575
Wortvektor-basierte Ansätze			
TF-IDF mit SVM	0.704	0.632	0.624
TF-IDF mit MLP	**0.724**	**0.663**	**0.658**
OH mit SVM	0.693	0.610	0.614
OH mit MLP	0.663	0.652	0.634
FastText mit SVM	0.587	0.568	0.529
FastText mit MLP	0.627	0.579	0.570

Tabelle 6.2: Vergleich der gewichteten Mittelwerte über Precision, Recall und den F_1-Score aus dem Testdatensatz in Szenario 2. Die jeweils besten Ergebnisse sind hervorgehoben.

effektiv vergleichen zu können, wird die gleiche Aufteilung der Trainings- und Testdaten wie in Szenario 1 gewählt.

Wie der Vergleich von Tabelle 6.1 und Tabelle 6.2 zeigt, fallen die Ergebnisse der meisten Ansätze mit Vorverarbeitung deutlich schlechter aus als zuvor. Die Ausnahme bilden Wortvektor-basierte Ansätze mit TF-IDF und OH. Die Ergebnisse von COMA stagnieren. Ein Einfluss der Vorverarbeitung ist für COMA nicht zu erkennen.

Das beste Ergebnis auf dem Testdatensatz erreicht die Kombination aus TF-IDF Vektorisierung in Kombination mit einem MLP mit einer Precision von 0.724, einem Recall von 0.663 und einem F_1-Score von 0.658. Diese Werte bilden nicht nur das Maximum von Szenario 2, sondern übertreffen auch die bisherigen Ergebnisse aus Szenario 1.

Bei der Betrachtung von Tabelle 6.2 fällt zusätzlich auf, dass die Vektorisierung mit TF-IDF bessere Ergebnisse erzielt als die OH-Vektorisierung. Außerdem erreicht ein MLP bei allen Wortvektorisierungsverfahren einen höheren Recall als eine SVM, sowie einen besseren F_1-Score.

6.3 Vergleich der Szenarien und Ergebnisanalyse

Auf Grundlage der Ergebnisse von Szenario 1 und Szenario 2 lässt sich noch nicht abschließend beurteilen, welches ML-Verfahren für das Unternehmen am besten funktioniert, da die zufällige Wahl von Trainings- und Testdaten eine bestimmte Kombination von Verfahren und Feature-Vektor bevorzugen könnte. Die bisherigen Ergebnisse zeigen lediglich,

1. dass eine Vorverarbeitung zu einer Verbesserung führen kann,

2. und dass wortvektorbasierte Ansätze zu besseren Ergebnissen führen können als andere Ansätze.

In diesem Abschnitt werden daher die Ergebnisse hinsichtlich ihrer Stabilität untersucht. Zusätzlich werden häufige Probleme, wie z. B. Verwechslung der Klassen, bei der Klassifikation analysiert.

6.3.1 Analyse der Vorhersageergebnisse

Um ein besseres Verständnis für falsch klassifizierte Attributnamen zu bekommen, zeigt Abbildung 6.1 die normalisierte Konfusions- oder Wahrheitsmatrix des TF-IDF mit MLP aus Szenario 2. Die Konfusionsmatrizen der anderen Kombinationen besitzen grundsätzlich die gleichen Charakteristika.

In der Konfusionsmatrix werden die richtigen Label aus den Testdaten mit den entsprechenden Vorhersagen verglichen. Dadurch wird deutlich, welche Klassen (Attributnamen) besonders häufig falsch vorhergesagt werden. Zusätzlich lassen sich allgemeine Probleme erkennen.

In Abbildung 6.1 sind zwei Auffälligkeiten hervorzuheben:

1. Die Diagonale ist in den meisten Fällen ausgeprägt. Hier spiegeln sich die Ergebnisse der vorangegangenen Abschnitte wider. Eine Ausnahme bilden beispielsweise `Hosts` und `Comments`, die das Modell nicht erlernen konnte.

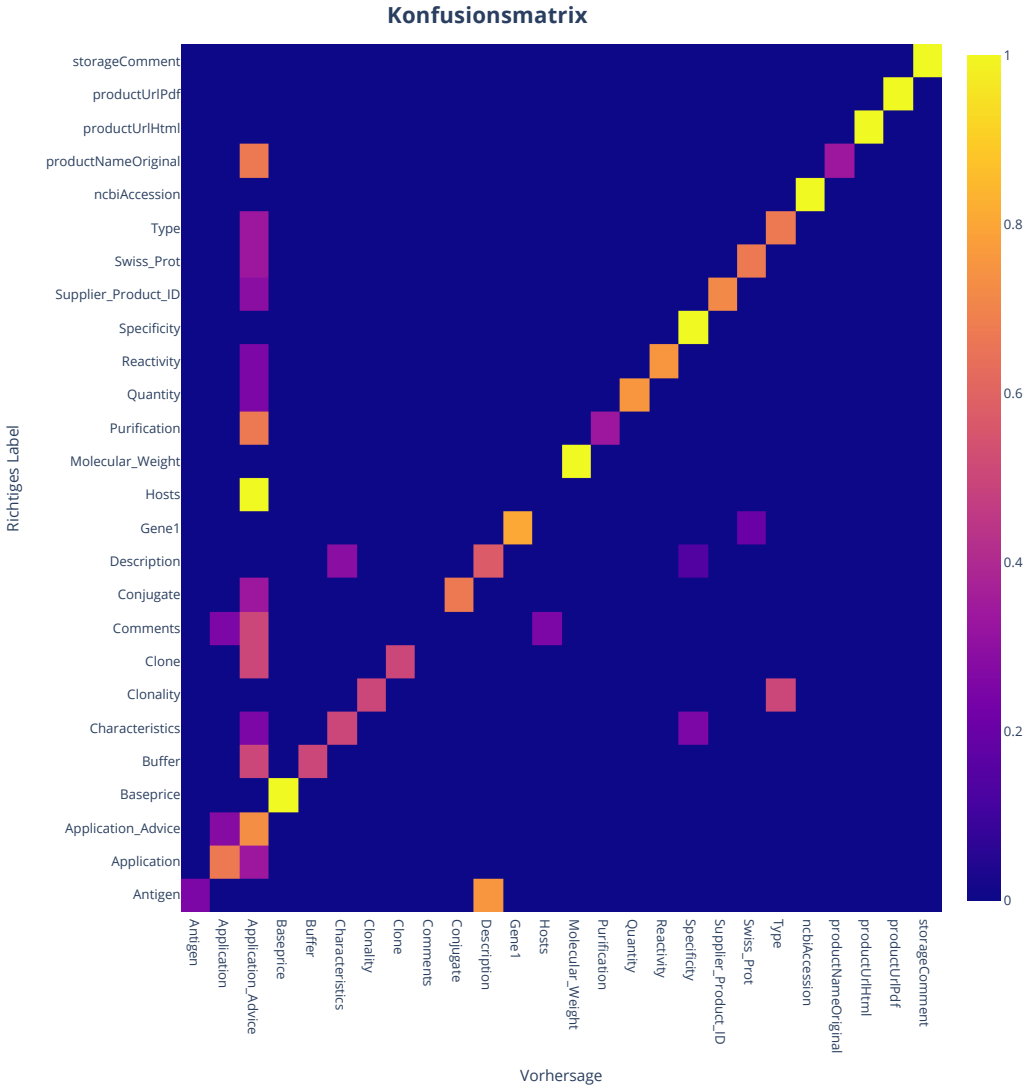


Abbildung 6.1: Normalisierte Konfusionsmatrix der Vorhersageergebnisse des TF-IDF mit MLP aus Szenario 2 auf dem Testdatensatz. Die Matrix zeigt zwei signifikante Beobachtungen: Erstens ist die zu erwartende diagonale Ausprägung der korrekten Vorhersagen vorhanden. Zweitens existiert ein starker Bias in Richtung `Application_Advice`.

2. Die Ausprägung in der Senkrechten zeigt, dass Zuordnungen im Zweifel als `Application_Advice` klassifiziert wird. Eine Erklärung ist die hohe Anzahl von Eingangsattributnamen und deren Varianz in der Benennung. Dies ist auch in den Vektorraumdarstellungen (vgl. Abbildung 5.6 und Abbildung 5.7)

Expression System
Raised in
Wirt

Tabelle 6.3: Eingangsattributnamen für das Zielattribut `Hosts` ***im Testdatensatz.***

zu sehen. Die Eingangsattributnamen für `Application_Advice` liegen in den Vektorräumen neben kleineren Gruppierungen auch als Grundrauschen zwischen anderen Gruppierungen vor.

Insbesondere die Eingangsattributnamen für den Attributnamen `Hosts` werden häufig falsch vorhergesagt, obwohl Tabelle 5.2 auf Seite 107 auf den ersten Blick ähnliche Eingangsattributnamen enthält. Bei Betrachtung der Eingangsattributnamen aus dem Testdatensatz (vgl. Tabelle 6.3) zeigt sich, dass die zufällige Datenaufteilung ungünstig ist, da sich zufällig drei außergewöhnliche Eingangsattributnamen im Testdatensatz befinden. Daher wird im folgenden Abschnitt die Stabilität der Ergebnisse betrachtet.

6.3.2 Ergebnisstabilität

Aufgrund der vorangegangenen Analyse muss neben der Bewertung der Ansätze auf dem Testdatensatz auch eine Überprüfung hinsichtlich der Ergebnisstabilität erfolgen. Dazu erfolgt eine Überkreuzvalidierung von verschiedenen Aufteilungen des Datensatzes:

Der Datensatz wird in fünf gleich große Teile gespalten, die jeweils die relativen Häufigkeiten der Zuordnungen aus dem Goldstandard beibehalten. Zum Training werden danach jeweils vier der fünf Teile verwendet und der verbleibende Teil zur Evaluation genutzt. Dadurch werden insgesamt fünf Modelle trainiert und evaluiert.

Tabelle 6.4 zeigt die Ergebnisse dieser fünffachen Überkreuzvalidierung für die beiden vorangegangenen Szenarien. Sie enthält die gewichteten Mittelwerte mit der Standardabweichung von Precision, Recall und F_1-Score.

Bei Betrachtung von Tabelle 6.4 lässt sich feststellen, dass eine Datenaufbereitung zur Stabilität der Vorhersagen beiträgt (kleinere Standardabweichungen in Szenario 2), aber die Mittelwerte bei MLP-basierten Klassifizierungsmethoden nur sehr geringfügig beeinflusst. Größere Unterschiede zwischen den beiden Szenarien zeigen sich bei der SVM-basierten Klassifikation. Während die OH- und FastText-basierte SVM in Szenario 2 im Bereich Precision erheblich schlechter abschneiden

	P	R	F_1
Szenario 1			
TF-IDF mit SVM	**0.71 ± 0.06**	0.62 ± 0.08	0.61 ± 0.07
TF-IDF mit MLP	0.69 ± 0.08	0.68 ± 0.06	0.67 ± 0.07
OH mit SVM	0.69 ± 0.05	0.61 ± 0.07	0.59 ± 0.06
OH mit MLP	0.70 ± 0.07	**0.69 ± 0.05**	**0.68 ± 0.06**
FastText mit SVM	0.69 ± 0.05	0.61 ± 0.07	0.59 ± 0.06
FastText mit MLP	0.70 ± 0.07	**0.69 ± 0.05**	**0.68 ± 0.06**
Szenario 2			
TF-IDF mit SVM	**0.71 ± 0.04**	0.68 ± 0.05	0.66 ± 0.05
TF-IDF mit MLP	0.69 ± 0.04	**0.69 ± 0.04**	**0.67 ± 0.05**
OH mit SVM	0.66 ± 0.06	0.64 ± 0.05	0.62 ± 0.05
OH mit MLP	0.70 ± 0.04	0.68 ± 0.04	0.66 ± 0.04
FastText mit SVM	0.64 ± 0.03	0.60 ± 0.05	0.57 ± 0.04
FastText mit MLP	0.70 ± 0.04	0.68 ± 0.04	0.66 ± 0.04

Tabelle 6.4: Gewichtete Mittelwerte mit Standardabweichung bei einer fünffachen Überkreuzvalidierung für die wortvektorbasierten Verfahren aus Szenario 1 und Szenario 2

(vgl. 0.69 zu 0.66 bzw. 0.64), verbessern sich der Recall und der F_1-Score von OH mit SVM. Die FastText-basierte Vektorisierung schneidet dagegen in allen Bereichen am schlechtesten ab.

Im Gegensatz zu den anderen Vektorisierungsmethoden erreicht TF-IDF in Kombination mit der SVM in Szenario 2 eine deutliche Steigerung bei gleichzeitiger Stabilisierung der Klassifikationsergebnisse. Dadurch erreicht die Kombination von SVM mit TF-IDF Werte auf dem Niveau der MLP-basierten Kombinationen.

Dennoch bleibt festzuhalten, dass ein MLP-basierter Ansatz in der Regel unabhängig von der Vektorisierung zu besseren Ergebnissen führt. Mindestens liefert ein MLP-basierter Ansatz jedoch gleich gute Ergebnisse wie ein SVM-basierter Ansatz. Die Datenaufbereitung zeigt bei der MLP-basierten Klassifikation nur einen geringfügigen Einfluss auf die Stabilität der Ergebnisse.

6.4 Szenario 3: Original Datensatz mit Datenerweiterung

In diesem Szenario wird der Datensatz aus Szenario 1 modifiziert. Das Ziel dieses Szenarios ist, zu überprüfen, ob eine synthetische Erweiterung des Datensatzes zu besseren Ergebnissen in der Praxis führt, weil darauf trainierte Modelle besser generalisieren und mehr Zuordnungen zu Zielattributnamen lernen können. Dieses Vorgehen wird oft für das Training von ML-Modellen in der Sprach- oder Bildverar-

beitung genutzt, wenn der Trainingsdatensatz nur eine kleine Anzahl von Beispielen enthält [77].

6.4.1 Trainings- und Testdatensatz

Der Datensatz für Szenario 3 wird durch eine synthetische Erweiterung des Datensatzes aus Szenario 1 gebildet. Dazu wurden Trennzeichen (z. B. _, -, oder ␣) in den Daten ausgetauscht oder entfernt und miteinander gemischt. Zusätzlich wurde die Klein- und Großschreibung der Eingangsattributnamen variiert. Bei der Verwendung des erweiterten Datensatzes muss allerdings darauf geachtet werden, dass die Erweiterung möglicherweise zu Duplikaten geführt hat, Eingangsattributnamen und deren Zuordnung also mehrfach im Datensatz auftritt. Diese Duplikate müssen vor der Aufteilung in Trainings- und Testdatensatz entfernt werden, um zu vermeiden, dass gleiche Eingangsattributnamen sowohl im Trainings- als auch im Testdatensatz vorhanden sind.

Der so entstandene Datensatz ohne Duplikate enthält 3367 Eingangsattributnamen (vgl. vorher 551) und deren Zielattributnamen. Aufgrund der Datenerweiterung umfasst der Datensatz Zuordnungen zu 86 Zielattributen. Die Aufteilung in Trainings- und Testdatensatz erfolgt erneut zufällig unter Beibehaltung der relativen Häufigkeit der Zuordnungen. Genau wie in den vorherigen Szenarien werden 80 % des Datensatzes als Trainingsdaten und 20 % als Validationsdaten genutzt.

6.4.2 Auswertung auf dem Testdatensatz

Tabelle 6.5 zeigt die Ergebnisse der getesteten Ansätze in Szenario 3. Grundsätzlich sind die erreichten Vergleichswerte deutlich höher als in den vorherigen Szenarien, was durch die Datenerweiterung begründet ist. Die Datenerweiterung führt zu ähnlicheren Eingangsattributnamen in Trainings- und Testdatensatz und somit gleichzeitig zu einer besseren Erkennungsrate. In der Praxis ist dieses Verhalten erwünscht, da Eingangsattributnamen, die ähnlich zu einem bereits existierenden Eingangsattributnamen sind, richtig erkannt werden sollen. Genau dieses Verhalten ist bei der Evaluation durch die Validationsdaten zu erkennen.

Dadurch sind auch die Ergebnisse der Ähnlichkeit-basierten Ansätze zu erklären, die durchweg besser abschneiden als wortvektorbasierte Verfahren. Hierbei ist besonders hervorzuheben, dass auch der pragmatische Ansatz wesentlich besser abschneidet als zuvor. Die Datensätze enthalten also tatsächlich sehr ähnlich geschrie-

	P	R	F_1
Einfache Ansätze			
Referenzansatz	0.297	0.203	0.210
Pragmatisch	0.912	0.908	0.904
Erprobte Ansätze			
COMA CE 3.0	0.260	0.334	0.268
Ähnlichkeitsbasierte Lernansätze			
RandomForest	**0.925**	**0.926**	**0.920**
SVM	0.905	0.909	0.903
MLP	0.896	0.892	0.888
Wortvektorbasierte Lernansätze			
TF-IDF mit SVM	0.816	0.799	0.788
TF-IDF mit MLP	0.889	0.885	0.880
OH mit SVM	0.791	0.767	0.755
OH mit MLP	0.892	0.890	0.885
FastText mit SVM	0.665	0.639	0.621
FastText mit MLP	0.681	0.666	0.659

Tabelle 6.5: Vergleich der gewichteten Mittelwerte über Precision, Recall und F_1-Score auf dem Testdatensatz aus Szenario 3. Die jeweils besten Ergebnisse sind hervorgehoben.

bene Eingangsattributnamen, sodass Ansätze, die auf maschinellem Lernen basieren, so gut wie keinen Vorteil haben.

Der beste Lernansatz ist der *Random-Forest*-Klassifizierer in Kombination mit Ähnlichkeitsmetriken. Nur dieser Ansatz erreicht bessere Werte als der pragmatische Ansatz.

Eine weitere Erkenntnis aus den Ergebnissen in Tabelle 6.5 ist, dass bei wortvektorbasierten Verfahren erneut MLPs bessere Ergebnisse erzielen als SVMs. Ebenso schneiden erneut die TF-IDF- und OH erheblich besser ab als FastText.

Ein unerwartetes Ergebnis ist das Abschneiden von COMA. COMA liegt mit einer Precision von 0.26 sogar unter dem einfachen Referenzansatz. Eine mögliche Erklärung wäre, dass COMA bei der Mehrheit der Eingangsattributnamen die Tokenisierung mehrerer Worte nicht korrekt durchführen kann.

Eine andere mögliche Erklärung ist die Komplexität der Eingangsattributnamen. Da COMA in anderen Anwendungen zusätzlich Strukturinformationen und Datentypen in die Entscheidungsfindung mit einbeziehen kann, könnte es durch das Fehlen dieser Informationen auf einfache Ähnlichkeitsfunktionen wie die Levenshtein-Ähnlichkeit zurückgreifen. Dies würde auch erklären, warum COMA insbesonde-

	p	r	f1
Szenario 3			
RandomForest	0.69 ± 0.07	0.66 ± 0.07	0.65 ± 0.06
TF-IDF mit SVM	0.70 ± 0.07	0.64 ± 0.06	0.63 ± 0.06
TF-IDF mit MLP	**0.75 ± 0.05**	**0.72 ± 0.06**	**0.71 ± 0.06**
OH mit SVM	0.66 ± 0.08	0.61 ± 0.06	0.59 ± 0.06
OH mit MLP	0.74 ± 0.06	**0.72 ± 0.06**	0.70 ± 0.06
FastText mit SVM	0.55 ± 0.05	0.53 ± 0.05	0.50 ± 0.05
FastText mit MLP	0.54 ± 0.05	0.52 ± 0.06	0.51 ± 0.05

Tabelle 6.6: Gewichtete Mittelwerte mit Standardabweichung einer fünffachen Überkreuzvalidierung der wortvektorbasierten Verfahren aus Szenario 3

re bei vergleichsweise einfach zu erkennenden Zuordnungen für Zielattribute wie `Baseprice` sehr gute (Precision 0.906, Recall 0.889, F_1-Score 0.897) Ergebnisse erzielt, aber viele andere Zuordnungen nicht ermitteln kann.

6.4.3 Auswertung der Überkreuzvalidierung

Durch die Datenaufbereitung und -erweiterung lassen sich aus den bisherigen Ergebnissen keine Schlüsse auf die Einsetzbarkeit in der Praxis ableiten. Genau wie für die beiden vorherigen Szenarien (vlg. Abschnitt 6.3.2) muss die Ergebnisstabilität betrachtet werden. Aufgrund der sehr guten Ergebnisse in der ersten Auswertung (Tabelle 6.5) wird zusätzlich zu wortvektorbasierten Ansätzen, der ähnlichkeitsbasierte Ansatz mit Random-Forest-Klassifikator in die Evaluation mit einbezogen.

Auch Tabelle 6.6 zeigt, dass die OH- bzw. TF-IDF-Vektorisierung auf den vorhandenen Daten deutlich effektiver ist, als eine Vektorisierung über FastText. Durch die Überkreuzvalidierung auf dem erweiterten Datensatz wird erneut deutlich, dass MLP-basierte Lernverfahren besser geeignet sind als SVM-basierte Verfahren. Zusätzlich schneidet der ähnlichkeitsbasierte Ansatz mit Random-Forest-Klassifikator bei der Überkreuzvalidierung schlechter ab, als die MLP-basierten Verfahren mit Ausnahme von FastText.

6.5 Szenario 4: Original Datensatz mit Vorverarbeitung und Datenerweiterung

Für Szenario 4 wurde der Datensatz aus Szenario 3 (vgl. Abschnitt 6.4) auf die gleiche Weise aufbereitet wie in Szenario 2 (vgl. Abschnitt 6.2). Daher liegen für Szenario 4

	P	R	F_1
Einfache Ansätze			
Referenzansatz	0.367	0.329	0.303
Pragmatisch	**0.894**	**0.877**	**0.871**
Erprobte Ansätze TODO			
COMA CE 3.0	0.264	0.337	0.271
Ähnlichkeits-basierte Lernansätze			
RandomForest	0.784	0.786	0.765
SVM	0.855	0.845	0.839
MLP	0.704	0.706	0.692
Wortvektor-basierte Lernansätze			
TF-IDF mit SVM	0.733	0.758	0.724
TF-IDF mit MLP	0.879	0.869	0.860
OH mit SVM	0.719	0.75	0.712
OH mit MLP	0.852	0.845	0.835
FastText mit SVM	0.591	0.643	0.585
FastText mit MLP	0.775	0.766	0.757

Tabelle 6.7: Vergleich der gewichteten Mittelwerte über Präzision, Recall und F_1-Score auf dem Testdatensatz. Die jeweils besten Ergebnisse sind hervorgehoben.

alle Eingangsattributnamen nur noch in Kleinbuchstaben und mit Leerzeichen getrennt vor. Da für Szenario 3 der Datensatz unter anderem durch die Variation der Trennungszeichen erweitert wurde, mussten vor der Auswertung der Ergebnisse alle so entstandenen Duplikate aus dem Datensatz entfernt werden.

Somit liegen leichte Abweichungen zu Szenario 3 vor: Statt 3367 Eingangsattributnamen enthält der Datensatz nur noch 1262. Zusätzlich sind fünf Zielattributnamen weniger erlernbar (86 in Szenario 3, 81 in Szenario 4), weil nach Entfernung der Duplikate keine ausreichende Anzahl an Eingangsattributnamen mehr vorhanden ist.

Die Ergebnisse der Verfahren können Tabelle 6.7 entnommen werden. Bemerkenswert sind die folgenden Ergebnisse:

- Fast alle Ansätze erreichen schlechtere Ergebnisse, wenn die Daten aufbereitet wurden. Die Ausnahme dazu bilden der einfache Referenzansatz, COMA und FastText-Vektorisierung in Kombination mit einem MLP.
- Der pragmatische Ansatz funktioniert besser als alle anderen Verfahren.
- Wortvektorbasierte Verfahren mit MLP-Klassifikator funktionieren erneut besser als SVM-Klassifikatoren.

	P	R	F_1
Szenario 4			
RandomForest	0.60 ± 0.09	0.62 ± 0.08	0.58 ± 0.08
TF-IDF mit SVM	0.63 ± 0.05	0.60 ± 0.04	0.57 ± 0.04
TF-IDF mit MLP	**0.72 ± 0.05**	**0.70 ± 0.06**	**0.68 ± 0.05**
OH mit SVM	0.60 ± 0.04	0.58 ± 0.03	0.55 ± 0.03
OH mit MLP	0.72 ± 0.06	**0.70 ± 0.06**	0.68 ± 0.06
FastText mit SVM	0.53 ± 0.04	0.55 ± 0.06	0.51 ± 0.05
FastText mit MLP	0.63 ± 0.07	0.62 ± 0.07	0.61 ± 0.07

Tabelle 6.8: Gewichtete Mittelwerte mit Standardabweichung einer fünffachen Überkreuzvalidierung der Wortvektor-basierten Verfahren aus Szenario 4

- Ähnlichkeitsbasierte und wortvektorbasierte Verfahren funktionieren ungefähr gleich gut. Je nach verwendetem Klassifikator variieren die Ergebnisse.

Ähnliche Ergebnisse zeigen sich in der Auswertung der Überkreuzvalidierung in Tabelle 6.8. Mit Ausnahme von FastText mit MLP erreicht keine Kombination die vorherigen Ergebnisse. Im Gegensatz zu den Ergebnissen aus Tabelle 6.4 findet durch eine Datenaufbereitung des erweiterten Datensatzes keine Stabilisierung der Ergebnisse statt.

6.6 Bewertung der Ergebnisse und Analyse möglicher Fehler

In den folgenden Abschnitten werden die Ergebnisse der Verfahren in den vier Szenarien zusammengefasst und bewertet. Die Bewertung erfolgt immer in Bezug auf den untersuchten Anwendungsfall im Antikörpermarkt.

6.6.1 Bewertung der Wortvektoren

Für die Vorhersage von Zuordnungen zwischen Eingangs- und Zielattributen bilden OH- bzw. TF-IDF-Vektoren mit MLP-Klassifikator die beste Grundlage. Bisher ist nicht eindeutig zu beziffern, welche der beiden Vektorisierungsmethoden in der Praxis effektiver ist. Dieses Ergebnis ist interessant, da eigentlich zu erwarten war, dass die FastText-Vektorisierung bessere Ergebnisse liefert.

Begründet werden kann diese Erkenntnis mit der Besonderheit von Attributnamen. FastText funktioniert sehr gut auf Sätzen (also komplexerer Sprache). Das

zur Vektorisierung verwendete FastText-Modell wurde auf vollständigen Sätzen trainiert, um die Wortvektoren zu berechnen. Der gewählte Ansatz, auf dieser Basis ein MLP zu trainieren, um Zielattribute zu identifizieren, könnte dazu geführt haben, dass die Wortvektoren in einem ungeeigneten Kontext gebildet wurden.

Im Gegensatz dazu arbeiten die einfacheren Wortvektorisierungsverfahren auf dem speziellen Vokabular der Eingangsattributnamen. Hier könnte dementsprechend eine Benachteiligung von FastText vorliegen. Allerdings ist ein Anpassen von FastText in den Szenarien aufgrund der ausschließlichen Verwendung von Attributnamen nicht möglich.

6.6.2 Bewertung von COMA

Die Ergebnisse von COMA übertreffen in den meisten hier vorgestellten Szenarien lediglich den einfachsten Ansatz. Hierfür sind mehrere mögliche Ursachen auszumachen. COMA wurde darauf ausgelegt neben der Ähnlichkeit von Worten auch zusätzliche Informationen, wie z. B. Struktur des Schemas und Datentypen der Attribute auszunutzen, da der angedachte Anwendungsbereich unter anderem im Datenbankschema-Abgleich lag. Bei der Integration der Produktkataloge liegen solche Informationen nicht vor, da alle Attributnamen lediglich als Zeichenkette interpretiert werden. Weiterhin könnte die Menge und die Ähnlichkeit der Zielattribute die Ergebnisse von COMA negativ beeinflusst haben. Diese Eigenschaften sind jedoch typisch für die Benennung der Attribute im Antikörpermarkt.

6.6.3 Bewertung der Wortähnlichkeiten

Die Kombination mehrerer Ähnlichkeitsmetriken als Feature-Vektor für ML-Verfahren funktionierte insbesondere in den Szenarien 1 und 2, besser als der einfache Referenzansatz und COMA. Daran zeigt sich, dass es effektiv sein kann ein ML-Verfahren die Gewichte der Metriken anhand von Beispielen lernen zu lassen, anstatt Heuristiken für die Gewichtungen zu verwenden. Wie die Auswertungen der einzelnen Szenarien zeigten, gibt es für ähnlichkeitsbasierte Lernansätze allerdings kein eindeutig zu favorisierendes ML-Verfahren. Im direkten Vergleich mit Wortvektoren können Ähnlickeitsmetriken nicht mithalten. Darüber hinaus ist die Bildung der Feature-Vektoren über Ähnlichkeitsmetriken komplizierter.

6.6.4 Mögliche Verwechslung von Attribut-Zuordnungen

Der für die Auswertung genutzte Goldstandard wurde aus einer realen Datenbank mit Zuordnungen für Eingangsattributnamen gebildet. Diese wird im Produktivbetrieb ständig erweitert und angepasst. Dadurch kann es zu Überschneidungen und Fehlern im Goldstandard gekommen sein, da mögliche spätere Korrekturen nicht in der Evaluation berücksichtigt wurden. In seltenen Fällen kann, je nach Zulieferer, gleiche Eingangsattributnamen für verschiedene Zielattributnamen genutzt werden. Diese Fälle können durch die hier vorgestellten Ansätze nicht gelöst werden, da dies eine $1 : M$ Beziehung darstellt. Diese wurden hier nicht betrachtet. Dadurch könnten einzelne Werte in der Auswertung minimal verfälscht sein.

6.7 Auswertung in der Praxis

Die betrachteten vier Szenarien zeigen, dass die Verfahren auf Wortvektor-Basis mit einem MLP-Klassifikator gut funktionieren können. Ob die Verfahren für eine vollständig automatisierte Verarbeitung geeignet sind oder die händische Zuordnung nur vereinfachen können, soll die folgende Auswertung zeigen.

Zur Evaluation in der Praxis wurden die trainierten Modelle aus den jeweiligen Szenarien verwendet und neue Eingangsattributnamen aus dem Produktivbetrieb angewandt. Diese wurden erst nach der Bildung des Goldstandards in die Zuordnungsdatenbank integriert. Hier muss beachtet werden, dass je nach Szenario eine unterschiedliche Anzahl von Zielattributen vorhergesagt wird. Aus diesem Grund wird in Abschnitt 6.7.3 durch die Betrachtung der häufigsten 26 Zielattribute eine bessere Vergleichbarkeit hergestellt.

6.7.1 Datensatz

Um praxisnahe Ergebnisse zu erzielen, wurden weitere Eingangsattributnamen und deren Zuordnungen aus dem produktiven Betrieb erhoben. Dieser Datensatz dient daher als weiterer Testdatensatz und besteht aus 204 Eingangsattributnamen von 50 Zielattributen (nur Zielattribute aus den zuvor trainierten Klassen). Davon entfallen 156 Eingangsattributnamen auf die 26 Zielattributnamen mit den meisten Zuordnungen aus Abbildung 5.5.

	P	R	F_1
Training aus Szenario 1			
OH mit MLP	0.81	**0.77**	0.77
TF-IDF mit MLP	**0.82**	**0.77**	**0.78**
Training aus Szenario 2			
OH mit MLP	0.79	0.72	0.74
TF-IDF mit MLP	0.79	0.74	0.75
Training aus Szenario 3			
OH mit MLP	0.76	0.72	0.73
TF-IDF mit MLP	0.76	0.72	0.73
Training aus Szenario 4			
OH mit MLP	0.77	0.73	0.74
TF-IDF mit MLP	0.77	0.76	0.76

Tabelle 6.9: Gewichtete Mittelwerte auf dem Testdatensatz

6.7.2 Ergebnisse der Klassifikation

In Tabelle 6.9 werden die Klassifikationsergebnisse der Wortvektor-Modelle, die im jeweiligen Szenario trainiert wurden, dargestellt. Die Modelle aus Szenario 1 und Szenario 2 können jeweils 26 Zielattribute vorhersagen. Die Modelle aus Szenario 3 können 86 Zielattributnamen als Zuordnung vorhersagen und 81 die Modelle aus Szenario 4. Dadurch bezieht sich die Bewertung der Modelle aus den ersten beiden Szenarien jeweils nur auf die 26 am häufigsten vertretenen Klassen aus dem Trainingsdatensatz, die auch im Testdatensatz auftreten. Die Ergebnisse der übrigen Modelle beziehen sich auf den gesamten Testdatensatz.

Bei Betrachtung der Ergebnisse schneiden Modelle, die auf dem Datensatz aus Szenario 1 trainiert wurden besser ab, als Modelle, die mit anderen Datensätzen trainiert wurden. Dagegen besitzt die Art der Vektortransformation nur einen geringfügigen Einfluss. Entscheidend sind die Daten, auf denen das Modell trainiert wurde.

Dies wird insbesondere im direkten Vergleich zwischen den Modellen aus Szenario 1 und Szenario 2 deutlich. Modelle aus Szenario 1 erzielen je nach betrachteter Metrik Verbesserungen um 0.02 bis zu 0.05, wobei das Modell aus Szenario 1 mit TF-IDF insgesamt am besten abschneidet.

Im direkten Vergleich von Modellen aus Szenario 3 und Szenario 4 erzielen die Modelle aus Szenario 4 geringfügig bessere Ergebnisse als die Modelle aus Szenario 3, wobei die TF-IDF aus Szenario 4 die besten Werte erzielt.

Die Ergebnisse aus der Praxis bestätigen die vorherigen Ergebnisse, wobei die Modelle in der Praxis teilweise sogar besser abschneiden als durch die Überkreuzvalidierung zu erwarten war. Das gilt insbesondere für die Ergebnisse der Modelle

	nur top 26	alle
Training aus Szenario 1		
OH mit MLP	0.79	–
TF-IDF mit MLP	0.81	–
Training aus Szenario 2		
OH mit MLP	0.84	–
TF-IDF mit MLP	0.84	–
Training aus Szenario 3		
OH mit MLP	0.87	0.84
TF-IDF mit MLP	0.88	0.84
Training aus Szenario 4		
OH mit MLP	0.90	0.86
TF-IDF mit MLP	0.88	0.85

Tabelle 6.10: Top-3-Genauigkeit der Verfahren aus verschiedenen Szenarien auf dem Testdatensatz.

aus Szenario 1, die in der Praxis deutlich besser abschneiden als in der Evaluation. Das Modell mit TF-IDF-Vektorisierung erreicht in der Praxis beispielsweise Ergebnissteigerungen um über 0.1 (Precision 0.82 zu 0.68, Recall 0.78 zu 0.66, F_1-Score 0.77 zu 0.65).

Trotz dieser erheblich besseren Ergebnisse reichen die erzielten Werte nicht aus, um die Zuordnung vollständig in eine Dunkelverarbeitung zu überführen. Daher werden im nächsten Abschnitt die Ergebnisse von Empfehlungen betrachtet, die Mitarbeitern das manuelle Zuordnen vereinfachen soll.

6.7.3 Ergebnisse als Empfehlungsgeber

Wenn keine vollständige Dunkelverarbeitung möglich ist, können Empfehlungen Mitarbeitern helfen, ihren Arbeitsablauf zu vereinfachen. Dazu sollte ein Empfehlungssystem Vorschläge für die richtige Zuordnung ermitteln. Damit ein Mehrwert für Mitarbeiter entsteht, dürfen allerdings nicht alle Attribute vorgeschlagen werden. Aus diesem Grund wurde die Anzahl der möglichen Vorschläge auf die drei Zuordnungen mit der höchsten Konfidenz beschränkt.

Die Top-3-Genauigkeit gibt an, wie hoch der Anteil von Empfehlungen ist, die die korrekte Zuordnung enthalten. In Tabelle 6.10 werden die Ergebnisse der Top-3-Genauigkeit der Verfahren aus dem vorherigen Abschnitt dargestellt. Hierbei wurde zusätzlich zur besseren Vergleichbarkeit unterschieden, ob ein Verfahren nur die 26 häufigsten Attributnamen unterscheiden kann, oder alle im Testdatensatz verfügbaren Attributnamen (50).

Im Folgenden werden zuerst die Ergebnisse der Eingangsattributnamen für die 26 häufigsten Zielattributnamen betrachtet. Dabei fällt auf, dass zwar die Ergebnisse in der Klassifikation (vgl. Tabelle 6.9) der Szenarien ohne Datensatzerweiterung (Szenario 1 und Szenario 2) besser waren, allerdings in der Top-3-Genauigkeit hinter den Szenarien, die auf erweiterten Datensätzen trainiert wurden, zurückliegen.

Des Weiteren ist festzuhalten, dass eine Vorverarbeitung der Daten für die Top-3-Genauigkeit deutliche Verbesserungen erkennen lässt. So liegt in jedem Datenvorbereitungsschritt eine Verbesserung der Top-3-Genauigkeit: Von Szenario 1, ohne Bearbeitung, zu Szenario 2 (Kleinschreibung und Worttrennung) um 0.03 bis 0.05. Mit Datenerweiterung steigt die Top-3-Genauigkeit von Szenario 1 zu Szenario 3 sogar um 0.07 bis 0.08. Wird die Datenerweiterung mit Kleinschreibung und Worttrennung kombiniert kann eine weitere Steigerung auf bis zu 0.90 erreicht werden. Dies bedeutet die richtige Empfehlung der Zuordnung in bis zu 90 % der Fälle.

Zusätzlich können die Modelle, die auf erweiterten Datensätzen trainiert wurden, mehr Zielattribute erlernen. Bei Auswertung der Eingangsattributnamen für alle 50 Zielattribute aus dem Testdatensatz fällt die Top-3-Genauigkeit im Vergleich zur Betrachtung der Eingangsattributnamen für die 26 häufigsten Zielattributnamen leicht ab. Eine mögliche Ursache ist, dass die Modelle auf die angelernten Zielattribute mit wenigen Eingangsattributnamen auf diese Eingangsattributnamen überangepasst sind. Eine andere Möglichkeit ist, dass ein Eingangsattributname nicht gelernt werden konnte, da er im Vorhinein nicht existierte.

Insgesamt zeigen die Ergebnisse die Effektivität der Datenaufbereitung und Datenerweiterung für praxisorientierte Anwendungen wie Empfehlungssysteme.

6.7.4 Auswirkungen auf die manuelle Arbeit

Um zu überprüfen, ob die Empfehlungen die Mitarbeiter bei der manuellen Katalogintegration tatsächlich unterstützen, wurde im Rahmen der Forschungskooperation ein MLP mit OH-Vektorisierung über einen representational state transfer (REST)-Endpunkt ausgeliefert. Dadurch konnten die Empfehlungen des Modells einfach in die Datenintegrationsplattform von antibodies-online eingebracht werden.

Bei einer anschließenden informellen Mitarbeiterbefragung wurden die Erfahrungen mit den Empfehlungen durchweg als positiv bezeichnet. So wurde eine Minderung des Zeitaufwands von mehreren Minuten auf wenige Sekunden festgestellt, wenn die korrekte Zuordnung unter den ersten drei Empfehlungen war. Bei einer

falschen Empfehlung wurden keine negativen Einflüsse auf den Integrationsprozess berichtet.

Dabei treten falsche Empfehlungen unter anderem dann auf, wenn auch für Menschen ohne Betrachtung des Inhalts keine Aussage über die Zuordnung getroffen werden kann. So musste z. B. `Expr24` dem Zielattribut `Application_Advice` zugeordnet werden. Fälschlicherweise wurde stattdessen `Supplier_Product_ID` als Zielattribut mit der höchsten Konfidenz bestimmt. In diesem Beispiel kann ein Mensch ohne Hintergrundwissen über den Zulieferer oder den Inhalt der Attributinstanzen allerdings auch keine qualitativ bessere Zuordnung vornehmen. Ein anderes Beispiel für eine nicht erkannte Zuordnung ist `Biological_Origin`. Hier sollte eine Zuordnung zu `Hosts` erkannt werden. Durch die starke Abweichung und fehlende Überschneidungen mit bekannten Eingangsattributnamen konnte die korrekte Zuordnung nicht erfolgen.

6.7.5 Einschränkungen in der Praxisauswertung

Bei der Auswertung in der Praxis ist nicht auszuschließen, dass im Zeitraum der Evaluation das ML-System zufällig genutzt wurde, um hauptsächlich gut gelernte Attributnamen zuzuordnen. Dadurch könnte die Bewertung zu positiv ausgefallen sein.

Die genaue wirtschaftliche Bedeutung kann nicht beziffert werden, da die Bearbeitungszeiten der Mitarbeiter weder vor, noch nach dem Einbau des Empfehlungssystems erhoben wurden. Die Angaben beruhen auf Interviews mit einzelnen Personen und spiegeln daher eher die subjektive Einschätzung der Befragten wider.

Kapitel 7

Attribut Label Ranking

Im vorherigen Kapitel wurden Zuordnungen lediglich über Attributnamen gelernt. Ausschlaggebend waren je nach Lernverfahren Wortähnlichkeiten, N-Gramme oder Wortvektoren. Diese Ansätze nutzen allerdings einen Großteil der zur Verfügung stehenden Informationen nicht, nämlich die Attributinstanzen bzw. den konkreten Spalteninhalt. Dadurch können $1:1$ oder $N:1$ Beziehungen vorhergesagt werden, nicht aber $1:M$ Beziehungen. Entsprechend wurde bei der bisherigen Modellierung nur von genau einer Zielklasse ausgegangen.

In diesem Kapitel wird mit dem Attribut Label Ranking (ALR) ein Verfahren entwickelt, das neben $1:1$ und $N:1$ Beziehungen auch $1:M$ Beziehungen erkennen kann. ALR erlaubt die Zuordnung einer Spalte C_i^{in} aus dem Eingabekatalog zu mehreren Spaltennamen L_k^{out} aus dem Zielschema.

Die in den folgenden Abschnitten vorgestellten Vorgehensweisen, Modellierungen und Evaluationen wurden bereits größtenteils in Schmidts et al. [S5, S7] veröffentlicht.

7.1 Aufbau

Mit dem Attribut Label Ranking wurde im Rahmen des Forschungskooperationsprojekts ein zweistufiges, spaltenorientiertes Verfahren zur Katalogintegration entwickelt, das neben Attributnamen auch Attributinstanzen zur Vorhersage der Zuordnungen nutzt. Wie in Abbildung 7.1 dargestellt, werden die Zuordnungen mit ALR nicht für einzelne Attributnamen oder Attributinstanzen vorhergesagt, sondern für eine gesamte Spalte aus dem Eingangskatalog. Die Voraussetzungen für die Anwendung von ALR sind, dass das Zielschema eine feinere Granularität als das Eingangsschema besitzt, mindestens aber die gleiche. Zusätzlich wird angenommen, dass

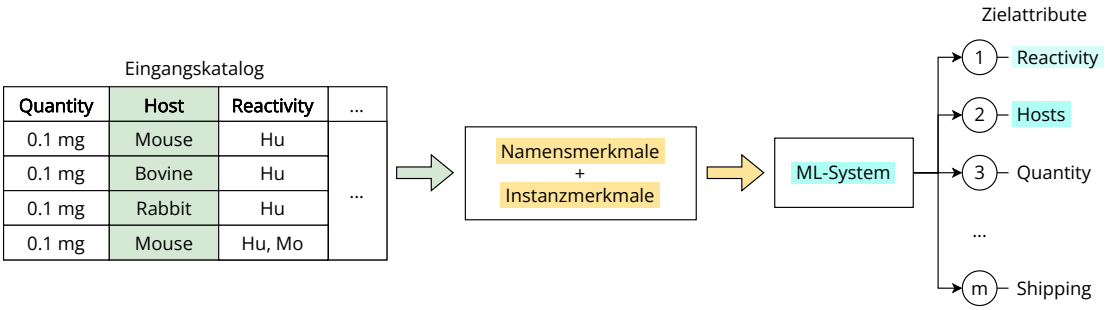


Abbildung 7.1: Grundsätzlicher Ablauf des Attribut Label Ranking

der Produktkatalog tabellarisch ist und die Produktdomäne des Eingangskatalogs zum Zielschema passt. Enthält ein Eingangskatalog beispielsweise Antikörper, sollte das Zielschema Attributnamen enthalten, die zu Antikörperprodukten passen.

ALR besteht aus zwei aufeinanderfolgenden Schritten. Im ersten Schritt werden Zuordnungen anhand von Attributnamen und -instanzen vorhergesagt (siehe Abbildung 7.2), während der zweite Schritt die Vorhersagen aggregiert und gewichtet (vgl. Abbildung 7.3). Für die Vorhersage wird jede Attributinstanz und der zugehörige Eingangsattributname jeweils in einen eigenen Feature-Vektor überführt. Die Vektoren für Attributinstanz und -namen werden unabängig voneinander gebildet. Sie dienen als Eingangswerte für ihre jeweiligen NN, bevor die einzelnen NN über ein nachgelagertes NN verbunden werden. Das kombinierte NN trifft letztendlich die Zuordnung. Das Vorgehen ist beispielhaft in Abbildung 7.2 dargestellt. Hier werdenm aus dem Paar `Host: Rabbit` die Feature-Vektoren extrahiert, bevor sie als Eingangswerte für das NN genutzt werden. Grundsätzlich können beliebige Verfahren aus Abschnitt 5.2 für Attributnamen genutzt werden, während für Attributinstanzen hauptsächlich Wortvektoren und Metadaten, wie die Anzahl der Buchstaben oder Sonderzeichen, genutzt werden können (siehe Abschnitt 7.2).

Dadurch durchlaufen sowohl Namen als auch Instanzen zunächst eigene NN, damit jedes NN eigene Gewichtungen für die Features lernen kann, bevor sie über eine Verbindungsebene wieder miteinander verbunden werden. Danach folgt ein weiteres Subnetzwerk bevor die Ergebnisebene mit der Vorhersage folgt. In Abbildung 7.2 werden dem Paar `Host: Rabbit` durch das NN die beiden Zielattribute $(1 : M)$ `Reactivity` und `Hosts` zugeordnet.

Im beschriebenen Vorgehen sind zwei Besonderheiten zu beachten:

1. Es werden alle möglichen Zielklassen (Attributnamen des Zielschemas) vorhergesagt, wodurch die Klassifikation mehrere Zielklassen gleichzeitig vorhersa-

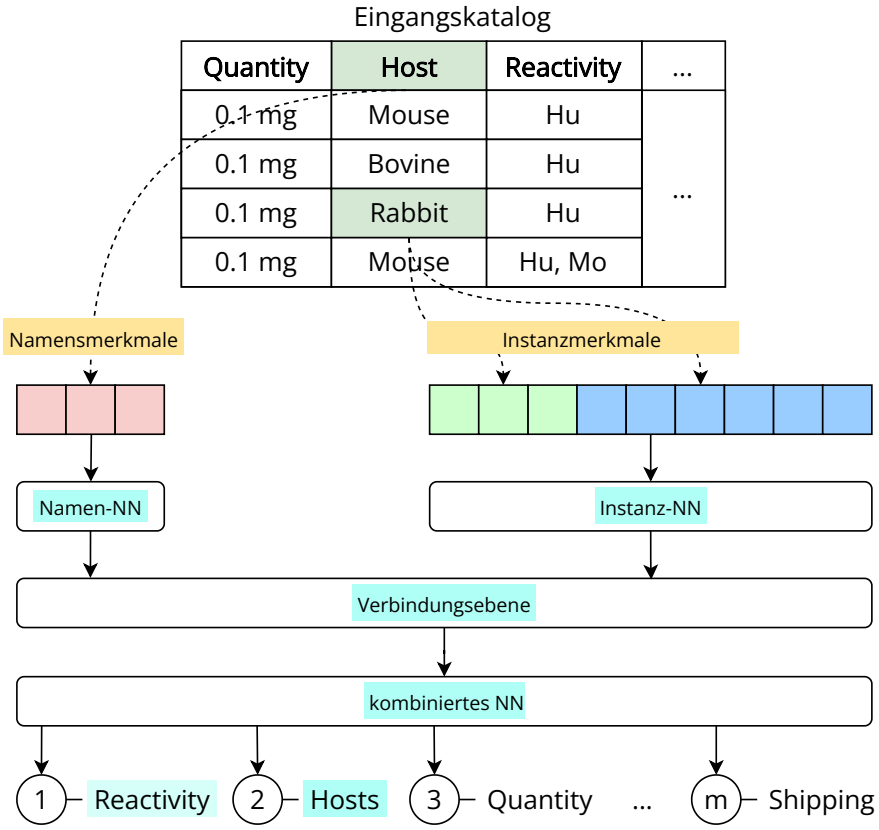


Abbildung 7.2: Grundsätzlicher Ablauf der Vorhersage von Zuordnungen bei einem Klassifikationsansatz anhand von Attributnamen

gen kann. Die Vorhersage jeder *einzelnen* Zielklasse liegt unabhängig von den anderen Zielklassen im Wertebereich $[0, 1]$, wodurch die Summe der Konfidenzen im Intervall $[0, M]$ liegt, wobei M der Anzahl aller möglichen Zielklassen entspricht. Bei einer exklusiven Klassifikation ergibt dagegen die Summe aller Konfidenzen immer genau eins.

2. Das Nutzen der Eingangswerte aus Attributinstanzen und -namen für unabhängige Subnetzwerke, die später miteinander verbunden werden, um eine gemeinsame Vorhersage durchzuführen. Ein solches NN wird auch als multimodales NN bezeichnet. Der Vorteil eines solchen Netzwerks ist, dass unterschiedliche Eigenschaften zunächst unabhängig voneinander betrachtet werden. Die verborgenen Abhängigkeiten werden erst später im Training gelernt.

Nachdem alle Zielklassen für alle Paare einer Spalte durch das NN vorhergesagt wurden, werden die Vorhersagen aggregiert. In Abbildung 7.3 wird beispielsweise einfaches Zählen als Aggregationsfunktion verwendet. Bei dieser Aggregation wird ein Zähler pro Zielattribut erhöht, wenn das NN dieses Attribut mit einem Wert größer einem Schwellwert vorhersagt. Über den Schwellwert können Vorhersagen mit niedriger Konfidenz ausgeschlossen werden. Basierend auf dieser Anzahl wird eine Rangfolge (Ranking) aus allen vorhergesagten Zielattributen einer Spalte erstellt. Die letztendliche Vorhersage des ML-Systems ist dann die relative Häufigkeit aller vorhergesagten Zielattribute einer Spalte.

Bezogen auf Abbildung 7.3 bedeutet dieses Vorgehen, dass für die Instanzen `Mouse`, `Bovine`, `Rabbit` und `Mouse` mit ihrem zugehörigen Attributnamen `Hosts` zunächst Feature-Vektoren gebildet werden, aus denen das zuvor trainierte NN aus dem ersten Schritt Vorhersagen trifft. Jede Vorhersage mit einer Konfidenz größer null wird gezählt. Da das Netzwerk `Hosts` als mögliche Bezeichnung für drei Vertreter (`Mouse`, `Rabbit`, `Mouse`) vorhersagt, erhält `Hosts` den höchsten Rang unter allen Bezeichnungen vor `Reactivity` und `ReactivityComment`. Folglich ist die resultierende Konfidenz des Gesamtsystems $\frac{3}{8} = 0.375$ für die primäre Zuordnung das Zielattribut `Hostss`.

Durch das Ranking nutzt ALR ein ähnliches Verfahren wie die Ensemble-Methode des Votings (vgl. Abschnitt 3.5.4). In diesem Fall werden jedoch nicht die Vorhersagen mehrerer ML-Modelle kombiniert, sondern die Vorhersagen bzgl. mehrerer Instanzen einer Spalte. Dahinter steht die Idee, dass das die primäre Zuordnung einer Spalte häufiger vorhergesagt wird als andere Zuordnungen. Darüber hinaus werden mithilfe des Rankings weitere Zuordnungen ermittelt, sodass auch $1 : M$ bzw. $N : M$ Beziehungen aufgedeckt werden können.

7.2 Bildung von Instanzvektoren

Um Attributinstanzen als Eingangswerte für NNs nutzen zu können, müssen sie ebenfalls in Feature-Vektoren transformiert werden. Dazu müssen u.a. Verfahren zur Wort- bzw. Textvektorisierung verwendet werden, da Attributinstanzen oftmals unstrukturierten Text enthalten. In den folgenden Abschnitten werden mehrere Möglichkeiten aufgezeigt, Feature-Vektoren aus Metadaten und Texten der Attributinstanzen zu gewinnen. Auf die Vektorisierung von Attributnamen wird nicht weiter eingegangen, da Abschnitt 5.2 bereits alle relevanten Verfahren abdeckt.

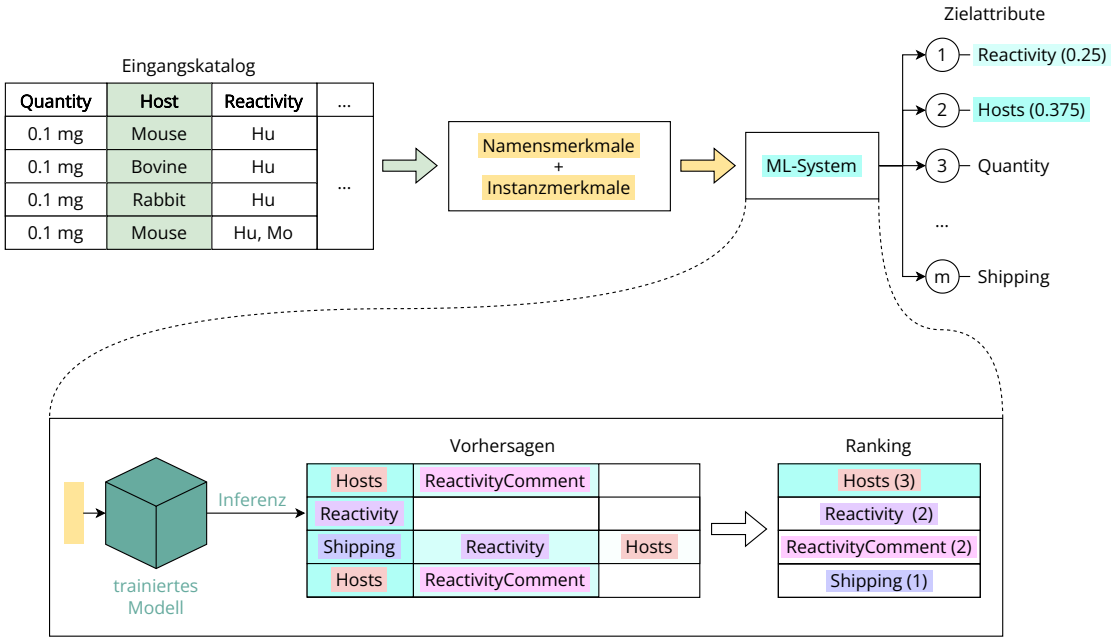


Abbildung 7.3: Grundsätzlicher Ablauf der Vorhersage von Zuordnungen bei einem Klassifikationsansatz mit ALR

7.2.1 Lexikalische Metadaten aus Attributinstanzen

Aus den einzelnen Zeichen einer Attributinstanz lassen sich Merkmale für einen Feature-Vektor ableiten. Chen et al. [14] haben bereits eine Reihe von Zeicheneigenschaften untersucht, die sich für die Vorhersage von Spaltennamen eignen, insbesondere wenn der Name einer Spalte des Typs IDs bestimmt werden soll. Da auch Attributinstanzen häufig den Typ ID enthalten und gleichzeitig einige Attributinstanzen, beispielsweise für einen Antikörpernamen oder eine Gensequenz, wie IDs aus häufig aus Folgen von Ziffern und Buchstaben bestehen, lassen sich diese lexikalischen Metadaten auch auf Produktdaten anwenden.

Tabelle 7.1 bietet eine Übersicht geeigneter Merkmale auf Zeichenbasis. Zu den von Chen et al. übernommenen Metadaten gehören die Anteile alphabetischer, numerischer und symbolischer Zeichen. Zusätzlich lässt sich die Textlänge als Merkmal nutzen. Sie entspricht der Anzahl Zeichen in einer Attributinstanz. Allerdings muss diese Länge für die Verwendung in einem NN im Wertbereich $[0, 1]$ liegen, sodass die Länge auf 1024 Zeichen begrenzt und der Wert anschließend normalisiert wird.

Beschreibung	Länge
Normalisierte Textlänge, begrenzt auf 1024 Zeichen	1
Anteil alphabetischer Zeichen	1
Anteil numerischer Zeichen	1
Anteil symbolischer Zeichen	1
Text enthält eine URL	1

Tabelle 7.1: Liste der Merkmale aus den Metadaten der Attributinstanzen

Da die Produktdaten häufig verschiedene URLs enthalten, die unter Umständen verschiedenen Zielattributnamen zugeordnet werden müssen, wird für ALR ein weiteres binäres Merkmal gewählt: Wenn der Text einer Instanz eine URL enthält, wird der Wert auf eins gesetzt. Andernfalls auf null. Dieses Merkmal kann anhand eines regulären Ausdrucks zuverlässig bestimmt werden. Allerdings muss die Zuordnung letztendlich anhand anderer, inhaltsbasierter Vektorisierungen erfolgen, da dieses Merkmal alleine nicht ausreicht, eine Zuordnung abzuleiten.

Diese Merkmale werden, wie beschrieben, nur aus Attributinstanzen und nicht aus Attributnamen extrahiert, da Attributnamen in den meisten Fällen aus Buchstaben von A bis Z bestehen. Zwar existieren auch Attributnamen, die eine Einheit mit Sonderzeichen enthalten (beispielsweise *Storage* °*C*) oder Attributnamen die Zahlen enthalten. Allerdings werden diese Sonderfälle bereits durch andere Verfahren, wie die OH-Kodierung, ausreichend abgebildet.

7.2.2 Textvektorisierung

Während die Zielattributnamen bei der Katalogintegration im Vorhinein bekannt sind, ist nicht bekannt, wie Attributinstanzen während des ETL-Prozesses transformiert wurden bzw. werden müssen. Des Weiteren könnten zwar Ähnlichkeiten zwischen Attributinstanzen aus dem Eingangskatalog und Attributinstanzen aus dem Marktplatzkatalog gebildet werden. Jedoch würde dieses Vorgehen zur Laufzeit zu Problemen führen, da im Marktplatz häufig mehrere Millionen Produkte enthalten sind, die alle mit einer neuen Instanz verglichen werden müssten. In dem gewählten Klassifikationsansatz ist dieses Vorgehen daher nicht praktikabel.

Besser geeignet für Instanzen, die Wertelisten oder unstrukturierten Text enthalten, sind Verfahren zur Textvektorisierung, die bereits bei anderen Aufgaben aus dem Bereich des natural language processing (NLP) eingesetzt werden.

OH und TF-IDF Zu den einfacheren Verfahren zählen hier die OH-Kodierung oder TF-IDF. Die Anwendung dieser Verfahren auf Attributinstanzen gleicht der Anwendung auf Attributnamen. Über eine gewählte Tokenisierung wird das Vokabular gebildet und im Anschluss erfolgt die Vektorbildung über das gewählte Verfahren.

Bereits vorgestellte Probleme, wie die Größe des Vokabulars als Länge des Feature-Vektors bleiben bestehen. Bei der Anwendung auf Attributinstanzen können so Vokabulargrößen von mehreren Tausend Token entstehen, je nachdem welche Form der Tokenisierung genutzt wird und wie beispielsweise N-Gramme kombiniert werden. Dadurch wird auch der Feature-Vektor im Vergleich zu dem für Attributnamen erheblich länger und muss in der Praxis sogar begrenzt werden, um Rechenzeit und Speicherplatz zu sparen. Trotz des größeren Vokabulars bleibt das Risiko für OOV-Fehler bestehen.

FastText Eine Alternative zu den genannten Verfahren bietet, wie bei Attributnamen, FastText als vortrainiertes Sprachmodell. Im Gegensatz zur Verwendung bei Attributnamen, bieten Instanzen genug Datenpunkte, um das FastText Modell auf die Domäne des Produktkatalogs zu spezialisieren. Dadurch können ggf. bessere Ergebnisse erzielt werden.

Die Berechnung der Vektoren für eine Attributinstanz erfolgt bei FastText über das arithmetische Mittel über alle normalisierten Tokenvektoren. Dabei erfolgt Normalisierung über die euklidische Norm und die Tokenisierung anhand von Leerzeichen oder Zeilenumbrüchen [78]. Hier ergibt sich insbesondere bei längeren Attributinstanzen das Problem, das ein Instanzvektor unscharf wird. Allerdings ist dies auch bei den anderen Verfahren möglich. Der große Vorteil von FastText, die konstante Feature-Vektor-Länge, bleibt jedoch erhalten.

Modernere Verfahren wie BERT [79] und ELMO [80] oder darauf aufbauende Verfahren werden hier nicht berücksichtigt. Durch die spezielle Sprache in Produktdaten, wie unvollständige Sätze, Aufzählungen, o.ä., müssten sie komplett neu trainiert werden. Diese Verfahren versuchen die Bedeutung eines Wortes anhand des Kontexts zu ermitteln. Der Kontext eines Produktattributs entsteht allerdings häufig erst aus der Kombination von Inhalt und Namen anstatt aus vorangegangenen Worten oder Sätzen. Dadurch ist es für diese Verfahren deutlich schwieriger, die Bedeutung der Worte abzuleiten.

7.3 Datengrundlage

Um ALR auf die Katalogintegration anzuwenden, müssen geeignete Trainingsdaten zur Verfügung stehen. Es gibt verschiedene Möglichkeiten, einen Trainingsdatensatz zu erstellen. Die erste Möglichkeit ist die manuelle Aufbereitung der Daten und das manuelle Nachhalten von Zuordnungen (Goldstandard). Dabei müsste genau festgehalten werden, welche Information aus dem Eingangskatalog welchem Zielattribut zugeordnet werden muss. Für ALR müsste folglich jeder Attributinstanz das entsprechende Zielattribut zugeordnet werden und ggf. zusätzliche Informationen, beispielsweise welche Transformationsschritte in dem ETL-Prozess angewandt wurden, bereitgestellt werden. Dieser Prozess ähnelt der eigentlichen Katalogintegration und beinhaltet einen übermäßigen manuellen Aufwand, der insbesondere für KMU nicht zu leisten ist.

Daher wird in diesem Abschnitt ein automatisierter Ansatz zur Generierung eines Silberstandards vorgestellt, der auf der Analyse von Produktkatalogen vor und nach dem Durchlauf des ETL-Prozesses beruht. Dadurch werden zwar die Zuordnungen im Vergleich zum Goldstandard ungenauer ermittelt, allerdings können gleichzeitig größere Datenmengen, die für ML-basierte Ansätze notwendig sind, genutzt werden. In den nachfolgenden Abschnitten wird zunächst die Datenerhebung beschrieben und eine Übersicht auf den erhobenen Datensatz gegeben, bevor das Verfahren zur automatisierten Erstellung des Trainingsdatensatzes vorgestellt wird. Des Weiteren wird eine Möglichkeit zur automatischen Datenerweiterung gegeben, die die Menge der Trainingsdaten weiter erhöhen kann. In der späteren Evaluation in Kapitel 8 werden die Ergebnisse der Datenerweiterung getrennt betrachtet, um deren Wirksamkeit einzeln zu beurteilen.

7.3.1 Übersicht und Datenerhebung

Für den automatisierten Ansatz müssen die Daten der Produktkataloge an zwei Stellen des ETL-Prozesses gesammelt werden. Wie in Abbildung 7.4 zu sehen ist, werden als Erstes die unveränderten Produktkataloge der Zulieferer gesammelt. Nachdem die Produktkataloge durch den ETL-Übersetzungsprozess (siehe auch Abbildung 1.3 auf Seite 8) in einen marktplatzkonformen Produktkatalog überführt wurden, können die marktplatzkonformen Kataloge erneut gesammelt werden. An dieser Stelle besitzt jeder Katalog das Zielschema. An beiden Stellen werden die Kataloge in einen Datensatz überführt, der somit die Produktkataloge vor und nach Durchlaufen des ETL-Prozesses beinhaltet.

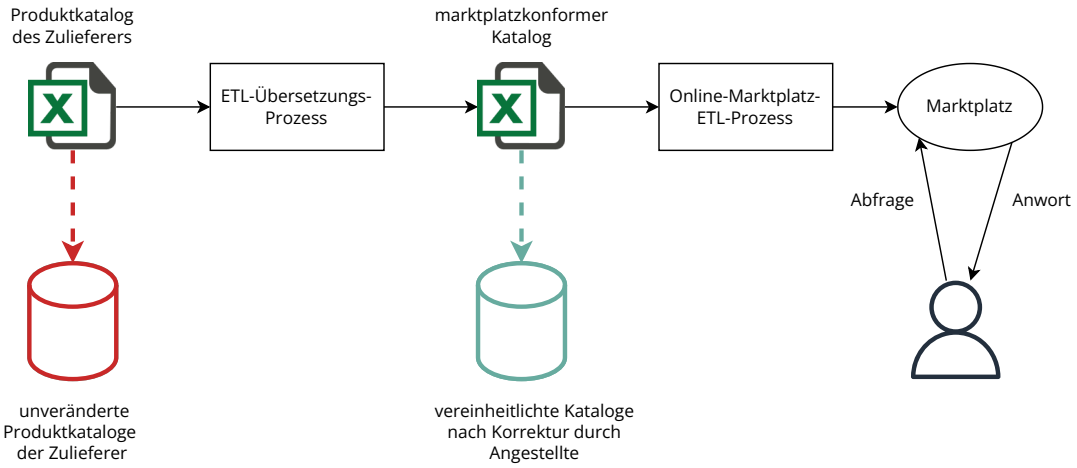


Abbildung 7.4: Datenerhebung im laufenden Integrationsprozess

Gesamtzahl der Produktkataloge	19
Gesamtzahl der enthaltenen Produkte	ca. 420.000
Produkte pro Katalog	3 – ca. 230.000
Zulieferer	12
Anzahl der nicht leeren Produktattribute pro Zulieferer	14 – 65

Tabelle 7.2: Eigenschaften der Eingangskataloge aus dem erhobenen Datensatz (vgl. Schmidts et al. [S5])

Der Vorteil dieses Vorgehens ist, dass die Datenerhebung im laufenden Produktivbetrieb eines KMU erfolgen kann, ohne diesen zu beeinträchtigen. Zusätzlich werden im ETL-Prozess bereits alle menschlichen Interaktionen durchgeführt, sodass davon auszugehen ist, dass bereits alle erforderlichen Zuordnungen und Transformationen durchgeführt wurden, damit die Kataloge für die weitere Verarbeitung im Online-Marktplatz bereitstehen.

Über dieses Verfahren wurden 19 Produktkataloge vor und nach Durchlaufen des ETL-Prozesses in einen Datensatz überführt, der ca. 420.000 Produkte enthält. Wie Tabelle 7.2 zu entnehmen ist, reicht die Anzahl der Produkte pro Katalog dabei von drei bis 230.000 Produkten von zwölf unterschiedlichen Zulieferern, die zur Beschreibung ihrer Kataloge 14 bis 65 Attribute nutzen. Die Eingangskataloge selbst enthalten zunächst mehr Attribute, allerdings ohne Inhalt. Es ist davon auszugehen,

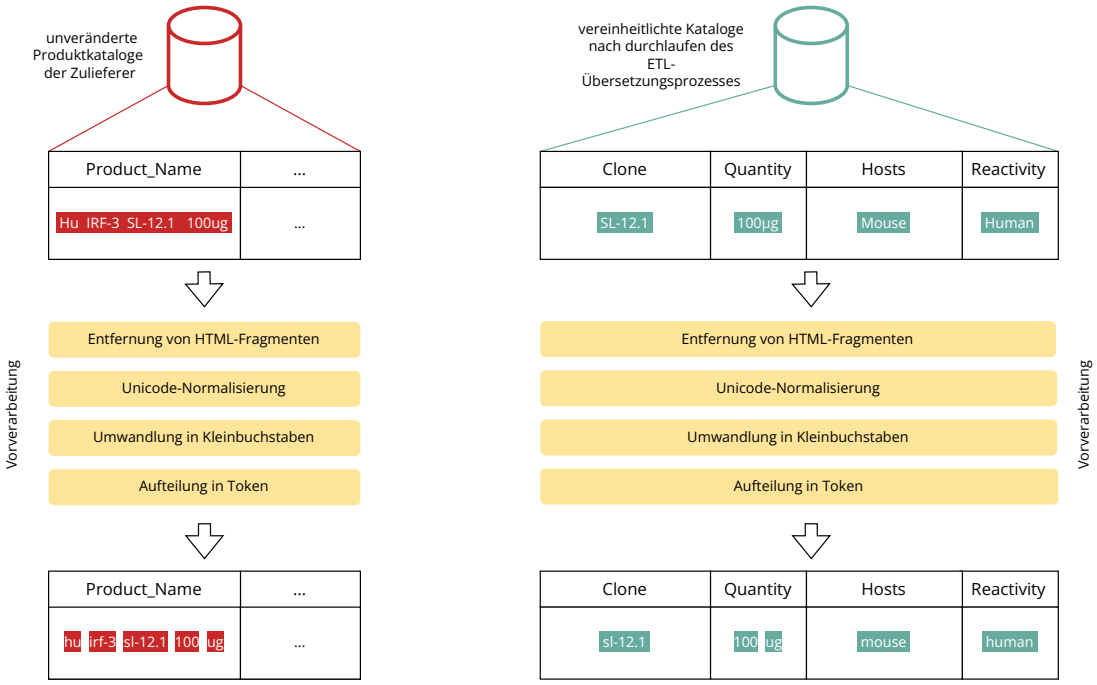


Abbildung 7.5: Vorverarbeitung der jeweiligen Eingangs- und Zielkataloge

dass diese Attribute nicht zur weiteren Beschreibung genutzt werden. Aus diesem Grund werden die Attribute ohne Instanz in der Tabelle nicht weiter berücksichtigt.

7.3.2 Erstellung eines Silberstandards

Nach der Datenerhebung müssen die Zuordnungen der Attributinstanzen ermittelt werden, damit aus den Rohdaten geeignete Datensätze für ALR erstellt werden können. Um aus den Rohdaten den Silberstandard zu erhalten, müssen die Daten der Attributinstanzen so aufbereitet werden, dass eine einfache Zuordnung zwischen dem unveränderten Eingangskatalog und dem marktplatzkonformen Katalog möglich ist, da diese Zuordnungen zum Zeitpunkt der Datenerhebung nicht bekannt sind.

Dieser Vorgang ist in Abbildung 7.5 dargestellt. Zur Aufbereitung der Attributinstanzen werden für Eingangs- und Zielkatalog die gleichen Schritte gewählt, damit das Ergebnis der Vorverarbeitung möglichst einheitlich ist. Die Vorverarbeitung besteht aus vier wesentlichen Schritten:

1. Entfernung von HTML-Fragmenten wie beispielsweise `<br>` oder `<ul>/<li>` (HTML-Listenelemente),

2. Unicode-Normalisierung,
3. Aufteilung in Token,
4. Umwandlung in Kleinbuchstaben.

Die Unicode-Normalisierung wandelt einzelne Buchstaben in ihr ASCII-Äquivalent um. Dadurch wird beispielsweise das μ aus der Attributinstanz 100μ in ein einfaches u umgewandelt. Auch Probleme im Umgang mit unterschiedlicher Zeichenkodierung und problematische Ergebnisse von optischer Zeichenerkennung (optical character recognition (OCR)), die Zeichen zwar nach menschlichen Gesichtspunkten korrekt identifizieren, aber ein anderes Unicode-Zeichen nutzen, können so ausgeglichen werden (vgl. Abschnitt 2.5).

Bei der anschließenden Aufteilung der Attributinstanzen in Token können entweder sehr einfache Verfahren, wie eine Trennung an Leerzeichen, genutzt werden oder hoch spezialisierte. Um hoch spezialisierte Verfahren nutzen zu können, müssen die Produktdaten und das Tokenisierungsverfahren allerdings aus einer ähnlichen Domäne kommen, damit die hinterlegten Regeln zur Tokenisierung sich auf die Produktdaten anwenden lassen. Im Falle von Antikörperdaten kann beispielsweise die Tokenisierung mittels sciSpacy[1] von Neumann et al. [81] genutzt werden.

Diese wurde speziell auf biomedizinische Texte angepasst, sodass die Tokenisierung nach zusätzlichen Regeln erfolgt. Dadurch können unter anderem Zahlen mit zugehörigen Einheiten in entsprechende Token aufgeteilt werden, auch wenn sie nicht durch Leerzeichen getrennt wurden. Gleichzeitig erlaubt sciSpacy die Auflösung verschiedener medizinischer Abkürzungen, die ebenfalls in der Vorverarbeitung genutzt wird. Die abschließende Umwandlung der Token in Kleinbuchstaben dient lediglich der Vereinheitlichung, damit in den marktplatzkonformen Katalog übernommene Token möglichst gut identifiziert werden können.

Nachdem Eingangs- und Zielkatalog vorbereitet wurden, kann die Bestimmung der Zuordnungen erfolgen. Dabei ist zu beachten, dass im marktplatzkonformen Zielkatalog nicht bekannt ist, welche Schritte der ETL-Pipeline zu dem Ergebnis geführt haben. Dazu könnte zum Beispiel die Anwendung von regulären Ausdrücken gehören, um Informationen in Zeichenketten in eine oder mehrere Zielattribute aufzuteilen, oder das Auflösen von Abkürzungen sowie die Standardisierung von Maßeinheiten. Wurde eine Attributinstanz aus dem Eingangskatalog unverändert in ein Zielattribut übernommen, ist von einer $1:1$ Beziehung zwischen den Attributen auszugehen.

[1] https://github.com/allenai/scispacy

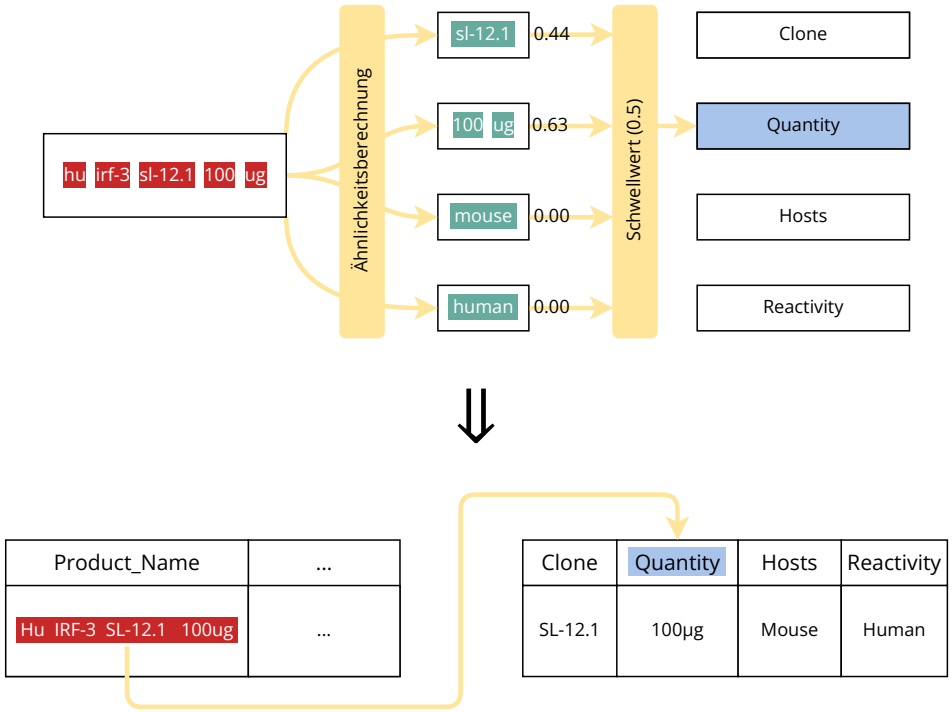


Abbildung 7.6: Datenerhebung im laufenden Integrationsprozess

Werden Produktattribute beispielsweise über reguläre Ausdrücke übernommen, finden sich die Attributinstanzen jedoch nicht unverändert im Zielkatalog wieder. Somit können komplexere Beziehungen oder leicht veränderte Informationen nur erkannt werden, wenn ein weicheres Kriterium als die vollständige Übernahme der Attributinstanz gewählt wird. Hier bieten sich Ähnlichkeitsmetriken, wie die Kosinus-Ähnlichkeit auf Tokenbasis an. So kann über einen Schwellwert definiert werden, wie hoch die minimale Ähnlichkeit ist, bevor eine Zuordnung gewertet wird.

Am Beispiel von Abbildung 7.6 lässt sich dieses Vorgehen nachvollziehen. Dabei wurde der Produktname (rot, oben) bereits vollständig vorbereitet und liegt in einzelnen kleingeschriebenen Token vor. Gleiches gilt für sämtliche weitere Attributinstanzen des Produkts aus dem Zielschema. Über die Kosinus-Ähnlichkeit der Token ergibt sich eine Ähnlichkeit von 0.44 mit `sl-12.1` und 0.63 mit `100 ug`, während für `mouse` und `human` keine Ähnlichkeit mit dem Produktnamen ermittelt wurde. Im Silberstandard wird anschließend die Zuordnung des unbearbeiteten Produktnamens (rot, unten) zum Zielattribut hinterlegt. Listing 7.1 zeigt den so entstehenden Silberstandard als Datensatz im JavaScript Object Notation (JSON)-Format. Die unveränderte Attributinstanz wird im Feld `content` gespeichert und alle Ähnlich-

```
{
  'content': 'Hu IRF-3 SL-12.1 100ug'
  'labels': [
    'Clone': 0.44,
    'Quantity': 0.63,
    ...
    'Reactity': 0.0
  ]
}
```

Listing 7.1: Ausschnitt der Zielstruktur im JSON-Format aus Schmidts et al. [S5]

keiten im Unterpunkt `labels`, wobei jeweils der Attributname als Schlüssel für den Ähnlichkeitswert genutzt wird. Dieser Datensatz wird im weiteren Verlauf genutzt, um den Trainings- und Testdatensatz zu erstellen.

Abhängig von der Wahl des Schwellwerts $S \in [0, 1]$ ergeben sich bei diesem Vorgehen unterschiedliche Zielklassen. Wird beispielsweise wie in Abbildung 7.6 ein Schwellwert von 0.5 festgelegt, ist `Quantity` die einzig gültige Zuordnung. Hierbei handelt es sich um den zu `sl-12.1` gehörigen Attributnamen aus dem Zielkatalog. Je näher der Schwellwert an 1.0 liegt, desto eher werden nur $1:1$ Kardinalitäten abgebildet und unverändert übernommene Attributinstanzen berücksichtigt. Bei einer vollständigen Übernahme einer Attributinstanz ist die Ähnlichkeit automatisch 1, sodass sie immer größer gleich einem Schwellwert ist.

Um aus diesen Zuordnungen Zielklassen für ein NN zu erhalten, müssen die Ähnlichkeitswerte auf 1.0 gesetzt werden, wenn die Ähnlichkeit größer gleich dem Schwellwert ist und andernfalls auf null. Da für jeden Attributnamen aus dem Zielschema eine eigene Klasse angenommen wird, deren Zielwert entweder null oder eins ist, lässt sich auf diese Weise ein Zielvektor für eine Mehrklassen-Klassifikation bestimmen. Der daraus resultierende Datensatz wurde in Schmidts et al. [S5] veröffentlicht. Welchen Einfluss die Wahl des Schwellwerts auf die Vorhersagen von ALR besitzen, ist Teil der Evaluation in Kapitel 8.

7.3.3 Datenerweiterung

Die oben beschriebene Strategie zum Aufbau eines Datensatzes mit Silber-Labeln ist einfach, jedoch ist der Datensatz für manche Anwendungen nicht praktikabel. Aufgrund der begrenzten Menge an gesammelten Daten aus dem Integrationspro-

zess könnte ein Modell in Evaluierungen auf einem Testdatensatz gut abschneiden. Es kann allerdings nicht ausgeschlossen werden, dass das Modell zu einem späteren Zeitpunkt auf Produktkataloge trifft, bei denen die Zuordnungen nur schlecht oder gar nicht ermittelt werden können. Um dies so gut wie möglich zu vermeiden, kann der initial genutzte Datensatz erweitert werden. Ziel einer Datenerweiterung ist auch Formulierungen von Herstellern abzudecken, die nicht im Trainingsdatensatz enthalten sind.

Für die Datenerweiterung kommen verschiedene Vorgehensweisen in Betracht:

- Eine Datenerweiterung, die auf einem Thesaurus basiert, ermöglicht die Verwendung bekannter Eingangsattributnamen. So kann der Thesaurus aus dem vorherigen Kapitel genutzt werden, um Eingangsattributnamen der Zulieferer für das gleiche Zielattribut auszutauschen. Dadurch können Benennungen von Zulieferern in den Trainingsdatensatz integriert werden, die nicht bei der Datenerhebung erfasst wurden. Für die Datenerweiterung wird anschließend eine komplette Spalte mit ausgetauschten Attributnamen dem Datensatz hinzugefügt. Dadurch ist eine höhere Robustheit bzgl. wechselnder Attributnamen zu erwarten. Solche thesaurusbasierten Vorgehensweisen wurden bereits in verschiedenen Anwendungen genutzt (vgl. [77]).

- Ein schwierigeres Verfahren zur Datenerweiterung ist der Austausch von Attributinstanzen. Hierfür muss bei Produktdaten die fachliche Bedeutung berücksichtigt werden. Bei Attributen, deren Instanzen durch den Typ Werteliste abgebildet werden, kann das gleiche Verfahren wie oben angewendet werden. Dafür müssen die Instanzen durch gleichwertige Elemente aus den bekannten Wertlisten ausgetauscht werden.

  Weitere Möglichkeiten sind das Verrauschen der Instanzvektoren, beispielsweise durch normalverteilte Zufallswerte, oder indem zufällig Token aus der Attributinstanz gelöscht werden. Alternativ können auch Token durch andere Token mit ähnlichen Wortvektoren ausgetauscht werden. [77]

  All diese erweiterten Möglichkeiten haben jedoch den Nachteil, dass ein Produktkontext nicht berücksichtigt wird. Dadurch wird ein Modell zwar weniger anfällig gegenüber Fehlern in den Eingangskatalogen, kann aber Zusammenhänge gleichzeitig schlechter lernen.

7.3.4 Einschränkungen bei der Datenerhebung

Trotz der manuellen Bearbeitung und Überprüfungen im ETL-Übersetzungsprozess können die Zielkataloge noch immer Qualitätsprobleme enthalten. Zum Beispiel sind stellenweise weiterhin Kodierungsfehler, Rechtschreibfehler oder Fragmente von regulären Ausdrücken enthalten. Diese Qualitätsprobleme können den beschriebenen Prozess für die Generierung des Silberstandards negativ beeinflussen, wenn einem Eingangsattribut mehrere Zielklassen zugeordnet werden, aber keine primäre Zuordnung mit einer Ähnlichkeit von 0.9 oder höher zu erkennen ist.

Des Weiteren wurde die verwendete Sprache der Attributinstanzen während des ETL-Übersetzungsprozesses standardisiert. Dadurch wurden Synonyme und Abkürzungen innerhalb der Attributinstanzen aufgelöst, sodass teilweise die Zuordnungen nicht über die Tokenähnlichkeit gefunden werden können. Trotzdem ist das gewählte Vorgehen geeignet, um den Datensatz automatisiert für die Verwendung in ML-Systemen aufzubereiten, da diese Qualitätsprobleme nur selten auftreten und somit das ML-System nur minimal beeinflussen.

Kapitel 8

Evaluation des ALR-Verfahrens

In diesem Kapitel wird ALR evaluiert. Dabei wird zunächst wird der MRR der primären Zuordnung bestimmt, um die Leistung als Empfehlungssystem für $1:1$ Beziehungen zu evaluieren. Zusätzlich werden die Vorhersagen der primären Zuordnung über ALR mithilfe der Metriken Precision, Recall und F_1-Score bewertet. Dadurch wird sichergestellt, dass sich die Ergebnisse von ALR im Vergleich mit anderen Klassifikationsergebnissen, wie den Ergebnissen aus Kapitel 6 oder der Referenzliteratur, einordnen lassen. Für die Bewertung von $1:M$ Zuordnungen werden die AUC pro Zuordnung und die LRAP ermittelt. Die LRAP-Metrik ermöglicht die Bewertung als Empfehlungsystem für $1:M$ Zuordnungen (siehe Abschnitt 3.6.3).

Für die Beantwortung von FF3 werden die genannten Metriken zusätzlich nach Attributtypen erhoben. Die Unterteilung erfolgt in den Typ *Text*, wenn Attributinstanzen aus mehr als zwei Token bestehen. Andernfalls werden die Instanzen dem Typ *Werteliste* zugeordnet. In der Evaluation werden die folgenden zwei Szenarien untersucht:

- Szenario 1 widmet sich der Fragestellung, welchen Einfluss der bereits beschriebene Schwellwert bei der Festlegung der Zuordnungen im Silberstandard auf die Leistung von ALR besitzt. In diesem Szenario wird eine vereinfachte Struktur für das NN gewählt.

- In Szenario 2 erfolgt die Datenerweiterung für einen Schwellwert, der aus dem vorherigen Szenario ermittelt wurde. Zusätzlich werden die Auswirkungen eines multimodalen NN und verschiedener Vektorisierungsverfahren für die Attributinstanzen untersucht.

Die Implementierung erfolgte in Python mithilfe der Frameworks PyTorch[1] in der Version 1.3.1 und Tensorflow[2] 2.3.0 zur Implementierung der NN und scikit-learn[3] zur Evaluation in der Version 0.22.1. Zusätzlich wurde die Bibliothek FastText [78] für die Generierung der FastText-Vektoren verwendet.

8.1 Szenario 1: ALR mit einem einfachen Netzwerk

Wie beschrieben, soll in diesem Szenario ermittelt werden, wie der gewählte Schwellwert die Leistungsfähigkeit von ALR beeinflusst und ob das gewählte Vorgehen zur Erzeugung der Silber-Labels bei der Katalogintegration genutzt werden kann. Dazu werden die Ergebnisse von ALR mit den Schwellwerten 0.9, 0.5 und 0.0 auf einem Trainingsdatensatz miteinander verglichen. Diese Wahl bildet die verschiedenen Möglichkeiten zur Bestimmung der Beziehungen ab:

- Ein Schwellwert von 0.9 führt dazu, dass nahezu ausschließlich vollständig übernommene Attributinstanzen als Zuordnungen berücksichtigt werden. Somit handelt es sich in der Regel um $1:1$ Kardinalitäten zwischen den Attributen aus Eingangs- und Zielkatalog. Der Schwellwert wird jedoch nicht auf 1.0 festgelegt, um zu vermeiden, dass Beziehungen unentdeckt bleiben, bei denen Rechtschreibfehler korrigiert oder eine Schreibweise normiert wurde. Gleichzeitig ist nicht bekannt, welche konkreten Schritte in der ETL-Pipeline durchgeführt wurden. Durch die Wahl des Schwellwerts von 0.9 können somit auch kleinere Änderungen noch berücksichtigt werden, die ansonsten verloren gehen würden.

- Der Schwellwert von 0.5 soll unscharfe, aber keine irrelevanten Zuordnungen erlernen. Dadurch wird untersucht, inwiefern sich $1:M$ Beziehungen erlernen lassen. Gleichzeitig kann dabei untersucht werden, ob im Vergleich zum Schwellwert von 0.9 Nachteile bei der Zuordnung von $1:1$ Kardinalitäten bestehen.

- Einen Sonderfall stellt der Schwellwert von 0.0 dar. Hier wird jegliche Ähnlichkeit zwischen den Attributinstanzen berücksichtigt, auch möglicherweise irrelevante Zuordnungen. Im Gegensatz zu den vorherigen Schwellwerten wer-

[1] https://pytorch.org/
[2] https://tensorflow.org/
[3] https://scikit-learn.org/

den die Werte für den Zielvektor jedoch nicht auf 1.0 gesetzt, wenn der Schwellwert überschritten wird (bei 0.0 würde sich ein Zielvektor nur aus Einsen ergeben), sondern direkt der Ähnlichkeitswert verwendet. Dadurch ergibt sich bei einem Schwellwert von 0.0 kein binärer Zielvektor für das NN, sondern ein kontinuierlicher.

In den folgenden Abschnitten werden die weiteren Konfigurationen und daraus resultierende Ergebnisse beschrieben.

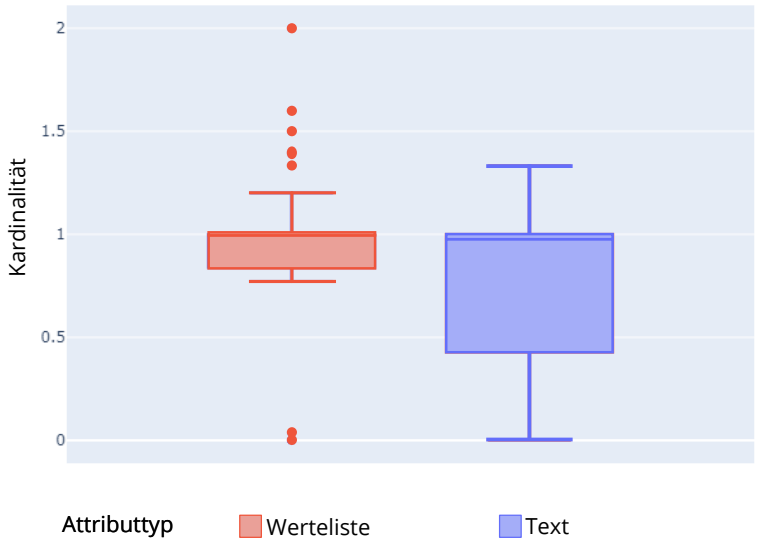
8.1.1 Datensatz

In diesem Szenario wird ein Teil des Datensatzes aus Abschnitt 7.3.1 genutzt und in einen Trainings- und einen Testdatensatz aufgeteilt. Der Trainingsdatensatz besteht aus ca. 50.000 Produkten von sieben verschiedenen Zulieferern, wobei verschiedene Produkttypen, wie beispielsweise enzyme-linked immunosorbent assay (ELISA) Kits oder primäre Antikörper, enthalten sind. Der Testdatensatz besteht aus ca. 25.000 Produkten, die von fünf verschiedenen Zulieferern stammen. Zulieferer in Trainings- und Testdatensatz sind so gewählt, dass kein Zulieferer in beiden Datensätzen enthalten ist.

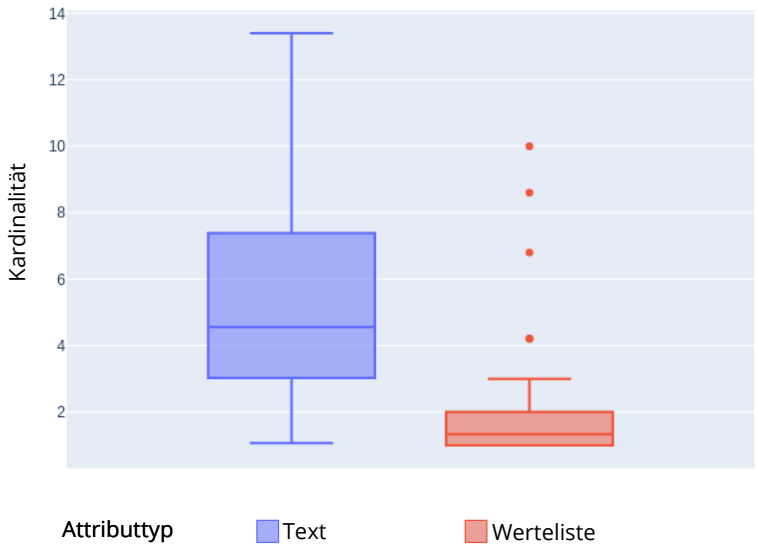
Die Datensätze der Zulieferer besitzen zwischen 14 und 65 verschiedene, gefüllte Attribute, die 94 Attributen aus dem Zielschema zugeordnet werden müssen. Das Zielschema besitzt eigentlich über 130 verschiedene Attribute. Allerdings wurden im vorhandenen Datenausschnitt nur 94 davon für Zuordnungen genutzt.

Der Einfluss der Schwellwerte auf die Anzahl Zuordnungen zwischen den jeweiligen Eingangs- und Zielkatalogen aus dem Silberstandard lässt sich anhand der Kardinalität im Trainingsdatensatz unter Berücksichtigung des Attributtyps in Abbildung 8.1 erkennen. Die Kardinalität ist nach Tsoumakas et al. [48] definiert als die durchschnittliche Anzahl der Zielklassen der Instanzen eines Datensatzes. In Abbildung 8.1 wird die Kardinalität pro Produktkatalog in einem Boxplot dargestellt und nicht zusammengefasst auf den gesamten Trainingsdatensatz. Dadurch lässt sich besser erkennen, wie der Schwellwert die Kardinalität beeinflusst und wie die Kardinalität sich über mehrere Kataloge verhält.

Bei einem Schwellwert von 0.9 ist in Abbildung 8.1a zu erkennen, dass die Kardinalität sowohl bei Wertelisten als auch textbasierten Attributinstanzen im Mittel eins beträgt. In einigen Produktkatalogen gibt es zusätzlich einzelne Instanzen die eine höhere Kardinalität besitzen, jedoch den Wert von zwei nie überschreiten. Darüber hinaus ist der Abbildung zu entnehmen, dass textbasierte Attribute in Produktka-



(a) 0.9



(b) 0.0

Abbildung 8.1: Kardinalität der Zuordnungen abhängig von ihrer Attributart bei unterschiedlichen Schwellwerten.

talogen oftmals eine Kardinalität kleiner eins besitzen. Wie beschrieben, ist diese Beobachtung darauf zurückzuführen, dass insbesondere Texte entweder gar nicht oder nahezu unverändert in die Attributinstanzen des Zielkatalogs übernommen wurden.

Eine deutliche Verschiebung ist bei dem Schwellwert von 0.0 in Abbildung 8.1b zu sehen. Bei Attributen aus der Kategorie Werteliste ist die Kardinalität zwar leicht erhöht, trotzdem müssen selten mehr als zwei Zuordnungen gelernt werden. Bei textbasierten Attributen müssen dagegen im Mittel fünf Zuordnungen gelernt werden. Dabei wird eine maximale Kardinalität von 14 erreicht. Im Gegensatz zum vorherigen Schwellwert, findet sich nahezu immer mindestens eine Zuordnung. Wird ein Schwellwert von 0.5 verwendet, ist die resultierende maximale Kardinalität für textbasierte Attribute fünf (nicht abgebildet).

Anhand dieser Vorbetrachtung lassen sich Hypothesen für die Ergebnisse von ALR ableiten:

1. Zuordnungen von Attributen, die mit dem Typ Werteliste kategorisiert werden, sind einfacher zu erlernen als textbasierte Zuordnungen.

2. Der gewählte Schwellwert besitzt nur einen geringen Einfluss auf das Lernen der Zuordnung von Wertelisten, da sie hauptsächlich $1:1$ Kardinalitäten abbilden.

3. Der Schwellwert von 0.5 eignet sich für Empfehlungssysteme besser als 0.9 und 0.0.

4. Bei einem Schwellwert von 0.5 lassen sich $1:M$ Beziehungen besser vorhersagen als bei 0.9 und 0.0

8.1.2 Aufbau des Modells für ALR

Der Aufbau des NNs für ALR wird für dieses Szenario vereinfacht, um zunächst zu untersuchen, bei welchem Schwellwert die generierten Zuordnungen aus dem Silberstandard für das Training von ALR geeignet sind. Gleichzeitig kann dadurch bewertet werden, ob sich der Aufwand für ein multimodales Modell in diesem Anwendungsfall lohnt. Joulin et al. [71] empfehlen die Verwendung flacher NN, um schnelle und genaue Ergebnisse bei der Klassifikation von Texten zu erzielen. Da Textklassifikation eine Teilaufgabe der Katalogintegration ist, wird, wie in Abbildung 8.2 dargestellt, ein vereinfachter Aufbau genutzt. Bei diesem Aufbau werden die Namens-

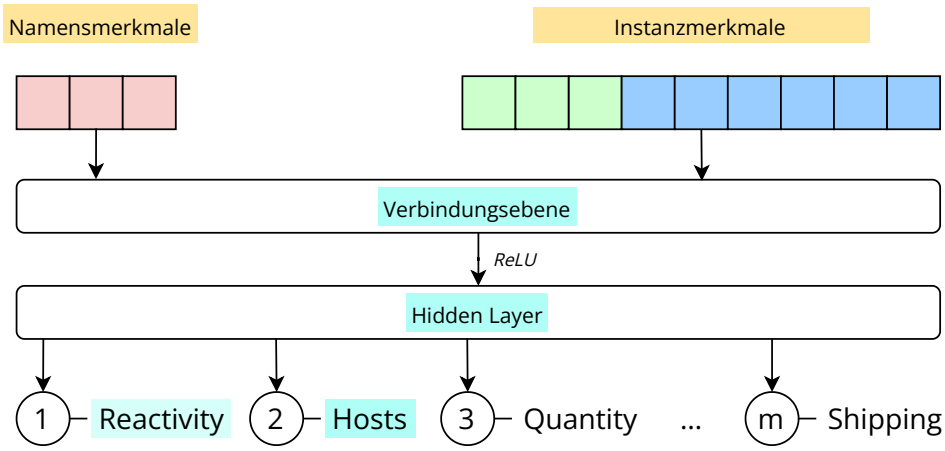


Abbildung 8.2: ALR mit einer einfachen Netzwerkstruktur. Statt zusätzliche NN für Namens- und Instanzvektoren zu nutzen, wird stattdessen direkt die Verbindungsebene genutzt.

und Instanzmerkmale als Feature-Vektoren direkt miteinander verbunden, ohne zusätzliche Netzwerkstrukturen.

Der multimodale Aufbau fällt dadurch weg. Stattdessen wird ein einzelner Feature-Vektor aus allen Merkmalen gebildet, der direkt als Eingangswert für das kombinierte Netzwerk, das hier einem einzigen Hidden Layer entspricht, dient. Verbunden werden die Knoten dieser Ebene über die Aktivierungsfunktion ReLU [43]. Alle Knoten sind vollständig miteinander verbunden. Dennoch können während des Trainingsprozesses einzelne Knoten ignoriert werden, damit das NN stabiler gegenüber Überanpassungen (vgl. Abschnitt 3.4) wird. Der Anteil abgeschalteter Neuronen pro Ebene im Trainingsprozess wird auch Dropout genannt.

Das Modell optimiert die Verlustfunktion Binary Cross Entropy (BCE) [43] über eine ADAGRAD-Optimierung [43]. Die Verlustfunktion BCE bietet die Möglichkeit jede Zielklasse unabhängig zu betrachten und dabei Werte in $[0, 1]$ als Konfidenz vorherzusagen. Im Vorhersageschritt von ALR wird jeder Vorhersagewert größer 0 als Kandidat für die Aggregierung betrachtet. Die ADAGRAD-Optimierung wird genutzt, da sie sich insbesondere bei der Vorhersage unabhängiger Zielklassen als besser erwiesen hat als das übliche Gradient Descent Verfahren [82].

Um zu ermitteln, welche Hyperparameterkonfiguration für diesen Netzwerkaufbau am besten geeignet ist, wurde Optuna[4] [83] gewählt. Über dieses Framework lassen sich die Hyperparameter optimieren, wenn Wertebereiche vorgegeben wer-

[4] https://optuna.org/

Lernrate	0.0095
Dropout-Rate	0.12
Neuronen im Hidden-Layer	407

Tabelle 8.1: Optimale Hyperparameter (gerundet)

den. Dafür muss für jeden Hyperparameter ein eigener Bereich angegeben werden. Häufig werden die drei Hyperparameter Lernrate, Dropout und Größe des Hidden-Layer optimiert.

Als Grenzbereich für die Lernrate wurde das Intervall $[0.005, 0.015]$ und für die Dropout-Rate das Intervall $[0.0, 0.2]$ gewählt. Werden Lernraten zu hoch gewählt, konvergieren NN häufig zu schnell. Bei einer zu niedrigen Lernrate konvergieren sie nur sehr langsam oder gar nicht [84]. Eine Alternative wäre die Verwendung einer adaptiven Lernrate. Bei einer hohen Dropout-Rate sind NN zwar sehr stabil, allerdings können sie meistens keine Zusammenhänge mehr lernen, da zu viele Neuronen abgeschaltet werden [84]. Die Größe des letzten, in diesem Fall einzigen, Hidden-Layers häufig liegt irgendwo zwischen dem arithmetischen Mittel von Eingangsneuronen und Anzahl an Zielklassen und der Anzahl der Zielklassen selbst [84]. Hier wurde ungefährt die dreifache Anzahl der Zielklassen als Neuronen für den Hidden-Layer gewählt. Um eine möglichst gute Konfiguration zu erhalten, wurde als untere Grenze 250 und als obere Grenze 500 Neuronen genutzt. Die optimalen durch Optuna ermittelten Hyperparameter finden sich in Tabelle 8.1.

8.1.3 Zusammensetzung des Feature-Vektors

Wie bereits erwähnt funktionieren Ansätze zur Vorhersage von Attributnamen anhand von Attributinstanzen, wie beispielsweise von Chen et al. [14] bereits gut, wenn Attributnamen für numerische, Wertelisten- und ID-basierte Attribute (vgl. Abschnitt 2.4) vorhergesagt werden sollen. Insbesondere bei Produktkatalogen enthalten Attributinstanzen jedoch häufig unstrukturierten Text, was die Kombination bereits genutzter Merkmale von Chen et al. mit weiteren Verfahren zur Textvektorisierung notwendig macht.

Dafür werden in diesem Szenario die Metadaten aus Tabelle 7.1 mit OH-Vektoren kombiniert, um die Feature-Vektoren der Attributinstanzen zu bilden. Zusätzlich wird der Feature-Vektor um Merkmale aus den Attributnamen erweitert. Damit das ALR zugrundeliegende NN Zuordnungen für jede Attributinstanz vorhersagen kann

bevor diese Spaltenweise aggregiert werden, werden im Feature-Vektor die folgenden Eigenschaften abgebildet:

- **Ähnlichkeit der Attributnamen:** In diesem Szenario wird die Kosinus-Ähnlichkeit der Subtoken Bi- und Trigramme zwischen Eingangs- und Zielattributnamen als Merkmal genutzt. Alternativ könnten die Subtoken N-Gramme auch direkt über eine OH-Kodierung genutzt werden. Aufgrund der beschränkten Anzahl von Attributnamen im Trainingsdatensatz ist an dieser Stelle kein nennenswerter Unterschied zu erwarten.

  Die Verwendung der Kosinus-Ähnlichkeit der Attributnamen besitzt an dieser Stelle den Vorteil, dass die Länge des Vektors auf die Anzahl der Zielklassen (94) beschränkt bleibt, während ein OH-Vektor aus Subtoken N-Grammen mit steigender Anzahl Eingangsattributnamen ebenfalls größer wird. Der Ähnlichkeitsvektor muss nur angepasst werden, wenn auch die Anzahl der Zielattribute bzw. Zielklassen steigt. In diesem Falle muss das Modell ohnehin neu trainiert werden.

- **Metadaten:** Als Metadaten können alle in Tabelle 7.1 angegebenen Eigenschaften der Attributinstanzen genutzt werden. Dazu zählen die normalisierte Anzahl der Zeichen, sowie der Anteil von numerischen, alphabetischen oder symbolischen Zeichen. Diese Metadaten sind insbesondere bei der Vorhersage von Attributnamen bzw. Zuordnungen zwischen Attributinstanzen des Typs ID nützlich [14]. Durch die damit vergleichbaren Eigenschaften von Gensequenzen, Antikörpernamen oder zusammengesetzten Produktnamen lässt sich ebenso für ALR ein positiver Einfluss vermuten.

- **Wortvektoren:** Um die Vorhersage der Zuordnungen von textbasierten Attributinstanzen zu ermöglichen, werden zwei einfache Verfahren genutzt und in der Auswertung miteinander verglichen. Bei beiden Verfahren handelt es sich um OH-Vektoren. Diese unterscheiden sich lediglich in der zuvor genutzten Tokenisierung. Die Token entsprechen entweder einem ganzen Wort oder einem Subtoken N-Gramm.

  Deshalb werden die Modelle, die mit Wort-Token genutzt werden, im weiteren Verlauf mit $\texttt{Wort}_i$ bezeichnet, wobei das i für den genutzten Schwellwert bei der Ermittlung der Zuordnungen steht. Analog dazu werden die Modelle basierend auf Subtoken N-Grammen mit $\texttt{N-Gramm}_i$ bezeichnet. Die Größe des

Vokabulars wird auf die am häufigsten genutzten 50.000 Token beschränkt, um den benötigten Speicher im Trainingsprozess des NN zu begrenzen.

Diese Merkmale werden in der weiteren Bewertung als kombinierter Eingangsvektor für das NN aus Abschnitt 8.1.2 genutzt.

8.1.4 Ergebnisse unter Betrachtung einer einzelnen Zielklasse

In diesem Abschnitt wird die Leistungsfähigkeit von ALR im Hinblick auf die gesamte Spalte bewertet. Zwecks Vergleichbarkeit mit vorherigen Ergebnissen und anderen Verfahren werden zunächst nur die Ergebnisse zur Vorhersage der primären Zuordnung als Zielklasse betrachtet. Diese ergibt sich aus der höchsten durchschnittlichen Ähnlichkeit der Attributinstanzen zwischen Eingangs- und Zielattributen im genutzten Datensatz. Die Vorhersage von ALR ist folglich korrekt, wenn das höchstrangige Label mit dem am häufigsten genutzten Silber-Label der Zuordnungen einer Spalte übereinstimmt. Ist diese primäre Zuordnung die einzige, handelt es sich um eine $1:1$ Beziehung zwischen Eingangs- und Zielattributen.

Durch die Einschränkung auf die Vorhersage einer primären Zuordnung können übliche Metriken zur Bewertung genutzt werden. Die folgenden Bewertungen berücksichtigen zwei Aspekte: Erstens die Resultate als Empfehlungssystem und zweitens die Ergebnisse der Klassifikation. Für die Bewertung als Empfehlungssystem wird der MRR (siehe Abschnitt 3.6.3) genutzt, während für die Klassifikation auf die üblichen Metriken Precision, Recall und F_1-Score zurückgegriffen wird. Da die Zielklassen unbalanciert sind, werden die Metriken pro Zielklasse erhoben und anhand ihrer Häufigkeit gewichtet. Der arithmetische Mittelwert aus den gewichteten Metriken wird im folgenden als *gewichteter Mittelwert* bezeichnet.

Ergebnisse als Empfehlungssystem

Zur Bewertung der Ergebnisse als Empfehlungssystem wird der MRR unter Berücksichtigung der primären Zuordnung gebildet. Dabei werden nur Zuordnungen mit einer minimalen Ähnlichkeit von 0.9 im Testdatensatz (Schwellwert 0.9) als Zielklasse berücksichtigt. Dies entspricht, wie beschrieben, hauptsächlich $1:1$ Beziehungen, deren Attributinstanzen im ETL-Prozess von Eingangs- bis Zielkatalog nur minimal verändert oder vollständig übernommen wurden. Tabelle 8.2 zeigt die Resultate aufgeschlüsselt nach Attributtyp und dem zugrundeliegenden Modell. Die Daten wurden für die Vorhersage nicht weiter aufbereitet.

	MRR		
ALR Modell	Gesamt	Wertelisten	Text
$\texttt{Wort}_{0.0}$	0.7212	0.7678	0.7030
$\texttt{Wort}_{0.5}$	0.7641	0.7857	0.7558
$\texttt{Wort}_{0.9}$	0.7075	0.7678	0.6840
$\texttt{N-Gramm}_{0.0}$	0.7679	0.8125	0.7506
$\texttt{N-Gramm}_{0.5}$	**0.8208**	**0.8393**	**0.8137**
$\texttt{N-Gramm}_{0.9}$	0.7467	0.8214	0.7176

Tabelle 8.2: Der MRR unterteilt nach Attributtyp und Gesamtergebnis. Das ALR zugrundeliegende Modell wurde mit verschiedenen Schwellwerten und Feature-Vektoren trainiert. Das jeweils beste Ergebnis nach Attributtyp ist fettgedruckt.

Im direkten Vergleich der `Wort`- und `N-Gramm`-basierten Modelle in Tabelle 8.2 ist festzustellen, dass `N-Gramm`-basierte ALR deutlich höhere Werte erreichen. Dabei erreicht $\texttt{N-Gramm}_{0.5}$ in beiden Attributtypen sowie insgesamt die höchsten Werte. Insgesamt kann ALR mit dem Modell $\texttt{N-Gramm}_{0.5}$ eine MRR von 0.8208 erreichen. Bei wertelistenbasierten Attributen schneidet diese Konfiguration mit 0.8393 etwas besser ab, während der erreichte Wert bei textbasierten Attributen mit 0.8137 geringer ausfällt. Das Ergebnis ist bemerkenswert, da das Modell mit einem Schwellwert von 0.5 trainiert wurde, aber eine höhere MRR erzielt als die Modelle, die mit einem Schwellwert von 0.9 trainiert wurden, obwohl der Schwellwert von 0.9 für den Testdatensatz verwendet wurde.

Die anderen Konfigurationen zeigen eine ähnliche Charakteristik: Im Allgemeinen ist der erzielte MRR bei wertelistenbasierten Attributen höher als bei textbasierten. Darüber hinaus ist der Abstand zwischen Wertelisten und Text bei der ALR Konfiguration $\texttt{N-Gramm}_{0.5}$ signifikant kleiner als bei den anderen Konfigurationen mit Ausnahme von $\texttt{Wort}_{0.5}$.

Die besseren Ergebnisse von `N-Gramm`-basierten ALR lassen sich durch OOV-Fehler und Qualitätsprobleme der Attributinstanzen erklären: Beispielsweise fehlt das Token `96test`, das nur im Testdatensatz vorkommt, im Vokabular der `Wort`-Modelle aufgrund des fehlenden Leerzeichens zwischen 96 und `test`. Enthält eine gesamte Spalte nur Token die zu OOV-Fehlern und damit Nullvektoren führen, kann ein Modell unter Umständen keine Zuordnung vorhersagen, falls andere Merkmale nicht

signifikant sind. Ein N-Gramm-basierter Ansatz erhält zumindest teilweise Übereinstimmungen, wenn das Vokabular beispielsweise die Trigramme von `test` (`tes`, `est`) enthält. Hier ist insbesondere die beschriebene Robustheit von N-Grammen gegenüber OOV-Fehlern für die besseren MRR-Werte verantwortlich.

Werden die N-Gramm- und Wort-Modelle einzeln betrachtet, schneiden jeweils die Modelle in jeder Kategorie am besten ab, die mit dem Schwellwert $t = 0.5$ trainiert wurden. Dieses Ergebnis ist durch den Aufbau von ALR zu erklären: Dadurch dass die Modelle darauf trainiert werden mehrere mögliche Zuordnungen (sofern vorhanden) vorherzusagen, wird die Vorhersage einer primären Zuordnung schwieriger für das Modell. Die anschließende Aggregation sorgt jedoch analog einem Mehrheitsentscheid (Voting vgl. Abschnitt 3.5.4) dafür, dass bei der spaltenweisen Betrachtung die Mehrheit der Vorhersagen die primäre Zuordnung enthält. Somit wird die Unschärfe der einzelnen Vorhersagen ausgeglichen und das Gesamtergebnis verbessert.

Klassifikationsergebnisse

Für die Evaluation der Klassifikationsergebnisse kann der gleiche Aufbau wie für die Bewertung der Empfehlungen genutzt werden. Folglich wird erneut die primäre Zuordnung als Zielklasse gewertet. Somit spiegeln die Klassifikationsergebnisse die Vorhersage von Zuordnungen mir $1:1$ Beziehungen wider. Im Gegensatz zur vorherigen Auswertung wird allerdings nur die Zuordnung mit dem höchsten Rang als Klassifikationsergebnis genutzt. Falls ALR für einzelne Spalten kein Ergebnis vorhersagen konnte, wird dieses als falsch-negativ gewertet. Daher verändern nicht vorhergesagte Spalten den Recall und den F_1-Wert, nicht aber die Precision.

Tabelle 8.3 zeigt die Gesamtergebnisse von ALR abhängig von dem genutzten Modell. Die Precision liegt für alle genutzten Modelle bei über 0.8 und erreicht den Höchstwert von 0.893 mit dem Modell $\texttt{Wort}_{0.5}$, gefolgt von $\texttt{N-Gramm}_{0.5}$ (0.8766) und $\texttt{Wort}_{0.9}$ (0.8735). Im Gegensatz zu den Empfehlungen schneidet $\texttt{Wort}_{0.5}$ somit bei der Precision besser ab als die N-Gramm-Modelle. Dennoch erreicht $\texttt{N-Gramm}_{0.5}$ den insgesamt höchsten Recall (0.805) und F_1-Score (0.8393), gefolgt von $\texttt{Wort}_{0.5}$ mit einem Recall von 0.75 und einem F_1-Score von 0.8153. Erneut ist zu erkennen, dass die Modelle, die auf einem Schwellwert von 0.5 trainiert wurden besser abschneiden als andere Modelle.

Tabelle 8.4 schlüsselt die Klassifikationsergebnisse nach Attributtyp auf. Der Tabelle ist zu entnehmen, dass bei Attributinstanzen aus Wertelisten die Precision für

	Gesamt		
ALR Modell	Precision	Recall	F_1
$\texttt{Wort}_{0.0}$	0.8130	0.6750	0.7376
$\texttt{Wort}_{0.5}$	**0.8930**	0.7500	0.8153
$\texttt{Wort}_{0.9}$	0.8735	0.7050	0.7803
$\texttt{N-Gramm}_{0.0}$	0.8368	0.7400	0.7854
$\texttt{N-Gramm}_{0.5}$	0.8766	**0.8050**	**0.8393**
$\texttt{N-Gramm}_{0.9}$	0.8595	0.7014	0.7724

Tabelle 8.3: Gewichteter Mittelwert der jeweiligen Metrik. Dargestellt wird das Gesamtergebnis von ALR mit den verschiedenen Modellkonfigurationen, wobei die besten Ergebnisse nach Metrik fettgedruckt sind.

alle Modelle bei größer gleich 0.9 liegt. Wie zuvor erreicht $\texttt{Wort}_{0.5}$ die höchste Precision (0.9778). Weiterhin ist zu erkennen, dass wortbasierte Modelle im Hinblick auf die Precision erheblich besser abschneiden als die N-Gramm-Modelle. Umgekehrt verhält sich der Recall. Hier schneiden die N-Gramm-Modelle besser ab. Dieses Ergebnis kann auf verschiedene Ursachen zurückzuführen sein:

Die hohe Precision bei dem Wort-basierten ALR bei Attributinstanzen aus Wertelisten liegt wahrscheinlich an einer Überanpassung der Modelle, sodass diese die Wertelisten mutmaßlich auswendig lernen konnten. Gleichzeitig ist der geringere Recall auf OOV-Fehler zurückzuführen. Die Unterschiede zwischen Precision und Recall der verschiedenen ALR-Konfigurationen haben nur einen geringen Einfluss auf den F_1-Score, sodass die Modelle ähnliche Werte erreichen, wobei ALR basierend auf $\texttt{Wort}_{0.5}$ mit 0.8713 den höchsten Wert erreicht, dicht gefolgt von $\texttt{N-Gramm}_{0.5}$ mit 0.8695.

Im direkten Vergleich zwischen text- und wertelistenbasierten Attributen erzielen alle Modelle schlechtere Ergebnisse bei der Vorhersage von primären Zuordnungen von textbasierten Attributen. Lediglich die ALR-Konfiguration $\texttt{N-Gramm}_{0.5}$ kann die zuvor erzielten Precision Werte nahezu erreichen. Gleichzeitig erreicht diese Konfiguration auch die höchsten Werte bezüglich Recall (0.7917) und F_1-Score (0.8420). Die zweitbesten Ergebnisse werden erneut von der Konfiguration $\texttt{Wort}_{0.5}$ erreicht. Begründen lassen sich diese Ergebnisse erneut durch OOV-Fehler bzw. ungünstigere Tokenisierung bei Wort-Modellen.

	Wertelisten			Text		
Model	Precision	Recall	F_1	Precision	Recall	F_1
$\texttt{Wort}_{0.0}$	0.9773	0.7679	0.8600	0.7585	0.6389	0.6936
$\texttt{Wort}_{0.5}$	**0.9778**	0.7857	**0.8713**	0.8681	0.7361	0.7967
$\texttt{Wort}_{0.9}$	0.9356	0.7679	0.8434	0.8543	0.6806	0.7576
$\texttt{N-Gramm}_{0.0}$	0.9320	0.8036	0.8631	0.8186	0.7153	0.7635
$\texttt{N-Gramm}_{0.5}$	0.9020	**0.8393**	0.8695	**0.8992**	**0.7917**	**0.8420**
$\texttt{N-Gramm}_{0.9}$	0.9000	0.8214	0.8589	0.8681	0.7014	0.7759

Tabelle 8.4: Klassifikationsergebnisse der primären Zuordnung aufgeschlüsselt nach Attributtyp. Die Tabelle zeigt mehrere Modell-Konfigurationen mit den erzielten gewichteten Mittelwerten je Metrik.

Weiterhin ist festzuhalten, dass Modelle, die ohne Schwellwert (0.0) trainiert wurden, in jeder Hinsicht deutlich schlechter abschneiden als Modelle, die einen Schwellwert nutzen. Durch den fehlenden Schwellwert können die NN Eigenschaften aus den genutzten Datensätzen schlechter lernen, als wenn die Kardinalitäten der Zuordnungen eingeschränkt werden. Auch die nachfolgende Aggregation und Bewertung einer gesamten Spalte eines Produktkataloges kann die Qualität der Ergebnisse nicht verbessern.

Die bisherige Betrachtung lässt sich um unsichere Zuordnungen erweitern. Werden statt sicheren Zuordnungen mit einer Ähnlichkeit größer 0.9 auch unsichere Zuordnungen, über einen Schwellwert von 0.5 berücksichtigt, verändert sich die primäre Zuordnung im Testdatensatz zwischen den Spalten. Die primäre Zuordnung ist lediglich die häufigste Zuordnung zwischen Eingangs- und Zielkatalog im Datensatz. Für die weitere Auswertung werden nur die beiden Konfigurationen mit dem Schwellwert von 0.5 genutzt, da sie bei nahezu sicheren Zuordnungen die besten Resultate erzielen konnten. Einschränkend ist hinzuzufügen, dass durch Mehrdeutigkeit trotzdem nicht klar ist, welche Zuordnungen genau im Integrationsprozess durchgeführt wurden.

Wie in Tabelle 8.5 zu sehen ist, bleiben Recall und F_1-Score der Konfiguration $\texttt{N-Gramm}_{0.5}$ auch bei unscharfen Zuordnungen insgesamt höher, als bei der Wort-Tokenisierung, wobei die Precision in dieser Konfiguration höher ist. Während beide Konfigurationen nur leicht schlechtere Precision Werte erreichen, fallen die Werte

	Gesamt		
ALR Modell	Precision	Recall	F_1
$\texttt{Wort}_{0.5}$	**0.8652**	0.6320	0.7304
$\texttt{N-Gramm}_{0.5}$	0.8418	**0.6877**	**0.7570**

Tabelle 8.5: Gesamtergebnis von ALR mit zwei Modellkonfigurationen unter Berücksichtigung unscharfer Zuordnungen. Die besten Ergebnisse nach Metrik sind fettgedruckt, wobei jeweils der gewichtete Mittelwert genutzt wird.

	Wertelisten			Text		
ALR Model	Precision	Recall	F_1	Precision	Recall	F_1
$\texttt{Wort}_{0.5}$	**0.9787**	0.6301	**0.7666**	0.8509	0.6327	0.7258
$\texttt{N-Gramm}_{0.5}$	0.8098	**0.6986**	0.7501	**0.8733**	**0.6837**	**0.7669**

Tabelle 8.6: Klassifikationsergebnisse aufgeschlüsselt nach Attributtyp bei unscharfen Zuordnungen. Die Tabelle zeigt zwei Modell-Konfigurationen mit den erzielten gewichteten Mittelwerten je Metrik.

bezüglich Recall und F_1-Score signifikant ab. Hier ist der Einfluss der unscharfen primären Zuordnung deutlich zu sehen.

Grundsätzlich lassen sich diese Veränderungen auch aufgeteilt nach Attributtyp in Tabelle 8.6 beobachten. Bei Wertelisten fällt jedoch auf, dass die Precision mit der $\texttt{Wort}_{0.5}$-Konfiguration im Vergleich zur sicheren primären Zuordnung nicht abnimmt, während der Precision Wert der $\texttt{N-Gramm}_{0.5}$-Konfiguration um ca. 0.1 sinkt. Die beiden anderen Metriken sinken für beide Konfigurationen ebenfalls um über 0.1, was bereits in der Gesamtbetrachtung sichtbar war.

Bei textbasierten Attributen schneidet, wie zuvor, die Konfiguration $\texttt{N-Gramm}_{0.5}$ am besten ab. Allerdings sind bei beiden Konfigurationen die Precision-Werte nur leicht schlechter als zuvor, während Recall und F_1-Score erheblich sinken.

Zusammenfassend lässt sich festhalten, dass bei der Klassifikation der primären Zuordnung die Konfigurationen mit einem Schwellwert von 0.5 insgesamt am besten abschneiden, wobei die wortbasierten Modelle Vorteile im Hinblick auf die Precision bei Wertelisten besitzen und $\texttt{N-Gramm}_{0.5}$ insbesondere bei textbasierten Attributinstanzen bessere Ergebnisse erzielen kann. Im Hinblick auf die Precision lässt sich auch die primäre Zuordnung von unscharfen Beziehungen vorhersagen, allerdings mit Einschränkungen bei Recall und F_1-Score.

	AUC		
ALR Model	Gesamt	Wertelisten	Text
$\texttt{Wort}_{0.0}$	0.732	0.691	0.740
$\texttt{Wort}_{0.5}$	0.711	0.681	0.719
$\texttt{Wort}_{0.9}$	0.709	0.768	0.684
$\texttt{N-Gramm}_{0.0}$	0.727	0.713	0.730
$\texttt{N-Gramm}_{0.5}$	**0.746**	0.707	**0.757**
$\texttt{N-Gramm}_{0.9}$	**0.746**	**0.821**	0.718

Tabelle 8.7: Gewichteter Mittelwert des AUC für ALR bei der Vorhersage mehrerer gleichzeitiger Zuordnungsmöglichkeiten. Verglichen werden die Modellkonfigurationen und Attributtypen, wobei die besten Ergebnisse fettgedruckt sind.

8.1.5 Ergebnisse mit mehreren Zielklassen

ALR hat das Ziel nicht nur die primäre Zuordnung für eine Spalte vorherzusagen, sondern möglichst alle vorhandenen Zuordnungen zu erkennen, um mehrdeutige Spalten bzw. Eingangsattribute zu identifizieren. Beispielsweise enthält ein Produktname oft zusätzliche Informationen, wie Informationen über die Reaktivität, die Menge und den Wirt eines Antikörpers. Wenn diese Daten in anderen Eingangsattributen fehlen, brauchen die Integrationsspezialisten einen Hinweis darauf, wo diese Informationen zu finden sind. Deshalb wird ALR so trainiert, dass auch $1:M$ Beziehungen von unsicheren Zuordnungen erkannt werden können.

Für die Evaluation mehrerer gleichzeitig angenommener Zuordnungen eignet sich die AUC-Metrik (vgl. Abschnitt 3.6.3). Der AUC Wert beschreibt die Fähigkeit eines Modells, zwischen Zielklassen, hier Zuordnungen, zu unterscheiden. Ein Wert von 0.5 bedeutet, dass die Zielklassen nicht unterschieden werden können, ein Wert von 1.0 bedeutet eine ideale Zielklassenunterscheidung. Um den unbalancierten Testdatensatz zu berücksichtigen, wird auch für AUC der gewichtete Mittelwert der Vorhersagen gewählt.

Die Auswertung des AUC der vorgestellten Konfigurationen ist Tabelle 8.7 zu entnehmen. Dabei erreichen ALR mit den beiden Konfigurationen $\texttt{N-Gramm}_{0.5}$ und $\texttt{N-Gramm}_{0.9}$ die höchsten AUC-Werte (insgesamt 0.746). Die Konfiguration $\texttt{N-Gramm}_{0.9}$ erreicht gleichzeitig für Wertelisten den höchsten Wert (0.821), während $\texttt{N-Gramm}_{0.5}$ zusätzlich den höchsten Wert für textbasierte Attribute (0.757) erzielt. Dies steht im Gegensatz zu den bisherigen Ergebnissen, bei denen nicht die Tokenisierung,

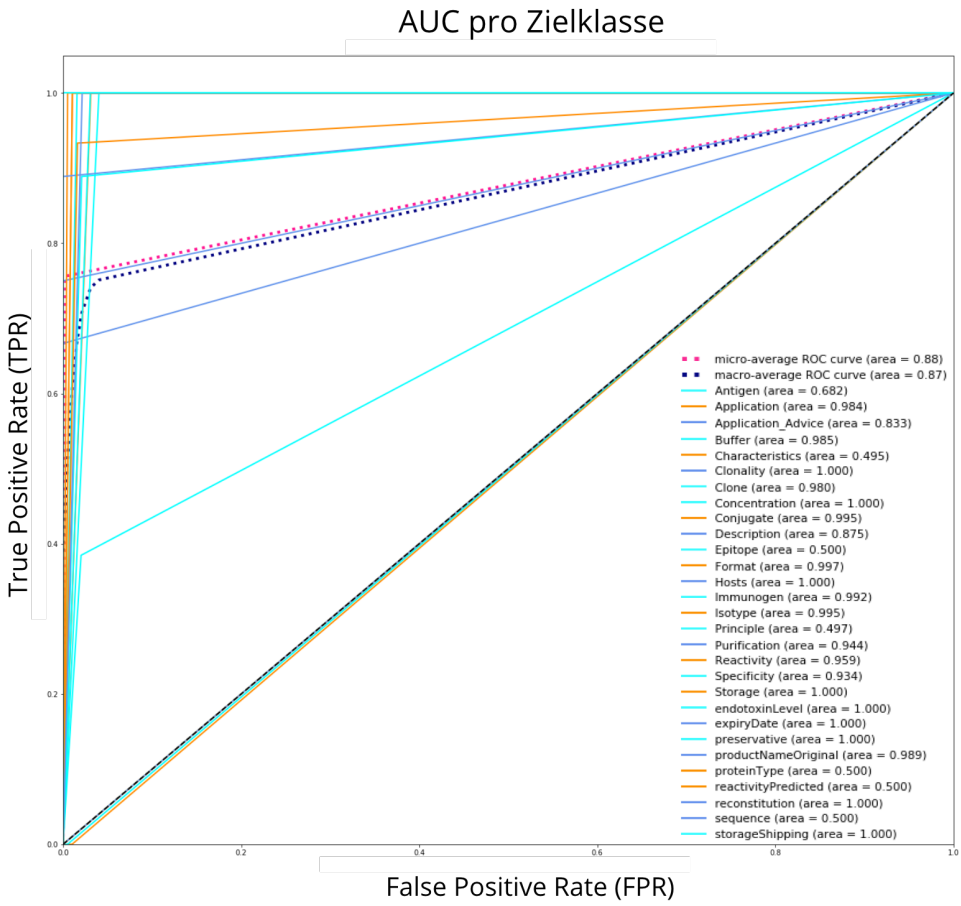


Abbildung 8.3: AUC pro Zielklasse

sondern der Schwellwert entscheidend für das gute Abschneiden ist. Des Weiteren kann der hohe Wert von $\texttt{N-Gramm}_{0.9}$ für Wertelisten darauf zurückzuführen sein, dass Wertelisten oft nur eine Zielklasse besitzen.

Die erzielten Werte deuten darauf hin, dass ALR zwar nicht alle Zielklassen ideal unterscheiden kann, aber dass diese Unterscheidung in den meisten Fällen möglich ist. Bei näherer Betrachtung der Zielklassen in Abbildung 8.3 sind beispielsweise einzelne Zielklassen zu finden (wie *Antigen, Characteristic, Epitope, Principle, proteinType, reactivityPredicted*), bei denen ALR keine Merkmale zur Unterscheidung erlernen konnte. Zusätzlich konnten weitere Zielattribute identifiziert werden, deren Mehrdeutigkeit auch von Spezialisten nicht immer eindeutig aufgelöst werden konnte (*Storage, storageComment, storageShipping*) oder die je nach Zulieferer anders ver-

wendet werden. Trotzdem konnte die Mehrheit der Zielattribute mit Werten von annähernd 1.0 gut unterschieden werden.

8.2 Szenario 2: Multimodales Modell, Datenerweiterung und FastText-Vektoren

Nachdem im vorherigen Abschnitt ALR mit einem vereinfachten Modell sowie einfachen Vektorisierungsverfahren für verschiedene Schwellwerte auf automatisch generierten Silber-Labeln evaluiert wurde, widmet sich dieses Szenario dem Einfluss eines multimodalen NN. Zusätzlich wird untersucht, wie sich die Datenerweiterung aus Abschnitt 7.3.3 auf die Klassifikations- und Empfehlungsergebnisse auswirkt.

Durch die generalisierte Struktur des NN (siehe Abbildung 7.2), können Features aus den Attributnamen unabhängig von Features aus den Attributinstanzen erlernt werden, bevor sie für die letztendliche Entscheidung wieder zusammengeführt werden. Des Weiteren kann eine Convolutional Neural Network (CNN) Architektur für das NN der Attributinstanzen genutzt werden, wodurch ein Vergleich zwischen einfachen NN-Architekturen und komplexeren ermöglicht wird.

Darüber hinaus wird in diesem Szenario ermittelt, ob komplexe Wortvektoren wie FastText, die auf die Produktdomäne angepasst wurden, bei Klassfikations- und Empfehlungsaufgaben besser abschneiden als die einfachen Wortvektoren. Wie im vorherigen Szenario wird auch in diesem zwischen Attributen, die auf Wertelisten basieren, und textbasierten Attributen unterschieden.

Da bei der Generierung der Silber-Labels nicht klar ist, inwiefern die Unicode-Normalisierung als aufwändiger Schritt in der Vorverarbeitung auf die Lernfähigkeit der NN einwirkt, werden in diesem Szenario zusätzlich Konfigurationen ausgewertet, bei denen keine Unicode-Normalisierung durchgeführt wurde.

8.2.1 Aufbau und Konfiguration

Durch den Aufbau dieses Szenarios soll die Vergleichbarkeit der zusätzlichen Konfigurationen mit den vorausgegangenen in Szenario 1 gewährleistet werden. Aus diesem Grund wird ein ähnlicher Aufbau der Datensätze verwendet, allerdings mit einem erweiterten Testdatensatz, um auch die Generalisierung von ALR abzubilden. Im Gegensatz zu Szenario 1 erfolgt keine Unterteilung der Datensätze anhand der Schwellwerte. Wie den vorherigen Ergebnissen zu entnehmen ist, eignet sich ein Schwellwert von 0.5 am besten. Szenario 2 verwendet daher ausschließlich den

Schwellwert von 0.5, der den unscharfen $1 : M$ Zuordnungen von Eingangs- und Zielkatalog entspricht

Wie in Szenario 1 gewählt enthält der Trainingsdatensatz ca. 50.000 Produkte von sieben verschiedenen Herstellern. Der Testdatensatz wird dagegen um 25.000 Produkte auf 50.000 Produkte erweitert. Beide Datensätze enthalten weiterhin verschiedene Produkttypen aus dem Antikörperbereich, wie ELISA-Kits oder primäre Antikörper. Ebenso gleicht sich die Anzahl der Zielattribute. Die Anzahl der vorherzusagenden Zuordnungen variierten weiterhin je nach Zulieferer und Produkttyp (vgl. Abschnitt 7.3.1).

Um neben den bereits eingesetzten einfachen Wortvektoren auch die komplexeren FastText-Vektoren in die Untersuchungen einbeziehen zu können, wurde das von FastText bereitgestellte Englische-Sprachmodell[1] genutzt. Dieses kann nur eingeschränkt mit Begriffen aus dem Antikörperumfeld oder der genutzten Fachsprache umgehen, da es auf allgemein verfügbaren Texten trainiert wurde. Da die genutzte Fachsprache in Produktbeschreibungen in diesen Texten nur eingeschränkt vertreten ist und somit die Bedeutungen der Worte unzureichend ermittelt werden können, wurde das Englische-Sprachmodell als Grundlage genutzt, um ein angepasstes Modell auf den Attributinstanzen zu trainieren (sog. Transfer-Learning).

Da FastText darauf angewiesen ist im Trainingsprozess möglichst ganze, grammatikalisch möglichst korrekte Sätze zu sehen, um für ähnliche Wörter ähnliche Vektorrepräsentationen zu ermitteln, wurden nur Attributinstanzen mit diesen Eigenschaften genutzt. Darunter fällt unter anderem die Produktbeschreibung, da diese dazu genutzt wird, einem möglichen Kunden eine vollständige textuelle Beschreibung eines Produkts sowie dessen Anwendungsmöglichkeiten zu liefern. Das Training der FastText-Vektoren erfolgte über alle geeigneten Attributinstanzen aus dem gesamten Datensatz aus Abschnitt 7.3.1, um eine ausreichende Datengrundlage zu gewährleisten.

Für den Einsatz der FastText-Vektoren bieten sich verschiedene Konfigurationen an:

1. Der Ersatz der bisher genutzten `N-Gramm`- bzw. `Wort`-Vektoren durch FastText-Vektoren kann ohne Änderung am Aufbau von ALR erfolgen. Dieser Ansatz benötigt vor allem deshalb keine weitere Veränderung, da die Dimension der FastText-Vektoren auf 300 beschränkt ist. Im Gegensatz zu den beiden anderen Vektorvarianten wird der Vektor hier nicht durch die Größe des Vokabu-

[1] https://fasttext.cc/docs/en/english-vectors.html

lars (über 50.000 Token) bestimmt. Daher ist eine tiefergehende Architektur mit zusätzlichen Layern höchstwahrscheinlich nicht zielführend.

2. Durch die geringere Dimensionalität können die FastText-Vektoren alternativ als Eingangswerte für ein CNN genutzt werden. Die Idee eines CNN zur Textklassifikation ist, Satzstrukturen und die Reihenfolge von Worten (Token) auszunutzen, um eine bessere Klassifikation durchzuführen. Dabei reichen bereits einfache CNN mit einer einzelnen Faltungsebene (Kombination von Worten) und einem einzelnen Max-Pooling (Nutzen des wahrscheinlichsten Wortes) aus, um gute Ergebnisse zu erzielen [85].

Für diese Untersuchung wird ein einfaches CNN genutzt. Dieses besteht aus vier Stufen. Zuerst werden die FastText-Vektoren jedes eines einzelnen Wortes einer Attributinstanz als Eingangswerte in eine Faltungsoperation (Convolution) genutzt. Die Länge der Sätze wird auf 128 Worte beschränkt. Sollten weniger Worte genutzt werden, wird die Wortfolge so lange wiederholt, bis der Satz 128 Worte enthält, da ein CNN eine starre Länge besitzt und jeder Satz genau gleich lang sein muss. Danach folgt eine Max-Pooling-Operation, die das Maximum aus zwei aufeinander folgenden Wortvektoren nutzt. Das CNN wird abgeschlossen durch eine einzelne vollverbundene Ebene. Der Feature-Vektor der Attributnamen wird mit dieser Ebene verbunden. Dieses kombinierte NN wird dann von einer Ebene abgeschlossen, deren Größe der Anzahl Zielklassen entspricht. Weitere Hintergrundinformationen zu CNNs und deren Einsatz zur Textklassifikation finden sich in den Veröffentlichungen [85, 86, 87].

Aus diesen beiden Konfigurationen ergeben sich deutliche Unterschiede hinsichtlich der Anfälligkeit für unzureichende bzw. schlechte FastText-Vektoren. Während der einfache FastText-Ansatz direkt über FastText einen Satzvektor bildet, sodass einzelne Abweichungen oder schlecht gelernte Worte weniger ins Gewicht fallen, ergibt sich bei der CNN-Architektur das Problem, dass gerade Attribute, die Wertlisten enthalten, stark ins Gewicht fallen. Zusätzlich können eigentliche Schlüsselworte, die bei der Zuordnung hilfreich wären, durch die Beschränkung auf 128 Worte pro Attribut die Vorhersage nicht beeinflussen. In der Folge muss die weitere Auswertung aufzeigen, welches Vorgehen bei Produktdaten vorteilhaft ist. In der Auswertung werden die beiden Modelle als FT bzw. FT-CNN bezeichnet.

Damit die Vergleichbarkeit zwischen den beiden Szenarien gewährleistet ist, werden die besten Konfigurationen aus Szenario 1 Referenzmodelle genutzt. Die Modelle werden analog zur bisherigen Namensgebung mit $\texttt{Wort}_{0.5}$ und $\texttt{N-Gramm}_{0.5}$

bezeichnet. Bei diesen Modellen fand keine weitere Vorverarbeitung statt. Wenn eine Vorverarbeitung durchgeführt wurde, wird dies als Index aufgeführt. Findet sich der Text `erw` im Index, wurde die Datenerweiterung aus Abschnitt 7.3.3 genutzt. Der Text `norm` im Index wird für Modelle genutzt, bei denen eine Unicode-Normalisierung durchgeführt wurde. Werden beide Texte im Modellnamen genutzt, wurden beide Vorverarbeitungsschritte durchgeführt.

8.2.2 Klassifikationsergebnisse

Für alle folgenden Auswertungen wurden, wie bereits beschrieben, Zuordnungen aus dem Silberstandard über den Schwellwert 0.5 generiert. Dadurch werden die Modelle auf unscharfe $1 : M$ Zuordnungen trainiert. Wie zuvor auch, werden in diesem Abschnitt die Ergebnisse der Klassifikation der primären Zuordnung ausgewertet. Dabei werden die Gesamtergebnisse sowie die Ergebnisse unterteilt nach Attributtyp aufgeschlüsselt. Grundlage der Auswertung sind weiterhin die gewichteten Mittelwerte der einzelnen Metriken, um den unbalancierten Testdatensatz zu berücksichtigen.

Tabelle 8.8 enthält die Gesamtergebnisse für die Klassifikation der primären Zuordnung. Die genutzten Referenzmodelle $\texttt{Wort}_{0.5}$ und $\texttt{N-Gramm}_{0.5}$ schneiden dabei deutlich schlechter ab als in Szenario 1. So sinkt beispielsweise die Precision von ALR mit dem Modell $\texttt{Wort}_{0.5}$ von 0.893 auf 0.8073 und der Recall von 0.75 auf 0.3197. Ebenso sinkt der F_1-Score auf 0.278. Eine ähnliche Verschlechterung ist für $\texttt{N-Gramm}_{0.5}$ zu erkennen. Hierbei sinkt der Recall jedoch nur auf 0.4832 und der F_1-Score auf 0.4556. Diese Verschlechterung lässt darauf schließen, dass die Modelle aus Tabelle 8.8 nicht gut generalisieren, da der Testdatensatz lediglich mehr Produkte umfasst als bei Szenario 1.

Beim Vergleich der verschiedenen `N-Gramm`-Modelle kann der Einfluss der jeweiligen Vorverarbeitung abgeleitet werden. Während die Unicode-Normalisierung so gut wie keinen Effekt besitzt bzw. sogar zu einer Verschlechterung der Klassifikationsergebnisse führt, ist der positive Einfluss der Datenerweiterung klar zu abzulesen. Dieser wirkt sich hauptsächlich in einer Verbesserung des Recalls von 0.4832 auf 0.5687 aus. Damit einher geht die Erhöhung des F_1-Scores auf 0.5425, während die Precision unabhängig von der Vorverarbeitung nahezu konstant bleibt. Erst die Kombination der beiden Vorverarbeitungsschritte führt zu einer nennenswerten Verbesserung der Precision auf 0.8239 bei gleichbleibend hohem Recall. Dadurch

	Gesamt		
ALR Model	Precision	Recall	F_1
$\texttt{Wort}_{0.5}$	0.8073	0.3197	0.2780
$\texttt{N-Gramm}_{0.5}$	0.8197	0.4832	0.4556
$\texttt{N-Gramm}_{\text{norm}}$	0.8191	0.4795	0.4595
$\texttt{N-Gramm}_{\text{erw}}$	0.8196	**0.5687**	0.5425
$\texttt{N-Gramm}_{\text{erw+norm}}$	0.8239	**0.5687**	**0.5495**
$\texttt{FT}_{\text{erw+norm}}$	**0.8416**	0.3940	0.3625
$\texttt{FT-CNN}_{\text{erw+norm}}$	0.5500	0.1636	0.1521

Tabelle 8.8: Gesamtergebnis der Klassifikation für die primäre Zuordnung von ALR mit verschiedenen Modellkonfigurationen. Die höchsten Werte sind jeweils fettgedruckt. Alle Ergebnisse sind die gewichteten Mittelwerte.

erreicht die Kombination der Vorverarbeitungsschritte gleichzeitig den höchsten F_1-Score von 0.5495.

Im Ergebnis führt folglich die Kombination der Vorverarbeitungsschritte zu den besten Resultaten. Daher werden in der Tabelle für die FastText-Varianten nur Werte für die kombinierten Schritte aufgeführt. Mit 0.8416 erreicht die einfache FastText-Variante die insgesamt höchste Precision. Allerdings fallen Recall und F_1-Score geringer aus als bei den `N-Gramm`-Modellen. ALR mit FastText-Modell und CNN-Architektur schneiden dagegen wesentlich schlechter ab, als alle anderen ALR-Varianten.

Insgesamt erreicht das Modell $\texttt{FT-CNN}_{\text{erw+norm}}$ nur eine Precision von 0.55, einen Recall von 0.1636 und einen F_1-Score von 0.1521. Die Unterschiede in den FastText-Varianten sind höchstwahrscheinlich auf die beschriebene Problematik bei den Satzvektoren zurückzuführen. Während sich ein Satzvektor für Attributinstanzen aus dem Mittelwert aller normalisierten Tokenvektoren zusammensetzt, wird die Anzahl der berücksichtigten Worte bei der CNN-Architektur auf 128 begrenzt. Somit verliert die CNN-Variante vermutlich wichtige Informationen für die Zuordnungen.

Diese Schlussfolgerung lässt sich bei der Betrachtung von Tabelle 8.9 erhärten. Während die Differenzen zwischen Attributen des Typs Werteliste und textbasierten Attributen aller anderen ALR Modelle kleiner 0.1 ist, fällt die Precision des Modells `FT-CNN` von 0.8422 auf 0.5. Wie schon in Szenario 1 erreichen alle Modelle für Wertelisten bessere Werte als für textbasierte Attributinstanzen. Einzige Ausnahme bildet das Modell $\texttt{Wort}_{0.5}$, das für Text einen höheren Recall und F_1-Score erreicht.

	Wertelisten			Text		
ALR Model	Precision	Recall	F_1	Precision	Recall	F_1
$\texttt{Wort}_{0.5}$	0.8637	0.2876	0.1827	0.8340	0.3316	0.3107
$\texttt{N-Gramm}_{0.5}$	**0.8941**	0.6301	0.5823	0.8101	0.4285	0.4148
$\texttt{N-Gramm}_{\text{norm}}$	**0.8941**	0.6301	0.5823	0.8338	0.4234	0.4340
$\texttt{N-Gramm}_{\text{erw}}$	0.8829	**0.6575**	**0.6064**	0.8312	**0.5357**	0.5343
$\texttt{N-Gramm}_{\text{erw+norm}}$	0.8829	**0.6575**	**0.6064**	**0.8413**	**0.5357**	**0.5460**
$\texttt{FT}_{\text{erw+norm}}$	0.8683	0.4520	0.4016	0.8386	0.3724	0.3470
$\texttt{FT-CNN}_{\text{erw+norm}}$	0.8422	0.2238	0.1769	0.5000	0.1786	0.1526

Tabelle 8.9: Klassifizierungsergebnisse für die primäre Zuordnung von ALR in Abhängigkeit des Attributtyps mit verschiedenen Modellkonfigurationen. Die höchsten Werte sind jeweils fettgedruckt. Alle Ergebnisse sind die gewichteten Mittelwerte.

Wie zuvor erreichen die N-Gramm-Modelle in jeder Hinsicht die besten Resultate. Die höchste Precision bei Wertelisten erreichen mit 0.8941 die Modelle $\texttt{Wort}_{0.5}$ und $\texttt{N-Gramm}_{\text{norm}}$. Auch Recall und Precision dieser beiden Modelle sind für Wertelisten identisch. Eine Verbesserung der Klassifikationsergebnisse durch die Unicode-Normalisierung tritt erst bei textbasierten Attributinstanzen auf. Hier sind Precision und F_1-Score etwas höher als bei dem N-Gramm-Modell ohne weitere Vorverarbeitung.

Die Datenerweiterung wirkt sich weiterhin hauptsächlich auf Recall und F_1-Score aus. Die beiden N-Gramm-Modelle mit Datenerweiterung erreichen sowohl für Wertelisten mit 0.6575 als auch für Texte mit 0.5367 die höchsten Recall-Werte. Bei textbasierten Attributinstanzen liegt der Unterschied mit über 0.1 allerdings deutlich höher als bei Wertelisten mit 0.02. Die Datenerweiterung führt folglich bei textbasierten Attributen zu einer deutlichen Verbesserung. Die Kombination der beiden Vorverarbeitungsschritte erreicht in Bezug auf Texte bei jeder Metrik die höchsten Werte. Allerdings führt die Datenerweiterung zu einer leichten Verschlechterung der Precision bei Wertelisten.

8.2.3 Ergebnisse als Empfehlungssystem mit mehreren Zielklassen

Neben der Auswertung der Klassifikationsergebnisse der primären Zuordnung, wird evaluiert, wie gut ALR mit den verschiedenen Modell-Konfigurationen die Rangfolge der Zuordnungen vorhersagt. Insbesondere bei Empfehlungssystemen zur Unter-

	LRAP Gesamt	
ALR Model	mit Ergebnis	Insgesamt
N-Gramm$_{0.5}$	**0.8567**	0.6009
N-Gramm$_{norm}$	0.8388	0.6057
N-Gramm$_{erw}$	0.8307	**0.6733**
N-Gramm$_{erw+norm}$	0.8348	0.6728
FT$_{erw+norm}$	0.8262	0.5382

Tabelle 8.10: LRAP-Werte bei der Vorhersage mehrerer Zuordnungen durch ALR. `Mit Ergebnis` ***zeigt den erreichten Wert, wenn ALR eine Vorhersage größer*** 0.0 ***getroffen hat.*** `Insgesamt` ***berücksichtigt auch die Fälle, in denen ALR keine Vorhersage treffen konnte.***

stützung manueller Aufgaben fördern richtige Empfehlungen das Vertrauen und vereinfachen den Arbeitsalltag.

Aus diesem Grund ist der LRAP (vgl. Abschnitt 3.6.3) eine geeignete Metrik. Hierbei wird die Rangfolge aller vorhergesagten Zuordnungen anhand der Konfidenz von ALR berücksichtigt und den Silber-Labels des Testdatensatzes gegenüber gestellt. Dabei werden die `N-Gramm`-Modelle mit dem einfachen FastText-Modell verglichen, da diese Modelle in den bisherigen Auswertungen die besten Ergebnisse erzielen konnten.

Die Ergebnisse in Tabelle 8.10 zeigen die erzielten LRAP-Werte ohne Berücksichtigung des Attributtyps. Hierbei werden die Ergebnisse in zwei Klassen unterteilt. In der ersten Ergebnisspalte der Tabelle werden für den LRAP nur Vorhersagen inkludiert, bei denen ALR mindestens eine Zuordnung erkannt hat. In der Spalte *Insgesamt* werden für den LRAP auch Vorhersagen eingeschlossen, bei denen kein Ergebnis durch ALR ermittelt werden konnte.

Erneut erreicht ALR mit dem Modell N-Gramm$_{0.5}$ den höchsten Wert (0.8567), wenn nur tatsächliche Vorhersagen, bei denen mindestens eine Zuordnung eine Konfidenz größer Null besitzt, berücksichtigt werden. In diesem Fall sorgen die Vorverarbeitungsschritte nicht für eine Verbesserung. Werden alle Vorhersagen einbezogen, auch wenn für diese keine eindeutige Rangfolge bekannt ist bzw. keine Zuordnungen abgeleitet werden konnten, zeigt die Datenerweiterung einen deutlichen Einfluss. Durch die Datenerweiterung wird ein Maximum von 0.6733 erreicht.

Die Differenz zwischen den beiden Klassen zeigt, dass ALR in vielen Fällen zwar eine Rangfolge bilden kann, aber trotzdem in einigen Fällen keine Vorhersage trifft.

	LRAP			
	Wertelisten		Text	
ALR Model	mit Ergebnis	Insgesamt	mit Ergebnis	Insgesamt
N-Gramm$_{0.5}$	**0.9725**	0.8046	**0.8077**	0.4215
N-Gramm$_{norm}$	**0.9725**	0.8046	0.7853	0.4305
N-Gramm$_{erw}$	0.9492	**0.8149**	0.7929	**0.5485**
N-Gramm$_{erw+norm}$	0.9492	**0.8149**	0.7979	0.5477
FT$_{erw+norm}$	0.9012	0.7402	0.7991	0.3603

Tabelle 8.11: LRAP-Werte bei der Vorhersage mehrerer Zuordnungen durch ALR, aufgeteilt nach Attributtyp. `Mit Ergebnis` ***zeigt den erreichten Wert, wenn ALR eine Vorhersage größer*** 0.0 ***getroffen hat.*** `Insgesamt` ***berücksichtigt auch die Fälle, in denen ALR keine Vorhersage treffen konnte.***

Ein Problem der LRAP-Metrik ist, dass in einigen Fällen durch die automatische Datengenerierung keine Referenz-Rangfolge gebildet werden kann. Dennoch zeigt Tabelle 8.10, dass über ALR relevante Empfehlungen erfolgen.

Bei der Gegenüberstellung der Ergebnisse nach Attributtyp in Tabelle 8.11 schneidet ALR unabhängig des gewählten Modells, wie zuvor, bei Wertelisten deutlich besser ab als bei textbasierten Attributinstanzen. Erneut erreicht N-Gramm$_{0.5}$ die höchsten Werte für Wertelisten und Texte, wenn nur die vorhandenen Vorhersagen berücksichtigt werden. Für Wertelisten sind die Ergebnisse identisch mit denen des Modells N-Gramm$_{norm}$. Hierdurch zeigt sich erneut, dass die Unicode-Normalisierung keinen Einfluss auf die Ergebnisse für Wertelisten hat.

Die Datenerweiterung führt wie bisher zu besseren Ergebnissen, wenn auch Vorhersagen ohne Zuordnung einbezogen werden. Die Erweiterung führt allerdings zu einer Verschlechterung des LRAP, wenn nur Vorhersagen mit Ergebnis berücksichtigt werden. Erneut kann das FastText-Modell nicht mit den N-Gramm-Modellen konkurrieren und schneidet in jeder Hinsicht schlechter ab.

Anhand Tabelle 8.11 ist ersichtlich, dass für Wertelisten ein hoher Automationsgrad erreicht werden kann, da Wertelisten häufig nur eine einzelne Zielzuordnung besitzen, wie auch bereits in Abschnitt 8.1 beschrieben wurde. Bei Texten hingegen zeigen sich deutlich schlechtere Ergebnisse, die unter anderem daraus resultieren, dass aus textbasierten Attributinstanzen durchschnittlich fünf Zuordnungen ermittelt werden müssen. Trotzdem zeigen die Ergebnisse, dass ALR in der Lage ist, Sach-

bearbeitern eine Hilfestellung zu bieten, auch wenn nicht alle Zuordnungen erkannt werden.

8.3 Einschränkungen in der Evaluation

In der Evaluation sind verschiedene Einschränkungen zu beachten. ALR ist so konzipiert, dass auch Produktdaten mit geringer Qualität verarbeitet werden können, was für den Umgang mit dem aus manuellen Integrationen gesammelten Datensatz erforderlich ist: Die Qualität variiert von fehlerfreien Produktdaten bis hin zu Attributinstanzen, bei denen Leerzeichen fehlen (z. B. *96test* statt *96 test*), Instanzen mit Rechtschreibfehlern oder sogar Instanzen mit Codierungsfehlern, willkürlichen Zeichenfolgen oder Zahlen (z. B. *™* statt ™, *ug* statt *µg*).

Trotzdem können sich diese Qualitätsprobleme negativ auf die vorgestellten Ergebnisse dieses Kapitels ausgewirkt haben, da der automatisiert abgeleiteten Silberstandard nicht validiert werden konnte. Dadurch können bei der Ähnlichkeitsbestimmung sowie dem Vokabular der verschiedenen ML-Verfahren OOV-Fehler aufgetreten sein. Diese OOV-Fehler können demensprechend ein Grund dafür sein, dass ALR keine Zuordnung vorhersagen kann.

Zusätzlich enthalten die einzelnen Spalten Duplikate, insbesondere Attributinstanzen, die durch eine Werteliste abgebildet werden können, wodurch für die Wertelisten ggf. eine Überanpassung (vgl. Abschnitt 3.4) im genutzten ML-Modell entstanden ist. Darüber hinaus enthält der Datensatz zwar eine große Anzahl von Attributen und Produkten, aber nur eine begrenzte Anzahl von verschiedenen Herstellern. Hier liegt ein weiterer Grund für eine mögliche Überanpassung.

Kapitel 9

Verwandte Arbeiten und Einordnung der Ergebnisse in den wissenschaftlichen Kontext

Dieses Kapitel bietet einen Überblick über verschiedene Verfahren und Systeme, die bereits in verwandten Problemstellungen, wie z. B. dem Schema-Matching für Datenbanken, eingesetzt wurden. Die im Folgenden vorgestellten Systeme stehen jeweils stellvertretend für eine bestimmte Lösungsart, beispielsweise heuristisch oder ML-basiert. Zusätzlich berücksichtigen die Systeme unterschiedliche Daten, wie Attributnamen oder Attributinstanzen. Bei der Vorstellung der Verfahren wird darauf eingegangen, inwiefern sie sich mit ALR oder den entwickelten ML-Verfahren auf Basis von Attributnamen vergleichen lassen. Abschließend werden die eigenen Verfahren und erzielten Ergebnisse in den wissenschaftlichen Kontext eingeordnet.

9.1 COMA/COMA

COMA (COmbination of MAtching algorithms) [46] ist ein Werkzeug für das Schema-Matching. COMA nutzt mehrere Matching-Verfahren parallel und orchestriert diese Verfahren eigenständig. Es besteht aus drei Teilen:

1. den Matching-Verfahren,
2. einem Framework, um diese zu kombinieren,
3. und einer Plattform, die der Evaluation der verschiedenen Verfahren dient.

Im Matching-Prozess werden zuerst verschiedene Verfahren angewandt, um Ähnlichkeitsmetriken zu bestimmen. Dazu gehören unter anderem die Ähnlichkeit von

N-Grammen, Pfadähnlichkeiten, Datentypnamen (beispielsweise String, Int usw.), sowie Synonymähnlichkeiten in einem Thesaurus. All diese Ähnlichkeiten werden in einem Ähnlichkeitswürfel, der alle Kombinationen von Attributnamen und den Ähnlichkeitsmetriken zwischen den Attributnamen von Eingangs- und Zielschema enthält, abgelegt. Im letzten Schritt werden die Ergebnisse aggregiert. Auch hier stehen mehrere Verfahren zu Verfügung.

Intern nutzt COMA einen gerichteten, azyklischen Graphen (directed acyclic graph (DAG)), um Schemata darzustellen. Dabei entspricht ein einzelner Knoten einem Attributnamen. Enthält ein Schema, wie bei XML-Dateien üblich, hierarchische Attribute, bei denen ein Attribut aus mehreren weiteren Attributen gebildet wird, werden diese durch Verbindungen bzw. Kanten zum übergeordneten Attribut dargestellt. Bei tabellarischen Strukturen mit gleichwertigen Attributen, wie bei der Katalogintegration, wird ein virtueller Wurzelknoten eingeführt. Dadurch soll der Kontext der einzelnen Elemente im Schema berücksichtigt werden. Dabei wird auch deutlich, dass COMA vor allem darauf abzielt, Zuordnungen zwischen Schemata aus strukturreichen Formaten zu ermitteln, wie XML oder OWL. Bei tabellarischen Schemata muss jedoch auf Strukturinformationen verzichtet werden, wodurch in erster Linie ein Matching auf Basis von Zeichenketten erfolgt.

COMA wurde bis 2013 in unregelmäßigen Abständen erweitert [74, 20, 75]. Das Ziel des Schema-Matchings wurde um die Zusammenführung zweier Schemata, z. B. Datenbank- und XML-Schemata, ausgeweitet. Des Weiteren können Nutzer Zuordnungen über eine grafische Oberfläche erkennen und bei Bedarf korrigieren. Auch COMA setzt dafür verstärkt auf Struktureigenschaften und Datentypinformationen.

In der Evaluation in Kapitel 6 konnte COMA keine besseren Ergebnisse erzielen als ein ML-Verfahren, obwohl die genutzten Ähnlichkeitsmetriken, aus denen ein ML-Verfahren die Zuordnungen lernen musste, denen von COMA sehr ähneln. Daher ist davon auszugehen, dass die ML-Verfahren Zusammenhänge und Gewichte der verschiedenen Ähnlichkeitsmetriken besser lernen konnten als COMA durch Heuristiken und Aggregationen aus dem Ähnlichkeitswürfel. Eine Ursache könnte die Abhängigkeit COMAs von Strukturinformationen sein, die nicht vorhanden waren.

Ein großer Vorteil der ML-Verfahren zeigt sich in der Entwicklung. So mussten keine Heuristiken implementiert werden, sondern es konnten bereits implementierte ML-Verfahren genutzt werden, die anhand der Feature-Vektoren Zuordnungen bestimmen konnten.

9.2 ProbaMap

ProbaMap ist ein automatisches System, das Beziehungen zwischen den Klassen zweier Taxonomien über Wahrscheinlichkeiten ermittelt [88]. In der Katalogintegration würden die Klassen den Attributnamen entsprechen, wobei Beziehungen der Produktattribute im Gegensatz zu Klassen allgemeiner Taxonomien nicht hierarchisch abgebildet werden können.

Wie COMA nutzt auch ProbaMap einen DAG als Grundlage, um die wahrscheinlichsten Zuordnungen der Klassen beider Taxonomien über Klassifikationsverfahren zu bestimmen. Die Berechnung der Wahrscheinlichkeiten erfolgt anhand von Instanzbeschreibungen, die bei der Katalogintegration den Attributinstanzen entsprechen. ProbaMap nutzt dazu einen Generierungs- und Testalogrithmus, der die Anzahl der Aufrufe eines Wahrscheinlichkeitsschätzers minimiert. Dabei werden die Zuordnungen, deren Wahrscheinlichkeit größer als ein gegebener Grenzwert ist, ermittelt. Tournaire et al. nutzen verschiedene Klassifikationsverfahren, um die Wahrscheinlichkeiten zu schätzen.

In ihren Experimenten evaluieren Tournaire et al. ProbaMap mit verschiedenen Klassifikationsverfahren wie SVMs und Entscheidungsbäumen und anhand mehrerer Taxonomiebeispiele, u.a. Taxonomien von Google und Yahoo. Dabei ist die Accuracy mit unter 0.4 zwar eher gering, aber die Experimente zeigen, dass ML-Ansätze in der Lage sind Zuordnungen zwischen Taxonomien zu bestimmen. Für die Katalogintegration ist das Vorgehen jedoch nur begrenzt anwendbar, da die Datenintegration hauptsächlich über ein flaches tabellarisches Schema erfolgt und nicht über Taxonomien. Dies führt zu vergleichbaren Problemen wie bei COMA.

9.3 Corpus Based Matching

Madhavan et al. [89] haben einen Ansatz für das Matching relationaler Schemata entwickelt, der neben Informationen aus den betrachteten Schemata auch domänenspezifisches Wissen über einen Schema-Korpus bzw. bereits bekannte Zuordnungen nutzt. Der Schema-Korpus ist im Anwendungsfall dieser Ausarbeitung vergleichbar mit den jeweiligen Trainingsdatensätzen.

Corpus Based Matching versucht die Zuordnungen anhand von Ähnlichkeiten zwischen Attributnamen und -instanzen aus der Analyse von großen Textdatensätzen zu erreichen. Ein solcher Datensatz kann zum Beispiel auf eine Domäne angepasst werden, indem bereits bekannte Schemata in den Korpus abgelegt werden. Dazu

gehören auch standardisierte und allgemein verfügbare Schemata. Mit jedem neuen Matching-Vorgang kann der Datensatz erweitert werden und gewinnt so stetig an Größe. Letztendlich ist das Ziel, dass jedes Konzept – hier ein Zielattributname – in einer Domäne eine Repräsentation erhält, die zusätzlich Varianten des Konzeptes abdeckt. Der Thesaurus und die Datenerweiterung in Kapitel 5 verfolgen ebenfalls dieses Ziel.

Der Korpus wird auf zwei Weisen genutzt: Erstens werden Kandidaten für eine Zuordnung über bereits bekannte ähnliche Zuordnungen aus dem Korpus gesucht. Zweitens werden ähnliche Attributnamen und -instanzen aus dem Korpus mittels Clustering-Verfahren gruppiert. Anhand dieser Statistiken werden Randbedingungen ermittelt, die die Auswahl der Zuordnungen einschränkt, sodass die Zuordnungen zwischen den Schemata ermittelt werden können. Der erste Schritt entspricht dem pragmatischen Ansatz für die Zuordnungsverfahren durch Attributnamen.

Corpus Based Matching ermöglicht insbesondere das Finden von $1:1$ Zuordnungen. Die Berechnung der besten Zuordnung erfolgt wiederum in zwei Phasen. Zuerst werden Eingangs- und Zielschema jeweils mit dem Korpus verglichen, wobei einzelne Attributnamen durch Variationen aus dem Korpus ersetzt werden. Danach werden die Zuordnungen zwischen allen so entstandenen Variationen ermittelt. Dabei werden in beiden Phasen die gleichen spezifischen Matching-Strategien mit verschiedenen Lernverfahren verwendet:

1. ein Namenslerner,
2. ein Textlerner,
3. ein Instanzlerner
4. und ein Kontextlerner.

Die Lernverfahren benutzen Techniken wie Tokenisierung und N-Gramme, um aus dem Korpus Trainingsdaten zu erstellen, während für die Zuordnung ein übliches Klassifikationsverfahren, wie Naive Bayes, genutzt wird. Zusätzlich können weitere Informationen, wie die Editier-Distanz (vgl. Abschnitt 4.3.1) oder das Auftreten bestimmter Muster im Kontextlerner genutzt werden, um eine bessere Zuordnung zu ermöglichen. Diese Lernansätze werden im *Corpus Based Matching* über eine logistische Regression als Metalerner kombiniert. Die so bestimmten Kandidaten werden schlussendlich mit den Randbedingungen aus den Korpus-Statistiken abgeglichen, um die finale Zuordnung zu bestimmen.

In Bezug auf die Katalogintegration sind insbesondere die folgenden Aspekte interessant:

1. Es wird ein Korpus über bekannte Zuordnungen aufgebaut (Trainingsdatensatz).
2. Aus diesem Korpus heraus können verschiedene Lernverfahren trainiert werden, um Zuordnungen bzw. Zugehörigkeiten zu bestimmen.
3. Verschiedene Eigenschaften (N-Gramme, Distanzen, Token) werden als Ausgangspunkt für Merkmale, die Konzepte charakterisieren, genutzt.

Zusätzlich wurde das *Corpus Based Matching* explizit mit dem Ziel entwickelt, 1:1 Beziehungen relationaler Schemata zu erkennen. Diese Eigenschaften sind grundsätzlich mit denen von Produktkatalogen vergleichbar, da tabellarische Schemata und häufig auch 1:1 Beziehungen zwischen den Attributen vorliegen.

Im Unterschied zur Katalogintegration sind hier jedoch alle Attributnamen und Attributinstanzen der Eingangs- und Zielschemata zu jedem Zeitpunkt bekannt, während bei der Katalogintegration zur Laufzeit nicht auf die Attributinstanzen des Zielschemas zugegriffen werden kann, sondern nur zur Trainingszeit. Zusätzlich muss bei der Katalogintegration bestimmt werden, welchem Zielattribut die Instanzen des Eingangsschemas zugeordnet werden.

9.4 ASMOV

ASMOV (Automatic Semantic Matching of Ontologies with Verification) ist ein Ansatz für das automatische Matching von Ontologien, das auf Daten bzw. Informationsintegration in der Bioinformatik spezialisiert ist [90]. Der Ansatz besteht zusammengefasst aus zwei Schritten. Zuerst werden Ähnlichkeiten berechnet und danach werden die Ergebnisse semantisch verifiziert.

Dazu nutzt der Algorithmus zwei OWL-Ontologien als Eingangsdaten, sowie optionale initiale Zuordnungen zwischen den beiden Ontologien. Das Resultat sind dann $N : M$ Zuordnungen. Der erste Schritt besteht, wie in einigen anderen Ansätzen auch, aus lexikalischen und strukturellen Matching-Verfahren (siehe z.B. Abschnitt 4.3.1) auf Basis der Zeichenketten, die das Schema der Ontologien beschreiben. Diese Verfahren werden genutzt, um einen gewichteten Mittelwert aller Ähnlichkeitsmaße zwischen den Elementen der beiden Ontologien zu berechnen. Zusätzlich können Ähnlichkeiten anhand von externen Ressourcen wie WordNet (vgl.

Abschnitt 4.3.3) oder vergleichbaren domänenspezifischen Quellen berechnet werden.

Die ermittelten Ähnlichkeiten werden als Kandidaten für die Zuordnung genutzt, je höher die Ähnlichkeit, desto wahrscheinlicher ist die Zuordnung. Um die finalen Zuordnungen zu bestimmen, müssen diese allerdings semantisch verifiziert werden. Dazu nutzt das System ein Ausschlussverfahren. Es erkennt Muster, die bewiesen zu Inkonsistenzen führen, wie Widersprüche oder Unvollständigkeit. Diese Schritte werden mit den resultierenden Zuordnungen als neue Initialzuordnung solange wiederholt, bis keine neuen zusammengehörigen Elemente mehr gefunden werden.

Zusammengefasst nutzt ASMOV mehrere Matching-Verfahren, um Ähnlichkeiten zwischen zwei Ontologien im OWL-Format zu bestimmen. Für die Katalogintegration besteht auch bei diesem Ansatz die Einschränkung, dass keine Struktureigenschaften der Eingangsdaten genutzt werden können. Zusätzlich können Ähnlichkeiten nur auf Ebene der Attributnamen genutzt werden, da zur Laufzeit nur im Zielschema Instanzen vorhanden sind, während im Zielschema nur die Attributnamen bekannt sind. In Kapitel 5 analog zu ASMOV die Kombination mehrerer Ähnlichkeitsmetriken als einen möglicher Ansatz verfolgt.

9.5 CIDER

CIDER (Context and Inference baseD alignER, [91]) ist ein ML-basiertes System zur Ermittlung von Zuordnungen zwischen zwei OWL-Ontologien. Als Erstes wird für jeden Term (Attributname) aus der Ontologie der Kontext extrahiert. Dies geschieht über die Auflösung von Synonymen, Hyperonymen, Hyponymen, Eigenschaften, usw. Zusätzlich können bei CIDER externe Wissensquellen wie WordNet genutzt werden. Dies dient vor allem der Auflösung von Mehrdeutigkeiten der zuzuordnenden Terme der Ontologien.

Danach werden, wie bei ASMOV, mehrere Ähnlichkeitsmaße genutzt, um die Ähnlichkeiten zwischen Attributnamen bzw. Termen zu bestimmen. Zur Berechnung der strukturellen Ähnlichkeit wird über den Kontext ein Vektorraummodell der Kontextmerkmale gebildet. All diese Ähnlichkeiten werden als Eingangswerte für ein NN, genauer ein MLP, genutzt, das Eigenständig die Gewichtung der Merkmale erlernt. Letztendlich werden die $1:1$ Zuordnungen über einen Grenzwert bestimmt.

Aus Sicht der Katalogintegration ist CIDER vor allem durch die Erweiterung von ASMOV durch die Anwendung eines NN auf die Kombination der Ähnlichkeitsme-

triken interessant, da die Zuordnungen zum Zeitpunkt des Trainings bekannt sind. Allerdings kann CIDER nur $1:1$ Zuordnungen erkennen, was zu Einschränkungen in der Anwendbarkeit führt.

9.6 Schema-Matching mit Wortvektoren

Nozaki et al. [92] versuchen Zuordnungen zwischen zwei tabellarischen Schemata anhand von Wortvektoren (siehe Abschnitt 5.4.2) zu bestimmen. Die Voraussetzung dafür ist, dass Attributinstanzen durch Worte oder Zeichenketten beschrieben werden. Da Attributnamen nicht berücksichtigt werden, handelt es sich um einen instanzbasierten Ansatz.

Die Grundidee basiert auf der Verwendung von bereits gelernten Word2Vec-Vektoren (siehe Abschnitt 4.4.3), die semantische Ähnlichkeiten abbilden können. Nach der Zerlegung der Attributinstanzen in Zeichenketten besteht die Vektorberechnung aus den folgenden Schritten:

1. Tokenisierung: die Zerlegung der Attributinstanzen in einzelne Worte.
2. Aufsummierung der einzelnen Wortvektoren, wobei ein Wortvektor im OOV-Fall gleich null ist.
3. Falls eine Instanz nur Token enthält, die nicht im Vokabular enthalten sind, ist der resultierende Vektor gleich null.
4. Der Ergebnisvektor wird anhand der relativen Häufigkeit der Attributinstanz gewichtet.

Dadurch erhält jedes Attribut einen Vektor, der alle Instanzen berücksichtigt. Abschließend werden die Zuordnungen von Quell- und Zielschema über die Kosinus-Ähnlichkeit (vgl. Abschnitt 4.4.5) ermittelt, indem ein Schwellwert angibt, ab wann eine Zuordnung vorliegt. Ist die Ähnlichkeit größer als der Schwellwert, liegt dementsprechend eine Zuordnung vor.

In diesem Schwellwert liegt allerdings ein großer Nachteil des Verfahrens, da nicht klar ist, ab wann in einer realen Anwendung eine Ähnlichkeit ausreichend ist. Nozaki et al. experimentieren mit verschiedenen Schwellwerten, die folglich zu unterschiedlichen Ergebnissen führen, aber aus fachlicher Sicht trotzdem vertretbar sind. Es ist somit nicht sicher, welche Ergebnisse als richtig oder falsch zu werten sind, solange die Zuordnungen aus semantischer Sicht noch sinnvoll sind.

Dadurch sind die Ergebnisse für $1 : M$ Beziehungen zwischen Quell- und Zielschema nicht eindeutig und nicht automatisierbar. Bei $1 : 1$ Beziehungen sind jedoch eindeutige Ergebnisse möglich. Eine weitere Schwäche sind OOV-Fehler, die insbesondere bei Katalogdaten häufig auftreten können, wie z. B. bei Identifikationsnummern. Somit werden Worte oder Token, die dem Modell unbekannt sind verworfen, obwohl wichtige Informationen enthalten sein können.

9.7 Zuordnung von Schema-Attributnamen durch Inhaltsanalyse

Der Ansatz von Chen et al. [14] ermöglicht die Zuordnung von Attributnamen durch die Analyse von Attributinstanzen anhand tabellarischer Datensätzen. Aus verschiedenen Tabellen werden anhand von Attributinstanzen Merkmale extrahiert und mit Hilfe von Random Forests[1] (vgl. Abschnitt 3.5.4) gelernt. Die gelernte Funktion soll das zugehörige Zielattribut aus einer Menge von Zielattributen mit der höchsten Wahrscheinlichkeit bestimmen.

Daher wird die Vorhersage der Schema-Attributnamen als Mehrklassen-Klassifikation betrachtet, wobei genau ein Zielattributname vorhergesagt werden soll. Jeder Attributname aus dem Trainingsdatensatz wird als Zielklasse genutzt. Als Merkmale für das Modelltraining werden ausschließlich Instanzmerkmale von Tabellenspalten verwendet. Darunter fallen unter anderem Minimal- und Maximalwerte von numerischen Spalten, Unigramme und prozentual enthaltene Buchstaben, Ziffern oder Symbole. Diese Merkmale wurden teilweise in ALR genutzt. Im Gegensatz zu Schema-Matching Ansätzen wird nicht auf Ähnlichkeiten von Zeichenketten oder Bedeutungen gesetzt, da ausschließlich Attributnamen von gelernten Attributinstanzen vorhergesagt werden.

Dieser Ansatz lässt sich auf Katalogdaten übertragen, da ebenfalls tabellarische Eingangsdaten vorliegen. Die Zielklassen müssten in einem solchen Fall lediglich an das Zielschema angepasst werden. Dies könnte beispielsweise erreicht werden, indem bereits integrierte Kataloge gelernt werden und das Modell auf neue Produktkataloge angewandt wird. Da Chen et al. eine exklusive Mehrklassen-Klassifikation nutzen, lässt sich jeder Spalte nur genau ein Attributname zuordnen. Dadurch können nur $1 : 1$ Beziehungen gebildet werden. Des Weiteren funktioniert der gewählte Ansatz auf Spalten mit Ganzzahlen und Gleitkommazahlen erheblich besser, als

[1] sklearn.ensemble.RandomForestClassifier

wenn Spalten nur Zeichenketten enthalten. Folglich müssten für die Anwendung auf die Katalogintegration bessere textuelle Merkmale bzw. Wortvektoren genutzt werden.

9.8 Erstellen von Vektoren auf relationale Datensätzen für die Datenintegration

Cappuzzo et al. [93] versuchen aus tabellarischen Datensätzen eine Vektortransformation zu generieren, indem sie ein dreistufiges Verfahren nutzen. Zuerst werden die tabellarischen Datensätze in einen kompakten Graphen umgewandelt, der Attributnamen, Attributinstanzen und Zeilennummern als Knoten miteinander verknüpft. Ausgehend von diesem Graphen werden aus den einzelnen aufeinanderfolgenden Knoten Sätze generiert, indem die Knoten zufällig durchlaufen werden.

Dabei werden die Token der Attributinstanzen miteinander über Leerzeichen verbunden werden. Diese generierten Sätze werden als Trainingsdatensatz für eine Wortvektortransformation, wie z.B. FastText, genutzt. Auf diese Weise können Instanzvektoren trainiert werden, die den gesamten Kontext der tabellarischen Produktkataloge berücksichtigen.

Als Beispielanwendung wird auch das Schema-Matching genannt. Hier verwendeten die Autoren ebenfalls die Kosinusähnlichkeit der erzeugten Vektoren. Mit diesem Vorgehen erreichten sie vielversprechende Ergebnisse bei der Ermittlung von Zuordnungen zwischen den Schemata verschiedener Unternehmen. So wurden beispielsweise Zuordnungen zwischen Schemata von Amazon und Google oder Amazon und iTunes bestimmt.

9.9 Einordnung der Ergebnisse in den wissenschaftlichen Kontext

In den vorangegangenen Abschnitten wurden verschiedene Ansätze für das Schema- und Ontologie-Matching vorgestellt. Diese Verfahren decken verschiedene Zeiträume der Entwicklung ab. Von ca. 2002 (COMA) bis ca. 2013 (siehe auch [94]; CIDER, 2011) wurden im Schema-Matching hauptsächlich Verfahren entwickelt, die Ähnlichkeitsmetriken nutzen, um Zuordnungen anhand dieser zu bestimmen. Dabei wurden die Zuordnungen entweder durch geschickte Kombinationen von Me-

triken berechnet (COMA) oder erlernt. Gleichzeitig fokussiert sich der Mehrheit der Verfahren auf die Vorhersage von $1:1$ Beziehungen.

Daher wurden in Kapitel 5 vergleichbare Ansätze verfolgt. Darunter fällt der pragmatische Referenzansatz als starke Vereinfachung von [89] und die Klassifikation über Ähnlichkeitsmetriken als Feature-Vektor mit verschiedenen ML-Verfahren. Den vorherigen Abschnitten zu entnehmen ist, wurden ML-Verfahren bereits vielfältig genutzt, um die Ähnlichkeit von Attributnamen zu bestimmen. Im Gegensatz dazu wurden in dieser Dissertation die Ähnlichkeiten direkt als Feature-Vektor genutzt, um die Zuordnungen vorherzusagen.

Zwischen 2014 und 2018 finden sich nur vereinzelte Publikationen aus dem bereich den Schema-Matchings oder verwandeten Gebieten. Beispielsweise versuchen Zhang et al. [95] durch unbekannte Menschen als Dienstleister (Crowdsourcing) Zuordnungen bestimmen zu lassen und Anam et al. [96] verfolgen den Ansatz Ontologie-Matching über einen Wissensgraphen durchzuführen. Auch Publikationen nach 2018 versuchen direkt aus Wortvektoren die Ähnlichkeit abzuleiten und als Grundlage für die Zuordnung zu verwenden [92].

Die Verwendung von Wortvektoren als Feature-Vektoren zur Klassifikation von Zuordnungen wurde außerhalb dieser Dissertation dagegen noch nicht erprobt. Hier bieten die Lernansätze mit OH-, TF-IDF- und FastText-Vektoren neue Erkenntnisse. Zusätzlich wurde auf die Datenerweiterung – analog zu der Idee aus [89] – zurückgegriffen. Das Verwenden von Wortvektoren und die Kombination mit Datenerweiterungstechniken stellt insofern eine wesentliche Neuerung dar, auch wenn die Verfahren grundsätzlich bereits bekannt waren. Es ist anzunehmen, dass sich die erzielten Ergebnisse aus Kapitel 6 auch auf Anwendungsfälle außerhalb des Antikörpermarkts übertragen lassen.

Das in Kapitel 7 entwickelte ALR-Verfahren kombiniert verschiedene vorgestellte Ansätze. So bildet der Ansatz zur Vorhersage von Attributnamen anhand von Attributinstanzen von Chen et al. [14] (siehe Abschnitt 9.7) die Grundlage des Instanz-NN in ALR. Die Vorhersage erfolgt bei Chen et al. im Gegensatz zu ALR allerdings als exklusive Klassifikation, sodass nur $1:1$ und $N:1$ Beziehungen vorhergesagt werden können.

Wie beschrieben, wurden die Eigenschaften der Attributinstanzen von Chen et al. in den Feature-Vektor übernommen. Dieser wurde zusätzlich um die bisher fehlenden Wortvektoren erweitert, da der Ansatz von Chen et al. hauptsächlich bei Wertelisten und nur eingeschränkt für textbasierte Attribute funktioniert. Insbesondere

für Produktkataloge ermöglicht diese Erweiterung bessere Vorhersagen, da in Produktkatalogen viele textbasierte Attribute existieren.

Weiterhin wurden in ALR die OH-Vektoren wegen der guten Ergebnisse beim Lernen der Attributnamen erneut verwendet. Ebenso wird statt dem Random Forests in [14] ein multimodales NN genutzt, um Attributnamen und Attributinstanzen zunächst unabhängig zu betrachten. Diese werden erst in einer tieferen Ebene des NN wieder zusammengeführt. Zusätzlich wurden die Vorhersagen im zweiten Schritt von ALR aggregiert. Hier wurde die Aggregationsidee von COMA wiederverwendet. Insgesamt konnte so ein neues, effektives Verfahren entwickelt werden.

Zwar wurden für die beiden Evaluationen von ALR und Chen et al. [14] verschiedene Datensätze genutzt, grundsätzlich sind die Konfigurationen und Szenarien allerdings vergleichbar. Chen et al. nutzen jedoch, wie beschrieben, einen exklusiven Klassifikationsansatz und können nur genau ein Zielattribut vorhersagen. Dadurch können bei ALR nur die Klassifikationsergebnisse für die primäre Zuordnung mit den Ergebnissen von Chen et al. verglichen werden. Dabei sticht insbesondere die Verbesserung der Zuordnung von textbasierten Attributen hervor:

Während Chen et al. je nach genutztem Feature-Vektor einen F_1-Score von 0.20 bis 0.23 für textbasierte Attribute angeben, erreicht ALR bei den hier vorgestellten Szenarien einen um mindestens 0.5 höheren Wert. Dadurch zeigt sich zum einen die Effektivität der Subtoken N-Gramme als Wortvektor im Vergleich zu einzelnen Buchstaben im Feature-Vektor. Zusätzlich können einzelne falsch vorhergesagte primäre Zuordnungen durch die nachfolgende Aggregation bei ALR ausgeglichen werden.

Andere Verfahren, wie zum Beispiel Anam et al. [96] oder Pomp et al. [97] verwenden Ähnlichkeitsmaße um ähnliche Konzepte bzw. Attributnamen zu erkennen. Im Gegensatz zu dem zuvor genannten Verfahren wird eine Analyse der Häufigkeitsverteilung der Attributinstanzen pro Spalte durchgeführt, um zu ermitteln, welche Attribute eine ähnliche Häufigkeitsverteilung besitzen.

Dieses Vorgehen lässt sich allerdings nicht auf die Katalogintegration übertragen, da bereits einfache Unterschiede zwischen den Zulieferern, wie beispielsweise andere Wirte für die Antikörperherstellung bereits zu anderen Verteilungen der Attributinstanzen führen. Trotzdem zeigt auch dieser Ansatz, dass die Kombination von Instanz- und Namensinformationen zu besseren Ergebnissen bei Zuordnungen führt. Insbesondere gilt dies sowohl bei Pomp et al. als auch in dieser Ausarbeitung für numerische Spalten und Attribute mit Wertelisten. Die Zuordnungsergebnisse für textbasierte Attribute sind jedoch weiterhin schlechter als die der Wertelisten.

Obwohl die Katalogintegration und der für die Auswertung von ALR genutzte Datensatz nur eingeschränkt mit den jeweiligen Datensätzen der anderen Autoren vergleichbar sind, lassen sich die Ergebnisse trotzdem einordnen. Die Ergebnisse von ALR bezüglich MRR und LRAP deuten darauf hin, dass ALR in einem ähnlichen Bereich liegt wie die MRR-Werte von Pomp et al. [97].

In ihrer Evaluation geben Pomp et al. eine maximale MRR für Wertelisten von 0.783 auf einem Fußballdatensatz und 0.764 auf einem Museumsdatensatz an. ALR erreicht einen maximalen Wert von 0.8393. Bei den textbasierten Attributen variiert ihr Ansatz je nach Datensatz von 0.783 (Museum) bis zu 0.983 (Fußball). Der Mueseumsdatensatz enthält allerdings deutlich komplexere Attributinstanzen als der Fußballdatensatz. Insofern kann der Museumsdatensatz eher mit den hier verwendeten Datensätzen aus dem Antikörpermarkt vergliechen werden. Unter Berücksichtigung der Komplexität der Datensätze erzielt ALR ähnlich gute Ergebnisse mit einer MRR von 0.8137.

Kapitel 10

Schlussbemerkungen

In diesem Kapitel werden die Ergebnisse der Dissertation in Bezug auf die aufgeworfenen Forschungsfragen zusammengefasst und beantwortet. Bei der Beantwortung der FF wird gleichzeitig auf die Anwendbarkeit für KMU eingegangen. Abschließend folgt ein Ausblick, in dem offene Themen für künftige Arbeiten diskutiert werden.

10.1 Zusammenfassung

Im schnelllebigen Umfeld des E-Commerce ist die automatische Integration von Produktkatalogen in Online-Marktplätze eine wirtschaftliche Notwendigkeit. Die Integration der Produktkataloge erfolgt üblicherweise über ETL-Prozesse, die aus verschiedenen Teilschritten bestehen. Ein wichtiger Teilschritt, der in dieser Dissertation untersucht wurde, ist das Ermitteln von Zuordnungen zwischen den Eingangsschemata und dem Zielschema. Diese Zuordnung erfolgt häufig manuell.

Die Schemazuordnung für Produktkataloge bringt, wie in Kapitel 2 beschrieben, verschiedene Herausforderungen mit sich. Dazu gehören die Identifizierung identischer Konzepte in Eingangs- und Zielschema sowie die Notwendigkeit zur Aufteilung von Attributinstanzen von Eingangsattributen auf mehrere Attribute des Zielschemas. Ebenso bestehen Produktkataloge oft aus einfachen Tabellen, die keine Informationen über die Beziehungen zwischen den Attributen innerhalb eines Katalogs liefern. Darüber hinaus müssen unterschiedliche Typen der Attributinstanzen, wie Identifikationsnummern, Wertelisten oder Texte, im Kontext der jeweiligen Attributnamen unterschieden werden.

Jede manuelle Katalogintegration erzeugt Paare von Produktkatalogen vor und nach dem Integrationsprozess. Bisher wurden diese Daten nicht genutzt, obwohl

sie das Potenzial besitzen, die Schemazuordnung zu vereinfachen. Ziel dieser Arbeit war daher zu betrachten, inwieweit die im Rahmen eines Integrationsprozesses anfallenden Daten genutzt werden können, um das Bestimmen der Zuordnungen zu automatisieren oder zumindest die manuelle Zuordnung durch ein Empfehlungssystem zu unterstützen. Dafür wurde die folgende Leitfrage untersucht:

> Inwieweit können bisher nicht oder wenig genutzte Daten aus bereits durchgeführten Katalogintegrationen verwertet werden, um den höchstmöglichen Automatisierungsgrad im Teilprozess der Schemazuordnung insbesondere für KMU zu erreichen?

Zwei grundlegende Ansätze wurden verfolgt, um die weiteren Forschungsfragen zu beantworten. Zunächst wurden ausschließlich die Attributnamen der Schemata für die Schemazuordnung verwendet. Dazu wurden mehrere ML-Verfahren in Kombination mit verschiedenen Vektorisierungsstrategien für Attributnamen mit Verfahren verglichen, die ohne ML auskommen. Aufbauend auf den dabei gewonnenen Erkenntnissen wurde dann mit ALR ein neues Verfahren entwickelt, das neben Attributnamen auch Attributinstanzen zum Lernen von Zuordnungen verwendet. Die Evaluation der Verfahren erfolgte jeweils anhand von Produktdaten aus Integrationsprozessen eines Online-Marktplatzes für Antikörper.

Aus der Leitfrage wurden drei aufeinander aufbauende Forschungsfragen abgeleitet, auf deren Ergebnisse im Folgenden detailliert eingegangen wird. Danach wird durch die gemeinsame Betrachtung der einzelnen Antworten die gestellte Leitfrage beantwortet.

> **FF1:** Wie lässt sich ausschließlich mit Informationen aus Attributnamen und vorhandenen Thesauri der höchste Automatisierungsgrad bei der Schemazuordnung erreichen?

FF1 Um diese Frage zu beantworten, wurden in Kapitel 6 mehrere Verfahren, von einfachen Algorithmen über COMA bis zu mehreren ML-basierten Verfahren verglichen. Die ML-basierten Verfahren nutzen einerseits verschiedene Vektorisierungsverfahren, wie Ähnlichkeitsmetriken oder TF-IDF, und andererseits unterscheiden sich die zugrundeliegenden Klassifikationsalgorithmen. Die Evaluation bestand aus vier Szenarien, die sich in der Datenmenge und deren Vorverarbeitung unterschieden.

Lernende Verfahren schnitten in den meisten Fällen deutlich besser ab als einfache Algorithmen oder COMA. Lediglich der pragmatische Ansatz, der einen Attributnamen mit allen bekannten Zuordnungen über einen Thesaurus vergleicht und den ähnlichsten Eingangsattributnamen zur Zuordnung wählt, konnte vergleichbare Ergebnisse erzielen. Dennoch besitzen die ML-Ansätze Effizienzvorteile, da die Laufzeit des pragmatischen Ansatzes bei steigender Anzahl von Elementen im Thesaurus deutlich ansteigt, während die Antwortzeit der ML-Modelle konstant bleibt. Hier erhöht sich lediglich die Trainingszeit, die bei der Vorhersage von unbekannten Zuordnungen keine Rolle spielt. Insbesondere sorgt die geringe Antwortzeit für weniger Wartezeiten beim Einsatz als Empfehlungssystem.

Grundsätzlich eigenen sich alle betrachteten ML-Ansätze. Besonders gut schnitten die Kombinationen von N-Grammen als OH- und TF-IDF-Vektoren mit einem MLP oder einer SVM ab. Auch die Ergebnisse auf einem später erhobenen Testdatensatz im Praxiseinsatz konnten überzeugen. Zwar wurde keine menschliche Genauigkeit erreicht, allerdings reicht die erzielte Top-3-Genauigkeit von bis zu 0.90 aus, um als Empfehlungssystem einen Mehrwert zu bieten.

Ein weiteres Ergebnis ist, dass FastText-Vektoren nicht zu besseren Werten in der Evaluation führten als OH- und TF-IDF-Vektoren. Dies ist insofern interessant, als dass FastText bei Klassifikationsaufgaben auf natürlichsprachlichen Texten deutlich bessere Ergebnisse liefert als OH- und TF-IDF-Vektoren [70]. Eine mögliche Ursache für dieses Ergebnis ist, dass die Attributnamen im verwendeten vortrainierten FastText-Modell schlecht repräsentiert werden, da die Namensgebung sehr spezifisch für den Antikörpermarkt ist. Auch Allweyer et al. [98] machten zeitgleich mit [S5] ähnliche Erfahrungen mit Produktdaten.

FF2: Inwiefern wird die Schemazuordnung durch Informationen aus Attributinstanzen, die zusätzlich zu Attributnamen verwendet werden, verbessert?

FF2 Mit ALR wurde ein Verfahren entwickelt, das neben Attributnamen auch Attributinstanzen in tabellarischen Produktkatalogen für die Vorhersage von Zuordnungen nutzt. Zur Beantwortung von FF2 können die Ergebnisse aus Kapitel 6 und Kapitel 8 miteinander verglichen werden, auch wenn die Szenarien aufgrund der Datenverfügbarkeit nicht identisch aufgebaut werden konnten. Dennoch lässt sich der Einfluss der Instanzinformationen erkennen, nicht jedoch genau beziffern. Zudem lassen sich mit ALR zusätzlich $1:M$ Beziehungen vorhersagen, was anhand von Attributnamen nicht möglich war.

Die Evaluationen zeigen, dass die zusätzlichen Instanzinformationen bei der Bestimmung der primären Zuordnung grundsätzlich zu höheren Precision-Werten führen, aber gleichzeitig niedrigere Werte für Recall und F_1-Score erzielen. Höhere Precision-Werte sind jedoch für die Automatisierung hilfreich, da Vorhersagen von ALR sehr treffsicher sind, auch wenn nicht für alle Spalten aus dem Eingangskatalog Zuordnungen gefunden werden können. Auch bei ALR zeigte sich bei den primären Zuordnungen, dass einfache OH-Vektoren mit N-Grammen bessere Ergebnisse erzielen als die FastText-Vektoren.

Die Ergebnisse für $1 : M$ Kardinalitäten anhand der AUC zeigen, dass ALR auch mehrere relevante Zuordnungen erkennen kann. So wurde für einige Zuordnungen eine AUC von 0.98 bis 1.0 erreicht. Hierdurch hat sich herausgestellt, dass ALR in der Lage ist, an vielen Stellen nahezu perfekt zwischen den Merkmalen zu unterscheiden, wann eine Zuordnung erfolgen muss.

FF3: Wie wirkt sich der Typ der Attributinstanzen auf die Qualität der Schemazuordnung und damit auf die Möglichkeit einer vollständigen Automatisierung aus?

FF3 Anhand von Kapitel 8 lässt sich ebenfalls erkennen, welcher Attributtyp besonders für einen hohen Automatisierungsgrad geeignet ist. Den größten Einfluss hatte ALR in der Evaluation auf Attribute, die aus Wertelisten bestehen. Hier wurden mit bis zu 0.97 deutlich höhere Precision-Werte für die primäre Zuordnung erreicht als bei den attributnamenbasierten Verfahren. Allerdings fielen gleichzeitig Recall und F_1-Score geringer aus. Wie beschrieben, ist die höhere Precision allerdings gewünscht, um Zuordnungen vollständig automatisiert zu treffen.

Auch bei textbasierten Attributinstanzen war die Precision von ALR mit bis zu 0.90 deutlich höher als bei den attributnamenbasierten Verfahren. Das interessanteste Ergebnis der Evaluationen ist, dass die Verwendung einfacher Sprachmerkmale als OH-Vektor kodiert bessere Ergebnisse liefern, als höherwertige Wortvektoren wie FastText. Bei üblichen Aufgaben der Sprachverarbeitung, wie Textklassifikationen, erzielen diese Vektortransformationen deutlich bessere Ergebnisse als die einfachen Feature-Vektoren [71].

In einer Studie von Zhang et al. [99] wird die menschliche Leistungsfähigkeit im Schema-Matching mit einer Precision und einem Recall von jeweils über 0.90 angegeben. Dabei mussten die Probanden aus Vorschlägen die richtige Zuordnung auswählen. Die richtige Zuordnung war immer in den Vorschlägen enthalten. Die getroffenen Zuordnungen wurden anschließend mit einem Goldstandard verglichen.

Mussten mehrere relevante Zuordnungen gleichzeitig bewertet werden, sanken Precision und Recall drastisch [99]. Es ist daher davon auszugehen, dass ALR mindestens für Wertelisten einen menschlichen Precision-Wert erreicht hat.

Die Wertelisten in der Evaluation umfassten zusätzlich IDs und alleinstehende URLs, die nicht in den Text einer Attributinstanz eingebettet wurden. Für diese Attributtypen kann die Integration aufgrund der hohen Precision-Werte automatisch erfolgen, wenn ALR eine Vorhersage trifft. Für Texte und numerische Werte sind die erzielten Werte nicht ausreichend. Hier sollte ALR als Empfehlungssystem genutzt werden. Die Qualität der Schemazuordnungen ist jedoch für beide Attributtypen für die primäre Zuordnung besser als bei bisherigen Verfahren (siehe Abschnitt 9.9).

Leitfrage Unter Betrachtung der vorangegangenen Antworten und Ergebnisse lässt sich die Leitfrage beantworten sowie der erzielbare Automatisierungsgrad anhand von Abbildung 10.1 einordnen. ML-Verfahren erzielten bereits bei der ausschließlichen Verwendung von Attributnamen bessere Ergebnisse als die Referenzverfahren. Die Verwendung der vorhandenen Daten in ML-Verfahren kann folglich erheblich dazu beitragen, den manuellen Aufwand zu verringern. Bei der ausschließlichen Verwendung von Attributnamen können die Verfahren dennoch nur als Empfehlungssystem eingesetzt werden. Für eine vollständig automatische Verarbeitung reichen die erzielten Ergebnisse im Antikörpermarkt nicht aus.

Eine höhere Stufe des Automatisierungsgrads wird mit ALR erreicht, wenn Eigenschaften von Attributnamen und -instanzen berücksichtigt werden. In diesem Fall kann für den Attributtyp Werteliste aufgrund der hohen Precision-Werte davon ausgegangen werden, dass die Zuordnungen, wie beschrieben, ohne manuellen Eingriff ermittelt werden können. Die Zuordnungen des Attributtyps Text können weiterhin über die Einbettung von ALR in ein Empfehlungssystem ermittelt werden.

Eine vollständige Automatisierung wurde nicht erreicht. Dennoch konnten die hier entwickelten und evaluierten Verfahren, wie in Abschnitt 9.9 beschrieben, deutlich bessere Ergebnisse für die Zuordnungen erzielen als vergleichbare Verfahren.

10.2 Ausblick

Bisher wurden die in dieser Dissertation entwickelten und evaluierten Verfahren nur auf Grundlage von Antikörper-Produktdaten evaluiert. Die Übertragung der ML-Verfahren, die ausschließlich Attributnamen für die Zuordnungen verwenden, benötigt lediglich einen ausreichend großen Thesaurus, der bekannte Zuordnungen

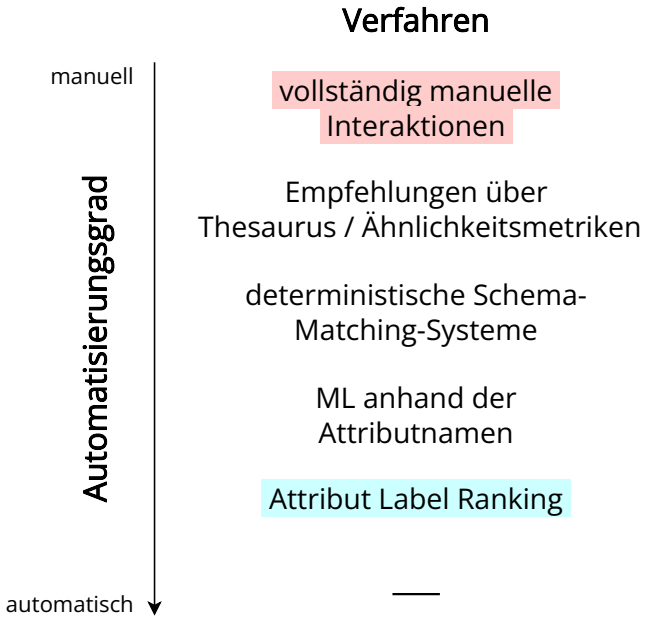


Abbildung 10.1: Abstufungen des Automatisierungsgrads verschiedener Verfahren bei der Ermittlung von Zuordnungen in der Katalogintegration

der Zulieferer enthält, allerdings nicht die vollständigen Produktkataloge vor und nach dem Integrationsprozess.

ALR lässt sich auf andere Bereiche außerhalb des Antikörpermarkts übertragen, wenn die folgenden beiden Voraussetzungen erfüllt sind:

1. Das Zielschema muss vor dem Training bekannt sein.
2. Aus den Rohdaten der Zulieferer und den zugehörigen marktplatzkonformen Produktkatalogen müssen sich Paare für den Silberstandard bilden lassen, mit dem ALR trainiert werden kann.

Hier sollten weitere Experimente auf größeren Datensätzen aus weiteren Anwendungsgebieten sowie manuell aufbereiteten Datensätzen als Goldstandard erfolgen, um die Qualität der Verfahren auch außerhalb des Antikörpermarkts nachzuweisen.

Um eine größere Datenmenge als Goldstandard zu erlangen, könnten die Verfahren über einen längeren Zeitraum als Empfehlungssystem in einem produktiven ETL-Prozesses laufen und dabei die Korrekturen der Mitarbeiter überwacht werden. Dadurch ließe sich für beide Ansätze über einen längeren Zeitraum ein Goldstandard mit einer genügend großen Anzahl von Beispielen erstellen.

Darüber hinaus bieten die aktuell aufkommenden großen Sprachmodelle (Large Language Models, LLMs) wie GPT-4 von OpenAI [100] weitere Verbesserungsmöglichkeiten für den Silberstandard. Insbesondere wurden LLMs bereits zur Automatisierung von Annotationen bei verschiedenen Aufgaben aus dem Bereich der Verarbeitung natürlicher Sprache [101] und bei der Dokumentensuche genutzt [102]. Bei letzterer wird durch das LLM ein Dokument erzeugt, das zu einer Suchanfrage passt und über Textvektorisierung mit bekannten Dokumenten verglichen. Ein ähnlicher Ansatz könnte sich auch dazu eignen, die Zuordnungen im Silberstandard besser ermitteln zu können. Bisher sind die genannten Verfahren aus [101] und [102] auf Basis von LLMs allerdings nur erste Fallstudien und wurden noch nicht wissenschaftlich validiert.

Trotz der bereits guten Ergebnisse als Empfehlungssystem und bzgl. der Precision auf Wertelisten, ist ein Verbesserungspotential bei ALR vorhanden. Zukünftige Ansätze oder Erweiterungen könnten die größte Einschränkung von ALR angehen: ALR berücksichtigt nur einzelne Spalten, ohne sie im Kontext der anderen Eingangsattribute zu verwenden. Dadurch kann die genaue Zuordnung nicht immer eindeutig aufgelöst werden, wenn aus Sicht von ALR mehrere gleichwertige Spalten aus dem Eingangskatalog für ein einzelnes Zielattribut geeignet sind. Diese Auflösung muss ggf. durch eine manuelle Interaktion erfolgen. Letztendlich ließe sich dadurch die Vorhersage der primären Zuordnung weiter verbessern.

Eine Möglichkeit diese Einschränkung aufzulösen könnte die Verwendung von Tabellenvektoren anstelle von Wortvektoren nach dem Verfahren von Cappuzzo et al. [93] bieten. Durch dieses Verfahren werden aus Attributnamen und -instanzen künstliche Sätze gebildet, von denen ein Vektorisierungsverfahren wie FastText, lernen kann. Dadurch würden die FastText-Vektoren direkt im Kontext der Produktattribute gelernt. Die vorgestellten Ergebnisse der Autoren im Schema-Matching lassen schließen, dass das Verfahren auch für die Katalogintegration nutzbar bzw. anpassbar ist. Die nach diesem Verfahren trainierten FastText-Vektoren könnten anstelle der OH oder der bisherigen FastText-Vektoren für ALR verwendet werden. Da diese Vektoren direkt im Kontext des Produktkatalogs gelernt werden, könnten sie bessere Ergebnisse erzielen. Auch hier sind weitere Experimente nötig.

Akronyme

ALR Attribut Label Ranking. vi, x, I, VIII, 16–18, 21, 77, 139, 140, 142–144, 146, 148, 151, 155, 156, 159–166, 168–172, 174–179, 181, 188, 190–192, 194–199

API application programming interface. 111

ASCII American Standard Code for Information Interchange. 149

AUC Area Under the Receiver Operating Characteristic Curve. vi, 17, 75, 76, 155, 169, 170, 196

B2C Business to Customer. 1

BCE Binary Cross Entropy. 160

CART Classification And Regression Tree. 64, 65

CBOW Continuous Bag Of Words. 94, 96

CNN Convolutional Neural Network. 171, 173, 175

CRISP-DM Industrieübergreifender Standardprozess für DM. v, 53, 56

CSV comma-separated value. 7, 9, 23, 31, 39, 41, 47

DAG directed acyclic graph. 182, 183

DM Data Mining. i, 51, 53–56

DOCX Word-Dokument. 9

DWH Data Warehouse. 2, 3, 30, 31

ELISA enzyme-linked immunosorbent assay. 157, 172

ETL extract, transform, load. 2, 4, 5, 9, 10, 12, 13, 30, 31, 34, 35, 43, 44, 46, 54, 144, 146, 147, 149, 153, 156, 163, 193, 198

FF Forschungsfragen. 14, 193

GAV global-as-view. V, 25, 26

GTIN Global Trade Item Number. 39, 42

ID Identifikationsnummer. 17, 36, 38, 40, 42, 43, 46, 48, 143, 161, 162, 197

JSON JavaScript Object Notation. vii, 150, 151

KMU kleine und mittelständische Unternehmen. I, 1–4, 6, 14, 146, 147, 193, 194

LAV local-as-view. V, 25, 26

LCS longest common subsequence. 85, 86, 112

LRAP Label Ranking Average Precision. 17, 74, 75, 155, 177, 178, 192

LTU Linear Threshold Unit. 62, 63

ML maschinellem Lernen. I, III, VI, 13, 15, 16, 20, 21, 49–57, 59, 60, 66–70, 73–76, 79–81, 92–95, 97, 99–105, 108, 110–112, 114, 116, 117, 119, 121, 122, 124, 127, 133, 138, 142, 146, 153, 179, 181–183, 186, 190, 194, 195, 197

MLP Multi-Layer Perzeptron. 63, 64, 117, 122–127, 129–137, 186, 195

MRR Mean Reciprocal Rank. ix, 73, 74, 76, 155, 163–165, 192

NLP natural language processing. 144

NN neuronales Netz. 15, 16, 62–66, 94, 140–143, 151, 155–157, 159–161, 163, 167, 171, 173, 186, 190, 191

OCR optical character recognition. 149

OH One-Hot. vi, 15, 93, 94, 97, 102, 103, 114, 116, 117, 119–123, 126, 127, 129–132, 135–137, 144, 145, 161, 162, 190, 191, 195, 196, 199

OOV Out Of Vocabulary. 80, 95, 96, 114, 116, 145, 164–166, 179, 187, 188

OWL Web Ontology Language. 26, 47, 111, 182, 185, 186

PCA Principal Component Analysis. 52

PDF Portable Document Format. 9, 43

RBF radiale Basisfunktion. 61

RDF Resource Description Framework. 36

ReLU rectified linear unit. 63, 160

REST representational state transfer. 137

ROC Receiver Operating Characteristic. 75

SVM Support Vector Machine. 15, 51, 59–64, 66, 75, 117, 121–123, 126, 127, 129–132, 183, 195

t-SNE t-distributed Stochastic Neighbor Embedding. 52

TF-IDF term frequency-inverse document frequency. 15, 93, 94, 102, 103, 114, 116, 117, 119–125, 127, 129–132, 135, 136, 145, 190, 194, 195

URL Uniform Resource Locator. VI, 17, 36, 37, 39, 42, 43, 46, 48, 144, 197

UT unstrukturierter Text. 42, 47

XLSX Excel-Arbeitsmappe. 7, 9, 31, 47

XML Extensible Markup Language. 23, 27, 47, 111, 182

Abbildungsverzeichnis

1.1 Ablauf der automatischen Produktdatenintegration für Online-Shop-Betreiber im Marktplatz von Galaxus. [8] 3
1.2 Übliche Katalogintegration im Online-Handel 5
1.3 Vereinfachter Überblick über den Ablauf zur Integration gelieferter Produktdaten in den Online-Shop bei antibodies-online. 8
1.4 Ausgangssituation 11

2.1 Vergleich der grundsätzlichen Abläufe des GAV- und LAV-Ansatzes . . 25
2.2 Ontologie-Matching-Prozess 27
2.3 Kategorisierung der Matching-Ansätze 28
2.4 Katalogintegration nach Euzenat und Shvaiko 30
2.5 Verteilung genutzter Attribute von Zulieferern 33
2.6 Beziehung zwischen Attributnamen aus tabellarischen Eingangs- und Zielschemata 34
2.7 Informationsursprung aus Sicht des Zielschemas 35
2.8 Typische Herausforderungen bei der Integration von Produktkatalogen 44

3.1 Vergleich des Entwicklungsvorgehens 50
3.2 CRISP-DM 53
3.3 Phasen der Entwicklung im Vergleich 55
3.4 Schemazuordnung mithilfe von ML-Systemen 60
3.5 Schemazuordnung mithilfe eines MLP 64
3.6 Schemazuordnung mithilfe eines Entscheidungsbaumes 66
3.7 Hard voting von verschiedenen Vorhersagen 67

4.1 CBOW-Architektur nach Mikolov et al. [69] 95

5.1 Grundsätzlicher Ablauf der Vorhersage von Zuordnungen bei einem Klassifikationsansatz anhand von Attributnamen 100

5.2 Bestimmung eines Feature-Vektors aus den Ähnlichkeitsmetriken . . 102
5.3 Anwendung der OH-Kodierung auf den Eingangsattributnamen *Product_Name*. 103
5.4 Nutzen von FastText zur Bestimmung eines Feature-Vectors 104
5.5 Zielattribute mit mindestens zehn verschiedenen manuell zugeordneten Eingangsattributnamen 106
5.6 Visualisierung des Vektorraums aus Ähnlichkeisfunktionen 113
5.7 Visualisierung des Vektorraums der TF-IDF-Vektoren 115

6.1 Normalisierte Konfusionsmatrix 125

7.1 Grundsätzlicher Ablauf des Attribut Label Ranking 140
7.2 Grundsätzlicher Ablauf der Vorhersage von Zuordnungen bei einem Klassifikationsansatz anhand von Attributnamen 141
7.3 Grundsätzlicher Ablauf der Vorhersage von Zuordnungen bei einem Klassifikationsansatz mit ALR 143
7.4 Datenerhebung im laufenden Integrationsprozess 147
7.5 Vorverarbeitung der jeweiligen Eingangs- und Zielkataloge 148
7.6 Datenerhebung im laufenden Integrationsprozess 150

8.1 Kardinalität der Zuordnungen abhängig von ihrer Attributart bei unterschiedlichen Schwellwerten. 158
8.2 ALR mit einer einfachen Netzwerkstruktur 160
8.3 AUC pro Zielklasse 170

10.1 Abstufungen des Automatisierungsgrads verschiedener Verfahren bei der Ermittlung von Zuordnungen in der Katalogintegration 198

Listings

5.1 Beispielimplementierung des einfachen Ansatzes 109
5.2 Beispielimplementierung des pragmatischen Ansatzes mit Thesaurus und Ähnlichkeitsmetriken 110

7.1 Ausschnitt der Zielstruktur im JSON-Format aus Schmidts et al. [S5] . 151

Tabellenverzeichnis

2.1 Auszug des Produktschemas von schema.org 37
2.2 Ausschnitt des allgemeinen Schemas von GS1 39
2.3 Ausschnitt von speziellen GS1 Attriuten 40
2.4 Ausschnitt der Attribute von Shopify 41

3.1 Häufig genutzte Kernel für SVM 61
3.2 Kategorisierung der Ergebnisse eines Schema-Matching-Systems . . 69
3.3 Konfusionsmatrix zwischen drei Attributen als Klassen. 72
3.4 Beispielhafte Ermittlung des MRR 74

5.1 Attributnamen mit verschiedenen Fehlern und typischen Problemen in der Zuordnung. 107
5.2 Eingangsattributnamen für das Zielattribut `Hosts` 107
5.3 Auflistung der zu valuierenden Verfahren 117

6.1 Vergleich der gewichteten Mittelwerte in Szenario 1 122
6.2 Vergleich der gewichteten Mittelwerte in Szenario 2 123
6.3 Eingangsattributnamen für das Zielattribut `Hosts` im Testdatensatz. 126
6.4 Gewichtete Mittelwerte einer fünffachen Überkreuzvalidierung aus Szenario 1 und Szenario 2 127
6.5 Vergleich der gewichteten Mittelwerte in Szenario 3 129
6.6 Gewichtete Mittelwerte mit Standardabweichung einer fünffachen Überkreuzvalidierung der wortvektorbasierten Verfahren aus Szenario 3 . 130
6.7 Vergleich der gewichteten Mittelwerte in Szenario 4 131
6.8 Gewichtete Mittelwerte mit Standardabweichung einer fünffachen Überkreuzvalidierung der Wortvektor-basierten Verfahren aus Szenario 4 132
6.9 Gewichtete Mittelwerte auf dem Testdatensatz 135
6.10 Top-3-Genauigkeit der Verfahren aus verschiedenen Szenarien auf dem Testdatensatz. 136

7.1 Liste der Merkmale aus den Metadaten der Attributinstanzen 144
7.2 Eigenschaften der Eingangskataloge aus dem erhobenen Datensatz . 147

8.1 Optimale Hyperparameter (gerundet) 161
8.2 MRR nach Attribubttyp 164
8.3 Gewichteter Mittelwert der Klassifikationsergebnisse 166
8.4 Klassifikationsergebnisse aufgeschlüsselt nach Attributtyp 167
8.5 Gesamtergebnis von ALR mit zwei Modellkonfigurationen 168
8.6 Klassifikationsergebnisse aufgeschlüsselt nach Attributtyp bei unscharfen Zuordnungen 168
8.7 Gewichteter Mittelwert des AUC für ALR 169
8.8 Gesamtergebnis der Klassifikation mit verschiedenen Modellkonfigurationen 175
8.9 Klassifizierungsergebnisse für die primäre Zuordnung von ALR 176
8.10 LRAP-Werte bei der Vorhersage mehrerer Zuordnungen durch ALR . 177
8.11 LRAP-Werte bei der Vorhersage mehrerer Zuordnungen durch ALR, aufgeteilt nach Attributtyp 178

Eigene Publikationen

[S1] O. Schmidts, M. Boltes, B. Kraft und M. Schreiber, „Multi-pedestrian tracking by moving Bluetooth-LE beacons and stationary receivers“, in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 18-21 September 2017, Sapporo, Japan*, 2017, Seiten 1–4. Adresse: https://juser.fz-juelich.de/record/842817/files/178.pdf.

[S2] O. Schmidts, B. Kraft, M. Schreiber und A. Zündorf, „Continuously evaluated research projects in collaborative decoupled environments“, in *Proceedings of the 5th International Workshop on Software Engineering Research and Industrial Practice*, Gothenburg Sweden: ACM, 29. Mai 2018, Seiten 2–9, ISBN: 978-1-4503-5744-9. DOI: 10.1145/3195546.3195549. Adresse: https://dl.acm.org/doi/10.1145/3195546.3195549 (besucht am 04. 03. 2023).

[S3] O. Schmidts, B. Kraft, I. Siebigteroth und A. Zündorf, „Schema Matching with Frequent Changes on Semi-Structured Input Files: A Machine Learning Approach on Biological Product Data:“ In *Proceedings of the 21st International Conference on Enterprise Information Systems*, Heraklion, Crete, Greece: SCITEPRESS - Science and Technology Publications, 2019, Seiten 208–215, ISBN: 978-989-758-372-8. DOI: 10.5220/0007723602080215. Adresse: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007723602080215 (besucht am 24. 07. 2020).

[S4] I. Siebigteroth, B. Kraft, O. Schmidts und A. Zündorf, „A Study on Improving Corpus Creation by Pair Annotation“, in *Proceedings of the Poster Session of the 2nd Conference on Language, Data and Knowledge (LDK-PS 2019)*, 2019, Seiten 40–44. Adresse: http://ceur-ws.org/Vol-2402/paper8.pdf.

[S5] O. Schmidts, B. Kraft, M. Winkens und A. Zündorf, „Catalog Integration of Low-quality Product Data by Attribute Label Ranking:“ In *Proceedings of the 9th International Conference on Data Science, Technology and Applications*, Lieusaint - Paris, France: SCITEPRESS - Science and Technology Publications, 2020, Seiten 90–101, ISBN: 978-989-758-440-4. DOI: 10.5220/0009831000900101. Adresse: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0009831000900101 (besucht am 24. 07. 2020).

[S6] M. Sildatke, H. Karwanni, B. Kraft, O. Schmidts und A. Zündorf, „Automated Software Quality Monitoring in Research Collaboration Projects“, in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, Seoul Republic of Korea: ACM, 27. Juni 2020, Seiten 603–610, ISBN: 978-1-4503-7963-2. DOI: 10.1145/3387940.3391478. Adresse: https://dl.acm.org/doi/10.1145/3387940.3391478 (besucht am 04. 03. 2023).

[S7] O. Schmidts, B. Kraft, M. Winkens und A. Zündorf, „Catalog Integration of Heterogeneous and Volatile Product Data“, in *Data Management Technologies and Applications*, S. Hammoudi, C. Quix und J. Bernardino, Herausgeber, Band 1446, Cham: Springer International Publishing, 2021, Seiten 134–153, ISBN: 978-3-030-83013-7 978-3-030-83014-4. DOI: 10.1007/978-3-030-83014-4_7. Adresse: https://link.springer.com/10.1007/978-3-030-83014-4_7 (besucht am 04. 03. 2023).

[S8] P. Kohl, O. Schmidts, L. Klöser, H. Werth, B. Kraft und A. Zündorf, „STAMP 4 NLP – An Agile Framework for Rapid Quality-Driven NLP Applications Development“, in *Quality of Information and Communications Technology*, A. C. R. Paiva, A. R. Cavalli, P. Ventura Martins und R. Pérez-Castillo, Herausgeber, Band 1439, Cham: Springer International Publishing, 2021, Seiten 156–166, ISBN: 978-3-030-85346-4 978-3-030-85347-1. DOI: 10.1007/978-3-030-85347-1_12. Adresse: https://link.springer.com/10.1007/978-3-030-85347-1_12 (besucht am 04. 03. 2023).

[S9] A. Büsgen, L. Klöser, P. Kohl, O. Schmidts, B. Kraft und A. Zündorf, „Exploratory Analysis of Chat-based Black Market Profiles with Natural Language Processing:“ In *Proceedings of the 11th International Conference on Data Science, Technology and Applications*, Lisbon, Portugal: SCITEPRESS - Science and Technology Publications, 2022, Seiten 83–94, ISBN: 978-989-758-583-8. DOI: 10.5220/0011271400003269. Adresse: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0011271400003269 (besucht am 04. 03. 2023).

[S10] D. Thewes, H. Werth, J. Wienströer, S. von Stillfried und Rattonitz, O. Schmidts, I. Siebigteroth, P. Boor, R. Röhrig und M. Meinecke, „Patient feature substitution with synthetic data for developing open corpora“, German Medical Science GMS Publishing House, 19. Aug. 2022, DocAbstr. 2. DOI: 10.3205/22gmds026.

[S11] A. Büsgen, L. Klöser, P. Kohl, O. Schmidts, B. Kraft und A. Zündorf, „From Cracked Accounts to Fake IDs - User Profiling on German Telegram Black Market Channels“, in to appear.

Literaturverzeichnis

[1] „E-commerce-einzelhandel - umsatz weltweit 2026“, Statista. (), Adresse: https://de.statista.com/statistik/daten/studie/244110/umfrage/globaler-umsatz-von-e-commerce/ (besucht am 21. 08. 2022).

[2] „B2c-e-commerce: Top 100 online shops“, Statista. (), Adresse: https://de.statista.com/statistik/daten/studie/170530/umfrage/umsatz-der-groessten-online-shops-in-deutschland/ (besucht am 21. 08. 2022).

[3] L. Rabe. „Themenseite: E-commerce in deutschland“, Statista. (), Adresse: https://de.statista.com/themen/247/e-commerce/ (besucht am 21. 08. 2022).

[4] „Rücksendungen: Retouren von paketen und artikeln in deutschland 2020“, Statista. (), Adresse: https://de.statista.com/statistik/daten/studie/1082408/umfrage/retouren-von-paketen-und-artikeln-in-deutschland/ (besucht am 21. 08. 2022).

[5] „Online-shopping: Gründe für retouren 2021“, Statista. (), Adresse: https://de.statista.com/statistik/daten/studie/1312304/umfrage/gruende-fuer-retouren-in-deutschland/ (besucht am 21. 08. 2022).

[6] „Retouren - maßnahmen zur reduzierung 2021“, Statista. (), Adresse: https://de.statista.com/statistik/daten/studie/1312345/umfrage/massnahmen-zur-reduzierung-von-retouren-in-deutschland/ (besucht am 21. 08. 2022).

[7] „Lagerbestandsdateivorlagen – Amazon Seller Central“. (), Adresse: https://sellercentral.amazon.de/gp/help/external/1641?language=de_DE&ref=efph_1641_cont_G5819 (besucht am 27. 12. 2021).

[8] „Technical integration“, Galaxus. (24. Juli 2019), Adresse: https://www.galaxus.ch/en/page/technical-integration-12466 (besucht am 27. 12. 2021).

[9] S. H. Center. „Verwenden von csv-dateien“, Shopify Help Center. (), Adresse: https://help.shopify.com/de/manual/products/import-export/using-csv (besucht am 09. 09. 2021).

[10] P. Schlieker, „IT-Assisted provision of product data to online retailers in the home & living sector“,

[11] M. Schreiber, „Towards Effective Natural Language Application Development“, 2019. DOI: 10.17170/kobra-20190529539. Adresse: https://kobra.uni-kassel.de/handle/123456789/11255 (besucht am 13. 11. 2021).

[12] antibodies-online GmbH. „Wir über uns“, Wir über uns. (2. Apr. 2023), Adresse: https://www.antikoerper-online.de/aboutus/ (besucht am 02. 04. 2023).

[13] T. Bönner, *Automating systematic transformations on structured data*, 15. Dez. 2017.

[14] Z. Chen, H. Jia, J. Heflin und B. D. Davison, „Generating Schema Labels Through Dataset Content Analysis“, in *Companion Proceedings of the The Web Conference 2018*, Serie WWW '18, Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, Seiten 1515–1522, ISBN: 978-1-4503-5640-4. DOI: 10.1145/3184558.3191601. Adresse: https://doi.org/10.1145/3184558.3191601 (besucht am 11. 10. 2018).

[15] A. Doan, A. Halevy und Z. Ives, „1 - Introduction“, in *Principles of Data Integration*, A. Doan, A. Halevy und Z. Ives, Herausgeber, Boston: Morgan Kaufmann, 1. Jan. 2012, Seiten 1–18, ISBN: 978-0-12-416044-6. DOI: 10.1016/B978-0-12-416044-6.00001-6. Adresse: https://www.sciencedirect.com/science/article/pii/B9780124160446000016 (besucht am 29. 12. 2021).

[16] M. Lenzerini, „Data Integration: A Theoretical Perspective“, in *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Serie PODS '02, New York, NY, USA: ACM, 2002, Seiten 233–246, ISBN: 978-1-58113-507-7. DOI: 10.1145/543613.543644. Adresse: http://doi.acm.org/10.1145/543613.543644 (besucht am 05. 04. 2018).

[17] A. Doan, A. Halevy und Z. Ives, „3 - Describing Data Sources“, in *Principles of Data Integration*, A. Doan, A. Halevy und Z. Ives, Herausgeber, Boston: Morgan Kaufmann, 1. Jan. 2012, Seiten 65–94, ISBN: 978-0-12-416044-6. DOI: 10.1016/B978-0-12-416044-6.00003-X. Adresse: https://www.sciencedirect.com/science/article/pii/B978012416044600003X (besucht am 28. 12. 2021).

[18] M. Mesiti, E. Jiménez-Ruiz, I. Sanz, R. Berlanga-Llavori, P. Perlasca, G. Valentini und D. Manset, „XML-based approaches for the integration of heterogeneous bio-molecular data“, *BMC Bioinformatics*, Jahrgang 10, Nummer 12, S7, 15. Okt. 2009, ISSN: 1471-2105. DOI: 10.1186/1471-2105-10-S12-S7. Adresse: https://doi.org/10.1186/1471-2105-10-S12-S7 (besucht am 28. 12. 2021).

[19] P. Shvaiko und J. Euzenat, „Ontology Matching: State of the Art and Future Challenges“, *IEEE Transactions on Knowledge and Data Engineering*, Jahrgang 25, Nummer 1, Seiten 158–176, Jan. 2013, ISSN: 1558-2191. DOI: 10.1109/TKDE.2011.253.

[20] P. A. Bernstein, J. Madhavan und E. Rahm, „Generic schema matching, ten years later“, *Proceedings of the VLDB Endowment*, Jahrgang 4, Nummer 11, Seiten 695–701, 11 2011.

[21] J. Euzenat und P. Shvaiko, „The Matching Problem“, in *Ontology Matching*, J. Euzenat und P. Shvaiko, Herausgeber, Berlin, Heidelberg: Springer, 2013, Seiten 25–54, ISBN: 978-3-642-38721-0. DOI: 10.1007/978-3-642-38721-0_2. Adresse: https://doi.org/10.1007/978-3-642-38721-0_2 (besucht am 03. 01. 2022).

[22] P. Shvaiko und J. Euzenat, „A Survey of Schema-Based Matching Approaches“, in *Journal on Data Semantics IV*, S. Spaccapietra, Herausgeber, Serie Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, Seiten 146–171, ISBN: 978-3-540-31447-9.

[23] J. Euzenat und P. Shvaiko, „Classifications of Ontology Matching Techniques“, in *Ontology Matching*, J. Euzenat und P. Shvaiko, Herausgeber, Berlin, Heidelberg: Springer, 2013, Seiten 73–84, ISBN: 978-3-642-38721-0. DOI: 10.1007/978-3-642-38721-0_4. Adresse: https://doi.org/10.1007/978-3-642-38721-0_4 (besucht am 03. 01. 2022).

[24] J. Euzenat und P. Shvaiko, *Ontology Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-38720-3 978-3-642-38721-0. DOI: 10.1007/978-3-642-38721-0. Adresse: http://link.springer.com/10.1007/978-3-642-38721-0 (besucht am 03. 01. 2022).

[25] D. Fensel, D. McGuiness, E. Schulten, Wee Keong Ng, Ge Peng Lim und Guanghao Yan, „Ontologies and electronic commerce“, *IEEE Intelligent Systems*, Jahrgang 16, Nummer 1, Seiten 8–14, Jan. 2001, ISSN: 1541-1672. DOI: 10.1109/MIS.2001.1183337. Adresse: http://ieeexplore.ieee.org/document/1183337/ (besucht am 06. 01. 2022).

[26] J. Euzenat und P. Shvaiko, „Applications“, in *Ontology Matching*, J. Euzenat und P. Shvaiko, Herausgeber, Berlin, Heidelberg: Springer, 2013, Seiten 3–24, ISBN: 978-3-642-38721-0. DOI: 10.1007/978-3-642-38721-0_1. Adresse: https://doi.org/10.1007/978-3-642-38721-0_1 (besucht am 03. 01. 2022).

[27] A. Doan, A. Halevy und Z. Ives, „10 - Data Warehousing and Caching“, in *Principles of Data Integration*, A. Doan, A. Halevy und Z. Ives, Herausgeber, Boston: Morgan Kaufmann, 1. Jan. 2012, Seiten 271–288, ISBN: 978-0-12-416044-6. DOI: 10.1016/B978-0-12-416044-6.00010-7. Adresse: https://www.sciencedirect.com/science/article/pii/B9780124160446000107 (besucht am 31. 12. 2021).

[28] S. K. Bansal, „Towards a Semantic Extract-Transform-Load (ETL) Framework for Big Data Integration“, in *2014 IEEE International Congress on Big Data*, Juni 2014, Seiten 522–529. DOI: 10.1109/BigData.Congress.2014.82.

[29] schema.org. „Schema.org is ten!“ (), Adresse: http://blog.schema.org/2021/06/schemaorg-is-ten.html (besucht am 16. 11. 2021).

[30] schema.org, *Schema.org on Github*, schema.org project, 15. Nov. 2021. Adresse: https://github.com/schemaorg/schemaorg (besucht am 16. 11. 2021).

[31] schema.org. „Product - Schema.org Type“. (), Adresse: https://schema.org/Product (besucht am 13. 11. 2021).

[32] K. Alldredge, K. Bansal, O. Sho und V. Skinner, „Want to improve consumer experience? Collaborate to build a product data standard“, Seite 8, 2020. Adresse: https://www.gs1.org/sites/default/files/want-to-improve-consumer-experience-collaborate-to-build-a-product-data-standard.pdf.

[33] „GS1 Web Vocabulary“. (), Adresse: https://www.gs1.org/voc/ (besucht am 16. 11. 2021).

[34] S. H. Center. „Produkte“, Shopify Help Center. (), Adresse: https://help.shopify.com/de/manual/products (besucht am 20. 11. 2021).

[35] S. H. Center. „Start, wachstum und skalierung deines unternehmens“, Shopify. (), Adresse: https://www.shopify.de (besucht am 20. 11. 2021).

[36] „Massifying data quality benefits in Colombian retail industry – GS1 Colombia“, Seite 6,

[37] A. L. Samuel, „Some Studies in Machine Learning Using the Game of Checkers“, *IBM Journal of Research and Development*, Jahrgang 3, Nummer 3, Seiten 210–229, Juli 1959, ISSN: 0018-8646. DOI: 10.1147/rd.33.0210.

[38] S. Studer, T. B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters und K.-R. Mueller, „Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology“, 11. März 2020. arXiv: 2003.05155 [cs, stat]. Adresse: http://arxiv.org/abs/2003.05155 (besucht am 23. 02. 2021).

[39] R. Wirth und J. Hipp, „CRISP-DM: Towards a Standard Process Model for Data Mining“, *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, Seiten 29–39, 2000.

[40] M. Banko und E. Brill, „Scaling to very very large corpora for natural language disambiguation“, in *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics - ACL '01*, Toulouse, France: Association for Computational Linguistics, 2001, Seiten 26–33. DOI: 10.3115/1073012.1073017. Adresse: http://portal.acm.org/citation.cfm?doid=1073012.1073017 (besucht am 12. 02. 2022).

[41] A. Halevy, P. Norvig und F. Pereira, „The Unreasonable Effectiveness of Data“, *IEEE Intelligent Systems*, Jahrgang 24, Nummer 2, Seiten 8–12, März 2009, ISSN: 1941-1294. DOI: 10.1109/MIS.2009.36.

[42] N. Cristianini, J. Shawe-Taylor und D. o. C. S. R. H. J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 23. März 2000, 216 Seiten, ISBN: 978-0-521-78019-3. Google Books: _PXJn_cxv0AC.

[43] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2. Auflage. O'Reilly Media, Inc., 2019, 856 Seiten, ISBN: 978-1-4920-3264-9.

[44] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 19. Nov. 1999, 340 Seiten, ISBN: 978-0-387-98780-4. Google Books: sna9BaxVbj8C.

[45] D. E. Rumelhart, G. E. Hinton und R. J. Williams, „Learning Internal Representations by Error Propagation", CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1. Sep. 1985. Adresse: https://apps.dtic.mil/sti/citations/ADA164453 (besucht am 16. 03. 2022).

[46] H.-H. Do und E. Rahm, „COMA: A System for Flexible Combination of Schema Matching Approaches", in *Proceedings of the 28th International Conference on Very Large Data Bases*, Serie VLDB '02, Hong Kong, China: VLDB Endowment, 2002, Seiten 610–621. Adresse: http://dl.acm.org/citation.cfm?id=1287369.1287422 (besucht am 13. 09. 2018).

[47] N. Craswell, „Mean Reciprocal Rank", in *Encyclopedia of Database Systems*, L. I. N. G. LIU und M. T. ÖZSU, Herausgeber, Boston, MA: Springer US, 2009, Seiten 1703–1703, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_488. Adresse: https://doi.org/10.1007/978-0-387-39940-9_488 (besucht am 31. 03. 2022).

[48] G. Tsoumakas, I. Katakis und I. Vlahavas, „Mining Multi-label Data", in *Data Mining and Knowledge Discovery Handbook*, O. Maimon und L. Rokach, Herausgeber, Boston, MA: Springer US, 2010, Seiten 667–685, ISBN: 978-0-387-09823-4. DOI: 10.1007/978-0-387-09823-4_34. Adresse: https://doi.org/10.1007/978-0-387-09823-4_34 (besucht am 30. 01. 2020).

[49] C. Ferri, J. Hernández-Orallo und R. Modroiu, „An experimental comparison of performance measures for classification", *Pattern Recognition Letters*, Jahrgang 30, Nummer 1, Seiten 27–38, Jan. 2009, ISSN: 01678655. DOI: 10.1016/j.patrec.2008.08.010. Adresse: https://linkinghub.elsevier.com/retrieve/pii/S0167865508002687 (besucht am 07. 04. 2022).

[50] A. Mikheev. „Text Segmentation", The Oxford Handbook of Computational Linguistics. (13. Jan. 2005), Adresse: https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199276349.001.0001/oxfordhb-9780199276349-e-10 (besucht am 10. 04. 2022).

[51] J. Euzenat und P. Shvaiko, „Basic Similarity Measures", in *Ontology Matching*, J. Euzenat und P. Shvaiko, Herausgeber, Berlin, Heidelberg: Springer, 2013, Seiten 85–120, ISBN: 978-3-642-38721-0. DOI: 10.1007/978-3-642-38721-0_5. Adresse: https://doi.org/10.1007/978-3-642-38721-0_5 (besucht am 03. 01. 2022).

[52] A. G. Jivani, „A Comparative Study of Stemming Algorithms", Jahrgang 2, Seite 10, 2011, ISSN: 2229-6093.

[53] S. Melnik, H. Garcia-Molina und E. Rahm, „Similarity flooding: A versatile graph matching algorithm and its application to schema matching“, in *Proceedings 18th International Conference on Data Engineering*, San Jose, CA, USA: IEEE Comput. Soc, 2002, Seiten 117–128, ISBN: 978-0-7695-1531-1. DOI: 10.1109/ICDE.2002.994702. Adresse: http://ieeexplore.ieee.org/document/994702/ (besucht am 21. 09. 2018).

[54] A. Doan, A. Halevy und Z. Ives, „4 - String Matching“, in *Principles of Data Integration*, A. Doan, A. Halevy und Z. Ives, Herausgeber, Boston: Morgan Kaufmann, 1. Jan. 2012, Seiten 95–119, ISBN: 978-0-12-416044-6. DOI: 10.1016/B978-0-12-416044-6.00004-1. Adresse: https://www.sciencedirect.com/science/article/pii/B9780124160446000041 (besucht am 21. 04. 2022).

[55] D. Bär, T. Zesch und I. Gurevych, „DKPro Similarity: An Open Source Framework for Text Similarity“, in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, Seiten 121–126. Adresse: http://www.aclweb.org/anthology/P13-4021.

[56] G. Navarro, „A guided tour to approximate string matching“, *ACM computing surveys (CSUR)*, Jahrgang 33, Nummer 1, Seiten 31–88, 2001.

[57] S. B. Needleman und C. D. Wunsch, „A general method applicable to the search for similarities in the amino acid sequence of two proteins“, *Journal of Molecular Biology*, Jahrgang 48, Nummer 3, Seiten 443–453, 28. März 1970, ISSN: 0022-2836. DOI: 10.1016/0022-2836(70)90057-4. Adresse: https://www.sciencedirect.com/science/article/pii/0022283670900574 (besucht am 26. 04. 2022).

[58] M. S. Waterman, T. F. Smith und W. A. Beyer, „Some biological sequence metrics“, *Advances in Mathematics*, Jahrgang 20, Nummer 3, Seiten 367–387, 1. Juni 1976, ISSN: 0001-8708. DOI: 10.1016/0001-8708(76)90202-4. Adresse: https://www.sciencedirect.com/science/article/pii/0001870876902024 (besucht am 26. 04. 2022).

[59] T. Smith und M. Waterman, „Identification of common molecular subsequences“, *Journal of Molecular Biology*, Jahrgang 147, Nummer 1, Seiten 195–197, März 1981, ISSN: 00222836. DOI: 10.1016/0022-2836(81)90087-5. Adresse: https://linkinghub.elsevier.com/retrieve/pii/0022283681900875 (besucht am 26. 04. 2022).

[60] W. E. Winkler und Y. Thibaudeau, „An application of the Fellegi-Sunter model of record linkage to the 1990 US decennial census.“, *US Bureau of the Census*, Seite 22, 1991.

[61] P. Jaccard, „Étude comparative de la distribution florale dans une portion des Alpes et des Jura“, *Bull Soc Vaudoise Sci Nat*, Jahrgang 37, Seiten 547–579, 1901.

[62] C. Romesburg, *Cluster Analysis for Researchers*. Lulu.com, 2004, 341 Seiten, ISBN: 978-1-4116-0617-3. Google Books: ZuIPv7OKm10C.

[63] J. Zobel und P. Dart, „Phonetic string matching: Lessons from information retrieval“, in *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996, Seiten 166–172.

[64] H. J. Postel, „Die Kölner Phonetik. Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse“, *IBM-Nachrichten*, Jahrgang 19, Seiten 925–931, 1969.

[65] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998, 452 Seiten, ISBN: 978-0-262-06197-1. Google Books: Rehu800zMIMC.

[66] Z. Wu und M. Palmer, „Verb Semantics and Lexical Selection“, in *32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, USA: Association for Computational Linguistics, Juni 1994, Seiten 133–138. DOI: 10.3115/981732.981751. Adresse: https://aclanthology.org/P94-1019 (besucht am 12. 02. 2023).

[67] Juan J. Lastra-Diaz, Alicia Lara-Clares, Ana Garcia Serrano. „HESML“, HESML. (), Adresse: http://hesml.lsi.uned.es/pairwise-semantic-similarity-measures (besucht am 12. 02. 2023).

[68] H. Schütze, C. D. Manning und P. Raghavan, *Introduction to Information Retrieval*. Cambridge University Press Cambridge, 2008, Band 39.

[69] T. Mikolov, K. Chen, G. Corrado und J. Dean, „Efficient Estimation of Word Representations in Vector Space“, 6. Sep. 2013. arXiv: 1301.3781 [cs]. Adresse: http://arxiv.org/abs/1301.3781 (besucht am 05. 05. 2022).

[70] P. Bojanowski, E. Grave, A. Joulin und T. Mikolov, „Enriching Word Vectors with Subword Information“, *Transactions of the Association for Computational Linguistics*, Jahrgang 5, Seiten 135–146, 1. Juni 2017, ISSN: 2307-387X. DOI: 10.1162/tacl_a_00051. Adresse: https://doi.org/10.1162/tacl_a_00051 (besucht am 13. 11. 2021).

[71] A. Joulin, E. Grave, P. Bojanowski und T. Mikolov, „Bag of Tricks for Efficient Text Classification“, 6. Juli 2016. arXiv: 1607.01759 [cs]. Adresse: http://arxiv.org/abs/1607.01759 (besucht am 03. 12. 2018).

[72] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch und A. Joulin, „Advances in Pre-Training Distributed Word Representations“, in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018. Adresse: https://arxiv.org/pdf/1712.09405.pdf.

[73] S. Massmann, S. Raunich, D. Aumüller, P. Arnold und E. Rahm, „Evolution of the COMA match system“, in *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, CEUR-WS. org, 2011, Seiten 49–60.

[74] D. Aumueller, H.-H. Do, S. Massmann und E. Rahm, „Schema and ontology matching with COMA++“, in *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, Acm, 2005, Seiten 906–908.

[75] P. Arnold und E. Rahm, „Enriching ontology mappings with semantic relations“, *Data & Knowledge Engineering*, Selected Papers from the 17th East-¬-European Conference on Advances in Databases and Information Systems, Jahrgang 93, Seiten 1–18, 1. Sep. 2014, ISSN: 0169-023X. DOI: 10.1016/j.datak.2014.07.001. Adresse: https://www.sciencedirect.com/science/article/pii/S0169023X14000603 (besucht am 13. 11. 2021).

[76] L. Van der Maaten und G. Hinton, „Visualizing data using t-SNE.“, *Journal of machine learning research*, Jahrgang 9, Nummer 11, 2008.

[77] B. Li, Y. Hou und W. Che, „Data augmentation approaches in natural language processing: A survey“, *AI Open*, Jahrgang 3, Seiten 71–90, 1. Jan. 2022, ISSN: 2666-6510. DOI: 10.1016/j.aiopen.2022.03.001. Adresse: https://www.sciencedirect.com/science/article/pii/S2666651022000080 (besucht am 17. 11. 2022).

[78] M. Research, *fastText*, Version 2cc7f54ac0, Meta Research, 2020. Adresse: https://github.com/facebookresearch/fastText/blob/2cc7f54a/src/fasttext.cc (besucht am 29. 06. 2023).

[79] J. Devlin, M.-W. Chang, K. Lee und K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding“, 24. Mai 2019. arXiv: 1810.04805 [cs]. Adresse: http://arxiv.org/abs/1810.04805 (besucht am 13. 11. 2021).

[80] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee und L. Zettlemoyer, „Deep Contextualized Word Representations“, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Juni 2018, Seiten 2227–2237. DOI: 10.18653/v1/N18-1202. Adresse: https://aclanthology.org/N18-1202 (besucht am 25. 06. 2023).

[81] M. Neumann, D. King, I. Beltagy und W. Ammar, „ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing“, in *BioNLP@ACL*, 2019. DOI: 10.18653/v1/W19-5034.

[82] J. Nam, J. Kim, E. Loza Mencía, I. Gurevych und J. Fürnkranz, „Large-Scale Multi-label Text Classification — Revisiting Neural Networks“, in *Machine Learning and Knowledge Discovery in Databases*, T. Calders, F. Esposito, E. Hüllermeier und R. Meo, Herausgeber, Serie Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2014, Seiten 437–452, ISBN: 978-3-662-44851-9. DOI: 10.1007/978-3-662-44851-9_28.

[83] T. Akiba, S. Sano, T. Yanase, T. Ohta und M. Koyama, „Optuna: A Next-generation Hyperparameter Optimization Framework“, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Serie KDD '19, New York, NY, USA: Association for Computing Machinery, 25. Juli 2019, Seiten 2623–2631, ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330701. Adresse: https://doi.org/10.1145/3292500.3330701 (besucht am 04. 12. 2022).

[84] J. Heaton, *Introduction to Neural Networks with Java, 2nd Edition*, 2nd edition. St. Louis, Mo: Heaton Research, Incorporated, 1. Okt. 2008, 440 Seiten, ISBN: 978-1-60439-008-7.

[85] Y. Kim. „Convolutional Neural Networks for Sentence Classification“. arXiv: 1408.5882 [cs]. (2. Sep. 2014), Adresse: http://arxiv.org/abs/1408.5882 (besucht am 04. 01. 2023), preprint.

[86] A. Jacovi, O. S. Shalom und Y. Goldberg. „Understanding Convolutional Neural Networks for Text Classification“. arXiv: 1809.08037 [cs]. (27. Apr. 2020), Adresse: http://arxiv.org/abs/1809.08037 (besucht am 04. 01. 2023), preprint.

[87] N. Kalchbrenner, E. Grefenstette und P. Blunsom. „A Convolutional Neural Network for Modelling Sentences“. arXiv: 1404.2188 [cs]. (8. Apr. 2014), Adresse: http://arxiv.org/abs/1404.2188 (besucht am 04. 01. 2023), preprint.

[88] R. Tournaire, J.-M. Petit, M.-C. Rousset und A. Termier, „Discovery of Probabilistic Mappings between Taxonomies: Principles and Experiments“, in *Journal on Data Semantics XV*, Serie Lecture Notes in Computer Science, S. Spaccapietra, Herausgeber, Berlin, Heidelberg: Springer, 2011, Seiten 66–101, ISBN: 978-3-642-22630-4. DOI: 10.1007/978-3-642-22630-4_3. Adresse: https://doi.org/10.1007/978-3-642-22630-4_3 (besucht am 21. 07. 2022).

[89] J. Madhavan, P. Bernstein, AnHai Doan und A. Halevy, „Corpus-Based Schema Matching“, in *21st International Conference on Data Engineering (ICDE'05)*, Tokyo, Japan: IEEE, 2005, Seiten 57–68, ISBN: 978-0-7695-2285-2. DOI: 10.1109/ICDE.2005.39. Adresse: http://ieeexplore.ieee.org/document/1410106/ (besucht am 02. 10. 2018).

[90] Y. R. Jean-Mary, E. P. Shironoshita und M. R. Kabuka, „Ontology matching with semantic verification“, *Journal of Web Semantics*, The Web of Data, Jahrgang 7, Nummer 3, Seiten 235–251, 1. Sep. 2009, ISSN: 1570-8268. DOI: 10.1016/j.websem.2009.04.001. Adresse: https://www.sciencedirect.com/science/article/pii/S1570826809000146 (besucht am 30. 07. 2022).

[91] J. Gracia, J. Bernad und E. Mena, „Ontology matching with CIDER: Evaluation report for OAEI 2011“, in *Proceedings of the Sixth International Workshop on Ontology Matching*, 2011.

[92] K. Nozaki, T. Hochin und H. Nomiya, „Semantic Schema Matching for String Attribute with Word Vectors“, in *2019 6th International Conference on Computational Science/Intelligence and Applied Informatics (CSII)*, Mai 2019, Seiten 25–30. DOI: 10.1109/CSII.2019.00012.

[93] R. Cappuzzo, P. Papotti und S. Thirumuruganathan, „Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks“, in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Serie SIGMOD '20, New York, NY, USA: Association for Computing Machinery, 11. Juni 2020, Seiten 1335–1349, ISBN: 978-1-4503-6735-6. DOI: 10.1145/3318464.3389742. Adresse: https://doi.org/10.1145/3318464.3389742 (besucht am 28. 11. 2021).

[94] J. Euzenat und P. Shvaiko, „Overview of Matching Systems“, in *Ontology Matching*, J. Euzenat und P. Shvaiko, Herausgeber, Berlin, Heidelberg: Springer, 2013, Seiten 201–283, ISBN: 978-3-642-38721-0. DOI: 10.1007/978-3-642-38721-0_8. Adresse: https://doi.org/10.1007/978-3-642-38721-0_8 (besucht am 03. 01. 2022).

[95] C. J. Zhang, Z. Zhao, L. Chen, H. V. Jagadish und C. C. Cao, „CrowdMatcher: Crowd-assisted Schema Matching“, in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Serie SIGMOD '14, New York, NY, USA: ACM, 2014, Seiten 721–724, ISBN: 978-1-4503-2376-5. DOI: 10.1145/2588555.2594515. Adresse: http://doi.acm.org/10.1145/2588555.2594515 (besucht am 11. 10. 2018).

[96] S. Anam, Y. S. Kim, B. H. Kang und Q. Liu, „Adapting a knowledge-based schema matching system for ontology mapping“, in *Proceedings of the Australasian Computer Science Week Multiconference*, Serie ACSW '16, Canberra, Australia: Association for Computing Machinery, 1. Feb. 2016, Seiten 1–10, ISBN: 978-1-4503-4042-7. DOI: 10.1145/2843043.2843048. Adresse: https://doi.org/10.1145/2843043.2843048 (besucht am 03. 02. 2020).

[97] A. Pomp, L. Poth, V. Kraus und T. Meisen, „Enhancing Knowledge Graphs with Data Representatives:“ In *Proceedings of the 21st International Conference on Enterprise Information Systems*, Heraklion, Crete, Greece: SCITEPRESS - Science and Technology Publications, 2019, Seiten 49–60, ISBN: 978-989-758-372-8. DOI: 10.5220/0007677400490060. Adresse: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007677400490060 (besucht am 27. 01. 2020).

[98] O. Allweyer, C. Schorr, R. Krieger und A. Mohr, „Classification of Products in Retail using Partially Abbreviated Product Names Only:“ In *Proceedings of the 9th International Conference on Data Science, Technology and Applications*, Lieusaint - Paris, France: SCITEPRESS - Science and Technology Publications, 2020, Seiten 67–77, ISBN: 978-989-758-440-4. DOI: 10.5220/0009821400670077. Adresse: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0009821400670077 (besucht am 15. 01. 2023).

[99] C. J. Zhang, L. Chen, H. V. Jagadish, M. Zhang und Y. Tong, „Reducing Uncertainty of Schema Matching via Crowdsourcing with Accuracy Rates“, *IEEE Transactions on Knowledge and Data Engineering*, Jahrgang 32, Nummer 1, Seiten 135–151, Jan. 2020, ISSN: 1558-2191. DOI: 10.1109/TKDE.2018.2881185.

[100] OpenAI. „GPT-4 Technical Report“. arXiv: 2303.08774 [cs]. (27. März 2023), Adresse: http://arxiv.org/abs/2303.08774 (besucht am 18. 06. 2023), preprint.

[101] V. D. Warmerdam. „Large Disagreement Modelling“. (2023), Adresse: https://koaning.io/posts/large-disagreement-models/ (besucht am 18. 06. 2023).

[102] J. Tobin. „The Full Stack - Augmented Language Models“. (2023), Adresse: https://fullstackdeeplearning.com/llm-bootcamp/spring-2023/augmented-language-models/ (besucht am 18. 06. 2023).