Automatic Topic Modeling in the Context of Digital Libraries: Mehrsprachige Korpus-basierte Erweiterung von text2ddc - eine experimentelle Studie

Bachelorarbeit

zur Erlangung des akademischen Grades eines

Bachelor of Science

vorgelegt von

Daniel Baumartz

angefertigt am

Lehrgebiet Texttechnologie

der Goethe-Universität Frankfurt am Main

Leiter: Prof. Dr. A. Mehler

23.06.2020

Daniel Baumartz
Informatik Bachelor
Sommersemester 2020

Bachelorarbeit

Automatic Topic Modeling in the Context of Digital Libraries: Mehrsprachige Korpus-basierte Erweiterung von text2ddc - eine experimentelle Studie

Daniel Baumartz

Abgabedatum: 23.06.2020

Lehrgebiet Texttechnologie
der Goethe-Universität Frankfurt am Main
Prof. Dr. Alexander Mehler

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Ebenso bestätige ich, dass diese Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

_____

Ort, Datum

_____

Unterschrift

Zusammenfassung

Diese Bachelorarbeit befasst sich mit der Themenklassifikation von unstrukturiertem Text. Aufgrund der stetig steigenden Menge von textbasierten Daten werden automatisierte Klassifikationsmethoden in vielen Disziplinen benötigt und erforscht. Aufbauend auf dem *text2ddc*-Klassifikator, der am *Text Technology Lab* der Goethe-Universität Frankfurt am Main entwickelt wurde, werden die Auswirkungen der Vergrößerung des Trainingskorpus mittels unterschiedlicher Methoden untersucht. *text2ddc* nutzt die *Dewey Decimal Classification* (DDC) als Zielklassifikation und wird trainiert auf Artikeln der Wikipedia. Nach einer Einführung, in der Grundlagen beschrieben werden, wird das Klassifikationsmodell von *text2ddc* vorgestellt, sowie die Probleme und daraus resultierenden Aufgaben betrachtet. Danach wird die Aktualisierung der bisherigen Daten beschrieben, gefolgt von der Vorstellung der verschiedenen Methoden, das Trainingskorpus zu erweitern. Mit insgesamt elf Sprachen wird experimentiert. Die Evaluation zeigt abschließend die Verbesserungen der Qualität der Klassifikation mit *text2ddc* auf, diskutiert die problematischen Fälle und gibt Anregungen für weitere zukünftige Arbeiten.

Danksagung

An dieser Stelle möchte ich mich ganz herzlich bei meinen Betreuern des *Text Technology Lab* der Goethe-Universität Frankfurt am Main, Herrn Prof. Dr. Alexander Mehler sowie Herrn Tolga Uslu, bedanken. Ein weiteres Dankeschön geht an meine Familie und Freunde, für ihre Unterstützung während meines gesamten Studiums.

Inhaltsverzeichnis

Abkürzungsverzeichnis

DDC *Dewey Decimal Classification* 4

DNB *Deutsche Nationalbibliothek* 4

GND *Gemeinsame Normdatei* 6

KNN künstliches neuronales Netzwerk 2

NLP Natural Language Processing 1

POS Part of Speech 11

TTLab *Text Technology Lab* 1

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Die *Themenklassifikation* (auch *Textklassifikation*, *Topic Classification*, *Text Classification* oder *Document Classification*) ist eine wichtige Aufgabe in aktuellen Bereichen, wie unter anderem den Bibliothekswissenschaften (Lorenz 2018), den Geisteswissenschaften (Goldstone und Underwood 2012), der Biologie (La Rosa u. a. 2015; Liu u. a. 2016) oder der Informatik und ein großes Forschungsfeld in der Computerlinguistik (Natural Language Processing (NLP)) (Blanck, Niekler und Kaulisch 2019; Bracewell u. a. 2009; Lösch u. a. 2011). Diese Disziplinen arbeiten mit immer größer werdenden Sammlungen von unstrukturierten, textbasierten Materialien (Big Text (Sokolova 2018)), wie Büchern, Fachpublikationen, Wikis oder nutzergenerierten Inhalten (User Generated Content). Diese Informationen für Anwendungen wie Text-Zusammenfassungen, Stimmungserkennung, Empfehlungen, E-Mail-Spam-Erkennung (oder allgemeiner Sortierung) oder das Finden von Inhalten (Querying) zugänglich zu machen, verlangt nach immer besseren Möglichkeiten, die Inhalte dieser Korpora zu strukturieren (Aggarwal und Zhai 2013; Shiri 2004). Die automatische Themenklassifikation ermöglicht eine Dimensionsreduktion durch die Einteilung von Texten oder Textsegmenten in Kategorien und stellt somit eine Möglichkeit bereit, auf die Informationen über eine strukturierte und abstrahierte Ebene der Inhalte zuzugreifen (Crain, Zhou und Yang 2012).

Das *Text Technology Lab* (TTLab) der Goethe-Universität Frankfurt am Main hat zu diesem Zweck den mehrsprachfähigen *text2ddc*-Themenklassifikator entwickelt, der in der Lage ist, beliebige Texte thematisch zu kategorisieren. Diese Bachelorarbeit baut auf *text2ddc* auf und erweitert ihn, mit dem Ziel, die Ergebnisse der Klassifikation zu verbessern.

Der Aufbau dieser Ausarbeitung gestaltet sich wie folgt: Der Rest dieses Einleitungskapitels stellt die Themen, Datenquellen, Methoden, Algorithmen, Tools und wichtigen Aspekte vor, die bei der Durchführung der Bachelorarbeit verwendet werden und auf die im weiteren Verlauf aufgebaut werden. Kapitel 2 beschreibt bekannte Klassifikationsmodelle und gibt einen Überblick über verwandte Arbeiten (Related Work) der Themenklassifikation. Der aktuelle Stand des *text2ddc*-Klassifikators und die sich daraus ergebenden Probleme sowie Aufgaben werden in Kapitel 3 dargestellt. Im folgenden Kapitel 4 wird auf die Umsetzung der Aufgaben und Experimente eingegangen, sowie eine Evaluation der Ergebnisse durchgeführt und diskutiert. Eine Zusammenfassung der Arbeit, sowie einen Überblick über mögliche zukünftige Arbeiten (Future Work) und Möglichkeiten zu weiteren Verbesserungen zeigt das letzte Kapitel 5 auf. Der Anhang B dokumentiert die entwickelten Softwarelösungen, mit denen die Daten generiert, sowie die Experimente durchgeführt werden.

1.2. Grundlagen

1.2.1. Themenklassifikation

Die Aufgabe der Themenklassifikation ist definiert als die Zuweisung von *Klassen* (auch Themen, Kategorien, Konzepte, Topics oder Label) zu Textdokumenten (Jacob 2004; McCallum 1999; Shiri 2004). Im einfachsten Fall, der *Single-Label Classification*, wird jedem Dokument aus einem Korpus genau eine Klasse zugewiesen (Bracewell u. a. 2009; Katakis, Tsoumakas und Vlahavas 2008). Besteht die Menge der Klassen aus genau zwei Elementen, wird dieser Spezialfall *Binary Classification* oder *Two-Class Classification* genannt, bei mehr als zwei spricht man von einer *Multi-Class Classification* (Katakis, Tsoumakas und Vlahavas 2008). Kann einem Dokument darüber hinaus mehr als eine Klasse zugewiesen werden, findet eine *Multi-Label Multi-Class Classification* statt, es entsteht somit eine n zu m Zuordnung von Dokumenten zu Themen (Madjarov u. a. 2012). Graph-basierte Klassensysteme können außerdem eine hierarchische Zuordnung ermöglichen (Sokolova 2018).

Eine Klassifikation kann von Experten durchgeführt werden, wie zum Beispiel die Kategorisierung von Büchern in Bibliotheken. Um diesen teuren und zeitaufwändigen (Goller u. a. 2000) Prozess zu vereinfachen, werden in der Computerlinguistik automatisierte Methoden erforscht und verwendet, um Klassifikatoren mittels *maschinellem Lernen* (Machine Learning) zu erstellen. Der grundlegende Gedanke des maschinellen Lernens ist es, Algorithmen und Modelle zu entwickeln, die in der Lage sind, ihre Aktionen anzupassen, also zu lernen, um so die Ergebnisse dieser Aktionen verbessern zu können (Marsland 2014). Dabei lernen diese Modelle im besten Fall nicht auswendig, hier spricht man von Überanpassung (Overfitting), sondern bauen ihr Wissen in einer verallgemeinerten Form auf, indem sie versuchen, Muster in ihren Trainingsdaten zu finden (Marsland 2014). Damit sind sie in der Lage, auch im Training nicht gesehene Daten verarbeiten zu können.

Von besonderem Interesse in dieser Arbeit sind *künstliche neuronale Netzwerke* (Artificial Neural Networks), die die Basis von *text2ddc* bilden. Ein künstliches neuronales Netzwerk (KNN) ist ein Netzwerk, bestehend aus in mehreren Ebenen angeordneten künstlichen, dem menschlichen Gehirn nachempfundenen, Neuronen. Üblicherweise existieren mindestens drei Ebenen, die Eingabeebene (Input Layer), ein oder mehrere versteckte Ebenen (Hidden Layer), sowie die Ausgabeebene (Output Layer) (Vega-Carrillo u. a. 2007). Bei wenigen versteckten Ebenen spricht man von flachen neuronalen Netzwerken (Shallow Learning), bei mehreren von tiefen Netzwerken (Deep Learning).

Es wird zwischen Methoden des *überwachten* (supervised) (Goller u. a. 2000) und *unüberwachten* (unsupervised) (Liu u. a. 2016) Lernen der Klassifikationsmodelle unterschieden. Bei überwachten Modellen kommt ein vordefiniertes Klassensystem zum Einsatz. Dieses könnte zum Beispiel Klassen wie *Politik* oder *Sport* enthalten. Das Modell lernt anhand von ausgewählten Beispielen, wie Texten die passenden Klassen zugewiesen werden. Dazu wird eine Sammlung von Textdokumenten (die *Trainingsmenge*) benötigt, denen bereits passende Klassen zugeordnet wurden. Dies führt zu zwei Phasen, der Lernphase und der Klassifikationsphase (Goller u. a. 2000), siehe auch Abbildung 1.1. Unüberwachte Modelle bestimmen die Menge der möglichen Klassen, in die sie einen Text einordnen, automatisch. Dies erfolgt zum Beispiel mittels einem Wahrscheinlichkeitsmodell basierend auf den verfügbaren Daten

im Textkorpus. Das Klassensystem ist damit abhängig von den Dokumenten, im Gegensatz zu überwachten Modellen, bei denen der Begriff einer Klasse nicht im Dokumenttext auftreten muss (Natarajan u. a. 2007). Eine Klasse könnte zum Beispiel aus Wörtern des Textes, wie *Merkel* und *CDU*, bestehen. Die Bedeutung der Klassen muss bei der Auswertung durch den Anwender interpretiert werden, zum Beispiel als *Politik*. Im Kapitel 2 werden ausgewählte bekannte überwachte und unüberwachte Modelle vorgestellt.

Vordefinierte Klassensysteme können weiterhin unterschieden werden zwischen geschlossenen (*Closed Topic Model*) und offenen (*Open Topic Model*) Systemen (Mehler und Waltinger 2009): Bei geschlossenen Systemen wird die Menge der Kategorien von oft wenigen Experten per Hand festgelegt und bleibt statisch. Dies ermöglicht eine Vergleichbarkeit der Klassifikationen über längere Zeiträume, obwohl auf lange Sicht gesehen auch hier Änderungen möglich sein sollten, um sich an ändernde Gegebenheiten anpassen zu können. Offene Klassensysteme werden hingegen aus *Social Ontologies* abgeleitet, wie zum Beispiel der Wikipedia, und damit durch die unterliegenden, sich häufig ändernden, sozialen Graphen der Nutzer gebildet (Mehler und Waltinger 2009).

Formal gilt nach Shiri (2004) für überwachtes Lernen mittels maschinellem Lernen: Gegeben sei ein Dokument $d \in D$, mit Merkmalen (Features) aus einem Dokumentenraum (Document Space) D, sowie eine Menge an Klassen $C = \{c_1, c_2, ..., c_j\}$. Gesucht ist ein Klassifikator F, der anhand von einer Menge von Trainingsbeispielen $(d, c) \in D \times C$ lernt, Dokumenten Klassen zuzuordnen: $F : D \to C$.

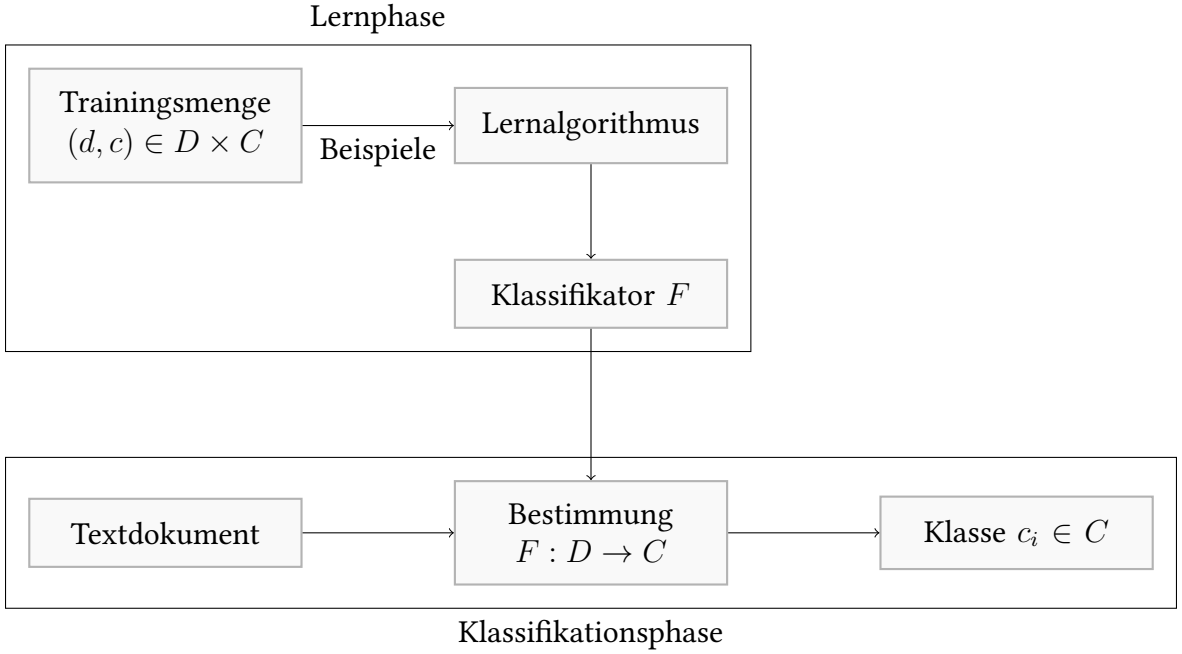

Abbildung 1.1.: Phasen des überwachten Lernens, in Anlehnung an Goller u. a. (2000).

1.2.2. DDC

Die *Dewey-Dezimalklassifikation* (*Dewey Decimal Classification* (DDC)) ist das von *text2ddc* verwendete Ziel-Klassifikationssystem (Baumartz, Uslu und Mehler 2018). Entwickelt von Melvil Dewey in 1876 in den USA, ist sie heute die international am weitesten verbreitete Klassifikation (Alex 2018). Die DDC kommt in 128 Ländern zum Einsatz, davon in über 60 zur Organisation der Nationalbibliothek, und wurde in über 30 Sprachen übersetzt (Mitchell und Vizine-Goetz 2009). Auch die *Deutsche Nationalbibliothek* (DNB) nutzt, unter anderem, die DDC [1]. Obwohl die DDC einer stetigen Weiterentwicklung unterliegt und um neue Klassen erweitert wird (pro Jahr werden einige hundert bis mehrere tausend Klassen überarbeitet (Alex 2018)), ist sie in der Nutzung als Klassifikationssystem als Closed Topic Model anzusehen (siehe 1.2.1).

Die Klassen in der DDC sind hierarchisch geordnet, von allgemein hin zu spezifisch. Auf der ersten Ebene besteht die DDC aus den folgende 10 Hauptklassen (Alex 2018; OCLC 2019):

000 Computer science, information & general works (Informatik, Informationswissenschaft, allgemeine Werke)

100 Philosophy & psychology (Philosophie und Psychologie)

200 Religion (Religion)

300 Social sciences (Sozialwissenschaften)

400 Language (Sprache)

500 Science (Naturwissenschaften und Mathematik)

600 Technology (Technik, Medizin, angewandte Wissenschaften)

700 Arts & recreation (Künste und Unterhaltung)

800 Literature (Literatur)

900 History & geography (Geschichte und Geographie)

Die Hauptklassen sind in zehn weitere Unterklassen (wie 330 Wirtschaft oder 940 Geschichte Europas) aufgeteilt, gefolgt von zehn weiteren Unterklassen auf der dritten Ebene (wie 321 Staatsformen und Regierungssysteme). Diese obersten, insgesamt 1 000 Klassen werden als DDC-Übersichten (DDC-Summaries, oder auch Haupttafeln) bezeichnet (Alex 2018). Weitere, immer spezifischere Klassen erweitern die DDC-Notation um zusätzliche angehängte Ziffern, getrennt durch einen Punkt (Alex 2018). Verschiedene Hilfstafeln (zum Beispiel T2 Geographic Areas) (Mitchell und Vizine-Goetz 2009) sowie, nach bestimmten Regeln erstellte, künstliche Notationen (OCLC 2019) ermöglichen eine sehr spezifische und genaue Klassifikation sämtlicher Sachverhalte.

[1] `https://www.dnb.de/DE/Professionell/DDC-Deutsch/ddc-deutsch_node.html`

Die DDC wird verwaltet und weiterentwickelt durch die OCLC (Online Computer Library Center)[2], die deutsche Übersetzung wurde an der DNB und der Fachhochschule Köln durchgeführt (Alex 2018).

Aufgrund ihres Ursprungs und Alters ist die DDC nicht frei von Kritik, besonders die teilweise ungleiche Verteilung ihrer Klassen (zum Beispiel übersteigt die Anzahl der Klassen zum Thema Christentum die aller anderen Religionen deutlich) oder ihrer Tiefe (zum Beispiel beim Thema Informatik) werden in Alex (2018) genannt.

In *text2ddc* kommen die bis zu 100 Klassen (nicht alle Notationen sind besetzt) der zweiten, und die bis zu 1 000 Klassen der dritte Ebene der DDC als Zielklassifikation zum Einsatz.

1.2.3. Wikipedia und Wikidata

Wikipedia[3] ist eine große, frei[4] verfügbare Online-Enzyklopädie, deren Artikel nach dem Prinzip des kollaborativen Schreibens von Freiwilligen verfasst werden (Wikipedia 2020). Auch wenn die soziale Dynamik zu einer Ungleichverteilung der Informationen führt (*skalenfreies Lexikon*) (Mehler, Gleim u. a. 2017), stellt Wikipedia eine wichtige Ressource, sowohl für die Forschung als auch als Datenquelle, dar (Gleim, Mehler und Song 2018; Uslu, Mehler, Baumartz u. a. 2018). Auch der *text2ddc* Klassifikator nutzt Artikel aus der Wikipedia als Lernkorpus.

Die Inhalte der Wikipedia sind nach Sprachen getrennt und organisiert in Artikeln, die jeweils einen lexikalischen Eintrag beschreiben (Wikipedia 2020). Sofern Artikel in mehreren Sprachen vorliegen, findet innerhalb der verschiedenen Sprachversionen eine Verlinkung statt. Jedem Artikel können beliebige Kategorien zugeordnet werden. Da diesen wiederum weitere Kategorien untergeordnet sein können, lässt sich aus allen Kategorien ein Wikipedia-Kategoriegraph erstellen. Das Kategoriesystem der Wikipedia kann somit ein Open Topic Model induzieren, siehe 1.2.1.

Eng verbunden mit der Wikipedia ist das Wikidata[5] Projekt, das sich selbst als „freie[6] und offene Wissensdatenbank“ (Wikidata 2020) beschreibt. Während die Artikel in Wikipedia aus zumeist unstrukturiertem Text bestehen, enthält Wikidata eine strukturierte Datenbank, bestehend aus Elementen (Items, dargestellt mittels einer eindeutigen ID beginnend mit `Q` gefolgt von einer Nummer), denen über Eigenschaften (Properties) verschiedene Aussagen (Statements) zugeordnet sind, wie etwa eine Beschreibung (Voß u. a. 2014). Dies ermöglicht eine Darstellung der Daten nach dem standardisierten RDF-Modell[7]. Wie die Wikipedia-Sprachversionen sind auch die Wikidata-Items mit ihren dazugehörigen Wikipedia-Artikeln verbunden. Darüber hinaus existieren viele Verlinkungen zu anderen Datenbanken und Ontologien, wie der DDC oder der GND.

[2] `https://www.oclc.org/en/dewey.html`
[3] `https://www.wikipedia.org`
[4] `https://creativecommons.org/licenses/by-sa/3.0/legalcode`
[5] `https://www.wikidata.org`
[6] `https://creativecommons.org/publicdomain/zero/1.0/`
[7] `https://www.w3.org/RDF/`

1.2.4. GND

Die *Gemeinsame Normdatei* (GND)[8] wird an der DNB entwickelt und verwaltet, unter anderem mit Beteiligung von allen Bibliotheksverbünden deutschsprachiger Länder (Behrens-Numann und Pfeifer 2011). „Diese Normdaten repräsentieren und beschreiben Entitäten, also Personen, Körperschaften, Konferenzen, Geografika, Sachbegriffe und Werke, die in Bezug zu kulturellen und wissenschaftlichen Sammlungen stehen." (Deutsche Nationalbibliothek 2020)

Die Einträge in der GND enthalten unter anderem auch Zuordnungen zu DDC-Klassen, angelegt innerhalb des CrissCross-Projektes (Alex 2018) und sind über Wikidata verlinkt.

1.2.5. Evaluation

Üblicherweise werden die Ergebnisse beim überwachten Lernen mittels Metriken wie *Accuracy*, *Precision*, *Recall* und *F_1-score* evaluiert, die die Güte eines Klassifikators auf einer Skala zwischen 0 und 1 beschreiben (Lewis 1991; Sorower 2010). Diese Metriken basieren auf einer *Wahrheitsmatrix* (Confusion Matrix), die aus den Testdaten generiert wird und jedes Beispiel in eine der folgenden vier Felder einteilt (Lewis 1991), im einfachsten Fall einer binären Klassifikation bestehend aus den Klassen A und $\neg A$:

- *Richtig positiv* (True Positive, TP) sind Beispiele der Klasse A, die korrekt der Klasse A zugeordnet wurden.
- *Falsch negativ* (False Negative, FN) sind Beispiele der Klasse A, die fälschlicherweise als zugehörig zu Klasse $\neg A$ erkannt wurden.
- *Falsch positiv* (False Positive, FP) sind Beispiel der Klasse $\neg A$, die fälschlicherweise in Klasse A einsortiert wurden.
- *Richtig negativ* (True Negative, TN) sind Beispiel der Klasse $\neg A$, denen korrekt die Klasse $\neg A$ zugewiesen wurde.

Bei n Beispielen und den durch einen Klassifikator für Beispiel i ermittelten Klassen ergibt sich nach Ikonomakis, Kotsiantis und Tampakas (2005):

Accuracy ist der Anteil der korrekten Entscheidungen eines Klassifikators.

$$A = \frac{1}{n} \sum_{i=1}^{n} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

Precision beschreibt den Anteil der korrekt klassifizierten Beispiele aus allen ermittelten Klassen und zeigt damit, wie viele Fehler ein Klassifikator verursacht.

$$P = \sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i}$$

[8] https://www.dnb.de/DE/Professionell/Standardisierung/GND/gnd.html

Recall ist der Anteil der korrekt erkannten Klassen von allen Klassen, die hätten erkannt werden sollen, und beschreibt damit, wie vollständig ein Klassifikator arbeitet.

$$R = \sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i}$$

F_1-score ist das harmonische Mittel (Harmonic Mean) von Precision und Recall und kommt zum Einsatz, um die Güte eines Klassifikators in einer Zahl auszudrücken.

$$F_1 = 2\frac{PR}{P + R}$$

Bei der *Multi-Label Multi-Class* Klassifikation, die in dieser Bachelorarbeit zum Einsatz kommt, kann ein Klassifikationsergebnis darüber hinaus auch nur teilweise korrekt sein (Sorower 2010). Der *text2ddc*-Klassifikator verwendet deshalb zur Evaluation die *Precision an Position 1* Metrik (*P@1*, auch *Top-1-Accuracy* (Natarajan u. a. 2007)), da für die Themenklassifikation hauptsächlich von Interesse ist, ob das wahrscheinlichste Thema korrekt klassifiziert wurde.

1.2.6. Verwendete Tools und Software

Zur Entwicklung der Software dieser Bachelorarbeit wurde auf die folgenden Programmier- und Abfragesprachen zurückgegriffen, siehe auch Anhang B für eine Dokumentation:

- SPARQL, eine Abfragesprache, die im Linked Data- und RDF-Umfeld häufig verwendet wird, dient dem Abrufen der Wikidata- und GND-Daten.
- Die Programmiersprache C++ wird eingesetzt, um den Wikipedia-Kategoriebaum zu verarbeiten, unter Verwendung der LEMON-Bibliothek[9].
- Verschiedene Bash-Skripte übernehmen unterschiedliche wiederkehrende Aufgaben, sowie die Steuerung des Lernprozesses.
- *fastText* (Joulin u. a. 2017) wird verwendet, um den Klassifikator zu trainieren.
- Die restliche Software, wie zum Beispiel das Erstellen der Trainingsdaten, wird in Python entwickelt. Hier kommen unter anderem die Bibliotheken scikit-learn (Pedregosa u. a. 2011), NLTK (Bird, Loper und Klein 2009) und numpy (van der Walt, Colbert und Varoquaux 2011) zum Einsatz.
- Blazegraph[10] kommt als Datenbank zum Einsatz, mit der im *TTLab* bereits gute Erfahrungen beim Nutzen großer Datenmengen wie der Wikidata gemacht wurde
- Zur Erstellung einiger Graphen und Statistiken wird R (R Core Team 2020) verwendet, mit ggplot2 (Wickham 2016) sowie dplyr (Wickham u. a. 2020).

[9] `https://lemon.cs.elte.hu/trac/lemon`
[10] `https://blazegraph.com`

2. Verwandte Arbeiten

Dieses Kapitel stellt kurz einige bekannte Topic Models und Klassifikationsalgorithmen vor, sowie weitere verwandte Arbeiten, die sich mit der Themenklassifikation, und im speziellen der Klassifikation mit der DDC, befassen.

Eines der bekanntesten Topic Models ist die *Latent Dirichlet Allocation* (LDA) (Blei, Ng und Jordan 2003). Die grundlegende Annahme der LDA ist, dass ähnliche Themen in verschiedenen Dokumenten eines Korpus durch ähnliche Worte beschrieben werden. Damit kann ein Thema über eine Wahrscheinlichkeitsverteilung von Wörtern dargestellt werden. LDA berechnet dann ein Modell, bestehend aus Wahrscheinlichkeiten und Wörtern, die ein Thema ausmachen, mit einer durch den Nutzer von außen festgelegten Anzahl von Themen. Dieses Modell wird so gebildet, dass die Wahrscheinlichkeiten der Wörter eines Themas das Auftreten derselben Wörter in den Dokumenten erklärt.

Mit *Supervised LDA* (sLDA) (Blei und McAuliffe 2010) und *Labeled LDA* (L-LDA) (Ramage u. a. 2009) existieren zwei unterschiedliche Arten von Supervised Topic Models, die auf der LDA basieren. Während sLDA auf eine Klasse pro Dokument beschränkt ist, unterstützt L-LDA mehrere und konnte teilweise die Ergebnisse des überwachten Modells der SVM übertreffen (Quercia, Askham und Crowcroft 2012).

Support Vector Machines (SVM) (Cortes und Vapnik 1995) versuchen, Trainingsdaten zweier Klassen, dargestellt als Vektoren in einem Vektorraum, möglichst stark zu separieren. Durch ein Mapping der Vektoren in einen hochdimensionalen Raum können diese Vektoren linear mittels einer Ebene (Hyperplane) getrennt werden. Ursprünglich für binäre Klasifikationsaufgaben entwickelt, ermöglichen Implementationen wie in scikit-learn (Pedregosa u. a. 2011) das Trainieren von mehreren Klassifikatoren mit je zwei Klassen, um eine *Multi-Class* Klassifikationen zu ermöglichen.

Eine für viele Klassen effiziente Methode ermöglichen *Random Forest-Klassifikatoren* (Breiman 2001), die eine Menge von unabhängigen Entscheidungsbäumen (Decision Tree) trainieren, mit jeweils einer zufälligen Auswahl von Merkmalen. Die Ergebnisse dieses Waldes von Klassifikatoren werden dann zusammengefasst und ausgewertet, um die Klassen eines Beispiels zu bestimmen (Ensemble Learning). Innerhalb dieser Bachelorarbeit wird ein Random Forest-Klassifikator eingesetzt, sowie für das Training des *text2ddc*-Klassifikators *fastText* (Joulin u. a. 2017). Dessen große Vorteile sind die Geschwindigkeit des Trainingsvorgangs, die durch ein flaches künstliches neuronales Netz erreicht wird. Trotzdem erreichen die Klassifikatoren eine ähnliche Qualität wie tiefe KNNs.

Forschung und Entwicklung zur Themenklassifikation werden bereits lange durchgeführt. So erschien 1999 ein Artikel, der die manuelle Klassifikation von Zeitungen mit der DDC untersucht (Kuhn 1999), und bereits ein Jahr früher wurde eine automatisierte hierarchische DDC-Klassifikation mittels eines Thesaurus vorgestellt (Jenkins u. a. 1998). Eine weitere Veröffentlichung setzt auf überwachtes Lernen mittels SVM, basierend auf einem stärker balancierten und flacheren DDC-Baum, unterstützt durch eine interaktive Auswahl der Er-

gebnisse (Wang 2009). Die Klassifikation von Metadaten (Titel, Untertitel und Keywords) der Einträge des Swedish National Union Catalogue betrachtet Golub, Hagelbäck und Ardö (2020), im Besonderen konnte hier eine deutliche Steigerung der Qualität festgestellt werden, als die Anzahl Trainingsdaten pro Klasse erhöht wurde.

Viele weitere Arbeiten und Systeme verwenden nicht die DDC als Zielklassifikation, sondern beispielsweise die Universal Decimal Classification (Möller u. a. 1999), oder Freebase-Objekte und die ihnen zugehörige Wikipedia-Artikel (Husby und Barbosa 2012). Baykan u. a. (2009) klassifizieren Webseiten anhand ihrer URL mittels SVM über die Kategorien des Open Directory Project (ODP/DMOZ). Ähnlich dem Ansatz in dieser Bachelorarbeit wurde in Nigam u. a. (2000) gezeigt, dass Klassifikationsergebnisse verbessert werden können, indem ein trainierter Klassifikator in einer Art Bootstrapping genutzt wird um neue, bisher nicht-annotierte, Daten für ein weiteres Training zu gewinnen.

3. Modell und Architektur

Dieses Kapitel beschreibt den *text2ddc*-Klassifikator und stellt die Aufgaben der Bachelorarbeit, die sich aus dem aktuellen Stand, sowie den Problemen von *text2ddc* ergeben, vor.

Vereinfacht beschrieben ermöglicht *text2ddc* eine thematische Klassifikation der Inhalte von beliebigem, unstrukturiertem Text. Aktuell werden neben Deutsch, Englisch, Französisch, Spanisch, Arabisch und Türkisch insgesamt 40 Sprachen unterstützt. Als Zielklassifikation verwendet *text2ddc* die zweite und dritte Ebene der DDC-Übersichten (siehe Abschnitt 1.2.2) und bietet damit zwei verschieden detaillierte Klassifikationsmöglichkeiten. Trainiert wurde der Klassifikator auf Artikeln der Wikipedia.

Erfolgreich zum Einsatz kam *text2ddc* bereits unter anderem zur thematischen Analyse und Visualisierung literarischer Werke (Mehler, Uslu u. a. 2019), im *text2City*-Projekt, welches Texte in der virtuellen Realität visualisiert (Kett 2020) und zur Bestimmung von Themen während der automatischen Annotation von Korpora in *Corpus2Wiki* (Hunziker u. a. 2019).

Eine vom *TTLab* bereitgestellte Webseite[1] ermöglicht eine einfache Nutzung von *text2ddc*, dazu ist der Klassifikator über den *TextImager* (Hemati, Uslu und Mehler 2016)[2] verfügbar.

3.1. Klassifikationsmodell

text2ddc ist ein auf einem flachen künstlichen neuronalen Netz (Neural Network) basierter Klassifikator (Uslu, Mehler und Baumartz 2019; Uslu, Mehler, Niekler u. a. 2018). Zum Trainieren des Klassifikators kommt die effiziente *fastText*-Bibliothek, entwickelt von Facebook Research, zum Einsatz (Joulin u. a. 2017). Deren Vorteile liegen in der Geschwindigkeit[3] bei gleichzeitiger Performance auf dem Niveau von tiefen KNN (Deep Learning Networks) (Joulin u. a. 2017). Basierend auf dem *Bag of Words*-Modell verwendet *text2ddc* Wörter als Eingabe und produziert über nur eine versteckte Ebene mittels der *Softmax Loss Function* (Joulin u. a. 2017) eine Wahrscheinlichkeitsverteilung in Form eines Vektors als Ausgabe, den *Labeled Topic Vector* (Baumartz, Uslu und Mehler 2018) (siehe Abbildung 3.1). Dessen Elemente bestehen aus den Themenklassen der DDC und geben die Wahrscheinlichkeit einer Zugehörigkeit der Eingabe zu jeder Klasse an.

Die zum Training benötigten Korpora beinhalten Wikipedia-Artikel, denen über die Verlinkung mit Wikidata und der GND sichere DDC-Notationen zugeordnet werden können. Das vom *TTLab* entwickelte NLP-Tool *TextImager* (Hemati, Uslu und Mehler 2016) kommt

[1] Die Übersichtsseite ist unter `https://www.texttechnologylab.org/applications/text2ddc/` erreichbar und die Web-UI unter `https://textimager.hucompute.org/DDC/`.

[2] UIMA-basierte (Ferrucci und Lally 2004) Annotationspipeline, mit einer Vielzahl von integrierten NLP-Tools sowie Visualisierungen, zum Beispiel diversen R-Bibliotheken (Uslu, Hemati u. a. 2017).

[3] *fastSense*, das ebenfalls auf *fastText* basiert, disambiguiert die komplette Wikipedia in nur 20 Minuten, während andere Tools hochgerechnet Wochen bis Monate für diese Aufgabe benötigen (Uslu, Mehler, Baumartz u. a. 2018).

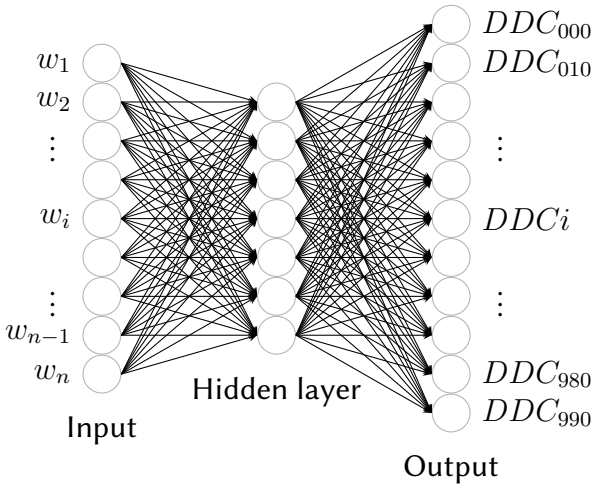

Abbildung 3.1.: Architektur des Modells (Uslu, Mehler, Niekler u. a. 2018, S. 6)

zum Einsatz, um die Texte vorzuverarbeiten (Lemmatisierung, Part of Speech (POS)-Tagging, Word Sense-Disambiguation (Uslu, Mehler, Baumartz u. a. 2018)), dazu werden mittels des Tools *word2vec* (Mikolov u. a. 2013) berechnete Embeddings verwendet.

3.2. Aktueller Stand

Aktuell erreicht der *text2ddc*-Klassifikator für deutsche Texte einen F_1-score von 87,4 % auf der zweiten DDC-Ebene und 78,1 % auf der dritten Ebene (siehe Tabelle 3.1 für die Ergebnisse der anderen Sprachen) (Uslu, Mehler und Baumartz 2019). Trainiert wurde auf den deutschen Wikipedia-Artikeln, denen DDC-Notationen zugeordnet wurden. Dies war möglich für die Anzahl von 15 136 Artikeln, mit 1 228 Token pro Artikel im Durchschnitt und 98 Klassen auf der zweiten und 641 auf der dritten DDC-Ebene (Baumartz, Uslu und Mehler 2018).

Das Trainingskorpus wurde lemmatisiert, von Funktionswörtern befreit und um folgende Features erweitert: Die Namen der Kategorien jedes Wikipedia-Artikels, Wikidata-Statements, Word-Embeddings, sowie mittels *fastSense* (Uslu, Mehler, Baumartz u. a. 2018) disambiguierte Wörter um Doppeldeutungen aufzulösen (Uslu, Mehler und Baumartz 2019). Bei den weiteren Sprachen kamen vereinfachte Trainingsdaten zum Einsatz, die auf Token sowie Word-Embeddings und Wikipedia-Kategorien basieren und durch die verlinkten Übersetzungen der deutschen Artikel gesammelt wurden (Uslu, Mehler und Baumartz 2019). Nach mehreren Parameterstudien stellten sich die folgenden Hyperparameter als optimal heraus: Eine *Learning Rate* (`lr`) von 0,2 mit einer *Update Rate* (`lrUpdateRate`) von 150, Wörter die seltener als 5 mal im Korpus zu finden sind werden entfernt (`minCount`), sowie 10 000 Trainingsiterationen (`epoch`) (Uslu, Mehler und Baumartz 2019).

Bei Experimenten nach Veröffentlichung von Uslu, Mehler und Baumartz (2019) konnte das *TTLab* den F_1-score von 87,4 % auf der zweiten DDC-Ebene halten und den Score der dritten Ebene von 78,1 % auf 79,9 % verbessern. Dies gelang über einen Trainingskorpus mit den folgenden Features:

Tabelle 3.1.: Aktuelle F_1-scores, sowie Anzahl der Klassen der zweiten und dritten DDC-Ebenen des *text2ddc*-Klassifikators.

	DDC Ebene 2		DDC Ebene 3	
Sprache	**Score**	**Klassen**	**Score**	**Klassen**
Deutsch	0,874	98	0,799	635
Englisch	0,854	97	0,726	624
Aragonesisch	0,818	80	0,772	412
Spanisch	0,797	94	0,705	599
Französisch	0,857	94	0,681	604
Italienisch	0,853	93	0,738	598
Limburgisch	0,697	80	0,653	394
Polnisch	0,837	95	-[1]	610
Russisch	0,853	96	0,761	612
Serbokroatisch	0,820	94	0,719	559
Urdu	0,351	82	0,234	455

[1] Da bisher für die polnische Sprachvariante auf der dritten Ebene der DDC kein Klassifikator vorlag, wurde dieser im Rahmen dieser Bachelorarbeit trainiert, dabei konnte ein Score von 0,752 erreicht werden.

- Lemmatisierung
- POS-Tagging
- Entfernung von Satzzeichen
- Entfernung von Funktionswörtern
- Wikipedia Kategorienamen
- Word-Embeddings

3.3. Probleme und Aufgaben

Zum Trainieren von auf künstlichen neuronalen Netzwerken basierten Klassifikatoren werden viele, möglichst gut annotierte Beispiele für jede Klasse benötigt (Golub, Hagelbäck und Ardö 2020; Zelikovitz 2004). Während in dem Trainingskorpus der zweiten DDC-Ebene bei 98 Klassen im Durchschnitt 183,76 Datensätze zur Verfügung stehen, liegt dieser Wert bei der dritten Ebene mit 32,56 deutlich darunter. Noch stärker ist dies bei den weiteren Sprachvarianten zu bemerken, da hier viele Trainingsbeispiele im Laufe der Übersetzung verlorengehen.

Auch die Verteilung der Beispiele pro Klasse ist entscheidend, siehe Abbildung 3.2. Da die Y-Achse logarithmiert dargestellt wird, werden insgesamt 7 Klassen mit einer Anzahl von 0 Beispielen angezeigt: Eine in der DDC unbesetzte Klasse, eine im Korpus fehlende, sowie fünf mit jeweils nur einem Text. Insgesamt stehen somit für 12 Klassen weniger als 10 Beispiele zum Trainieren bereit.

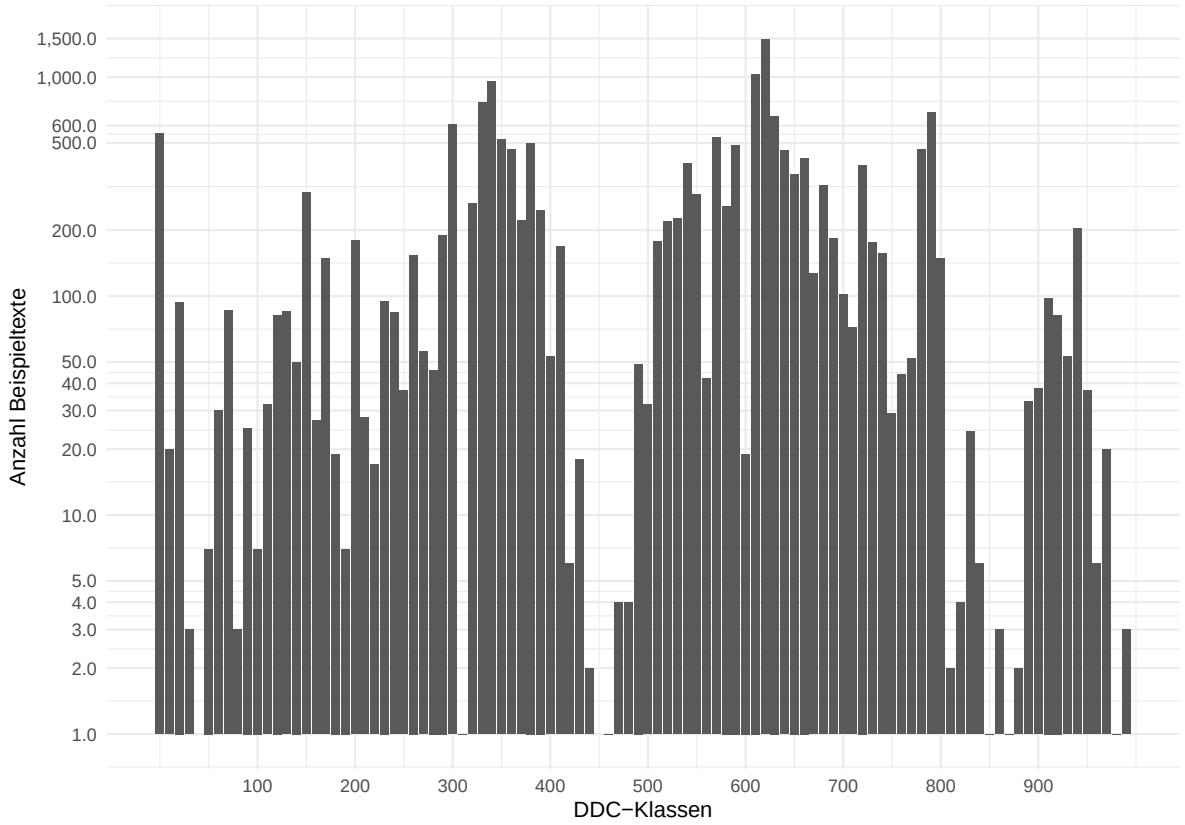

Abbildung 3.2.: Die Grafik zeigt die Verteilung der Beispielstexte der zweiten DDC Ebene. Die Y-Achse ist logarithmiert, dies führt dazu, dass insgesamt 7 Klassen eine Anzahl von 0 aufweisen: Eine unbesetzte Klasse, eine fehlende und fünf mit nur einem Text.

In starker Abhängigkeit von der Größe des Trainingskorpus steht auch die Anzahl an DDC-Klassen, die dieser abdeckt. Während die deutsche Version von *text2ddc* mit insgesamt 98 Klassen der zweiten Ebene alle bis auf eine beinhaltet, ist dieser Wert auf der dritten Ebene mit 635 deutlich geringer.

Diese Bachelorarbeit untersucht deshalb, als Reaktion auf diese Probleme, die Auswirkungen einer Vergrößerung des Trainingskorpus auf die Ergebnisse der Klassifikation. Dabei ergeben sich folgende Aufgaben:

1. **Aktualisierung** der über Wikidata und die GND abgerufenen DDC-Notationen, mit dem Ziel, direkt von den Änderungen in diesen Datenquellen zu profitieren. Siehe dazu Kapitel 4.1 für eine ausführliche Beschreibung.

2. **Vergrößern des Trainingskorpus** mittels verschiedener Methoden, die in Kapitel 4.2 vorgestellt werden. Da nicht alle Wikipedia-Artikel DDC-annotiert vorliegen, sollen diese Annotationen algorithmisch erstellt werden.

3. **Trainieren des *text2ddc*-Klassifikators** basierend auf dem erweiterten Trainingskorpus, siehe Kapitel 4.3.

4. **Evaluation der Ergebnisse**, durchgeführt in Kapitel 4.4. Dabei wird die Frage diskutiert, ob eine Verbesserung der Güte des Klassifikators festzustellen ist, und falls nicht, wo die Ursachen dafür liegen könnten.

4. Umsetzung und Experimente

In diesem Kapitel wird die Umsetzung der insgesamt vier Aufgaben (siehe Kapitel 3.3) beschrieben. Im ersten Teil wird die Aktualisierung des sogenannten *True Corpus* erklärt, dies ist das Korpus bestehend aus sicher DDC-annotierten Wikipedia-Artikeln, bezogen über Wikidata sowie die GND, da von Menschen per Hand erstellt. Es folgt die Vorstellung der verschiedenen Methoden, wie das Trainingskorpus, über die Aktualisierung herausgehend, erweitert wird. Mit der Erläuterung des Trainingsvorgangs des Klassifikators, sowie der Evaluation der Ergebnisse und einem Fazit schließt dieses Kapitel.

Die Grafik 4.1 bildet eine Übersicht des gesamten Prozesses ab.

4.1. Aktualisierung der Daten

Der erste Schritt der Umsetzung bestand aus dem Aktualisieren der bisher von *text2ddc* verwendeten Daten. Dies hatte vor allem zum Ziel, die inzwischen deutlich veralteten Wikidata-Daten zu erneuern, die bisher aus einem Dump der Wikidata vom 10.08.2015 gewonnen wurden. Aus diesem älteren Dump konnten für die 15 136 Artikel der deutschen Sprachversion der Wikipedia insgesamt 25 955 DDC-Notationen gefunden werden, davon 98 einzigartige auf der zweiten Ebene sowie 641 auf der dritten Ebene.

In Kapitel 3.2 wurde bereits ein kurzer Überblick über die Erstellung des *text2ddc*-Korpus gegeben, die folgenden Abschnitte beschreiben diesen Prozess nochmals ausführlicher. Zusammengefasst werden im ersten Schritt alle DDC-Notationen aus der GND extrahiert (siehe 4.1.1), dies führt zu einem Mapping von GND-Einträgen auf DDC-Klassen. Um diese auf einen Wikipedia-Artikel zurückzuführen, wird die Wikidata verwendet, deren Elemente oft sowohl auf Wikipedia-Artikel als auch auf GND-Einträge verlinken (siehe 4.1.2). Da auch Wikidata in einigen Fällen direkte DDC-Notationen bereitstellt, werden diese ebenfalls integriert. Weitere Informationen über Wikipedia-Artikel werden verwendet, um die Übersetzungen in verschiedene Sprachen zu ermöglichen. Die gesammelten Daten werden zusammengeführt, sodass am Ende des Prozesses eine direkte Zuordnung von DDC- Klassenannotationen zu Wikipedia-Artikeln steht (siehe 4.1.4). Dies wird im folgenden als *True Corpus* bezeichnet, da diese Sammlung an Wikipedia-Artikel ausschließlich mit sicher bestätigten, korrekten DDC-Klassen annotiert ist.

4.1.1. GND

Die Datensätze in der GND können eine Angabe zu einer oder mehreren DDC-Klassen enthalten, diese sind unter dem Parameter `DDC-Notation` zu finden. Zum Beispiel ist dem Eintrag zu dem Begriff *Algebraische Topologie* [1] die Notation 514.2, also *Algebraische Topologie* zugeordnet:

[1] `http://d-nb.info/gnd/4120861-4`

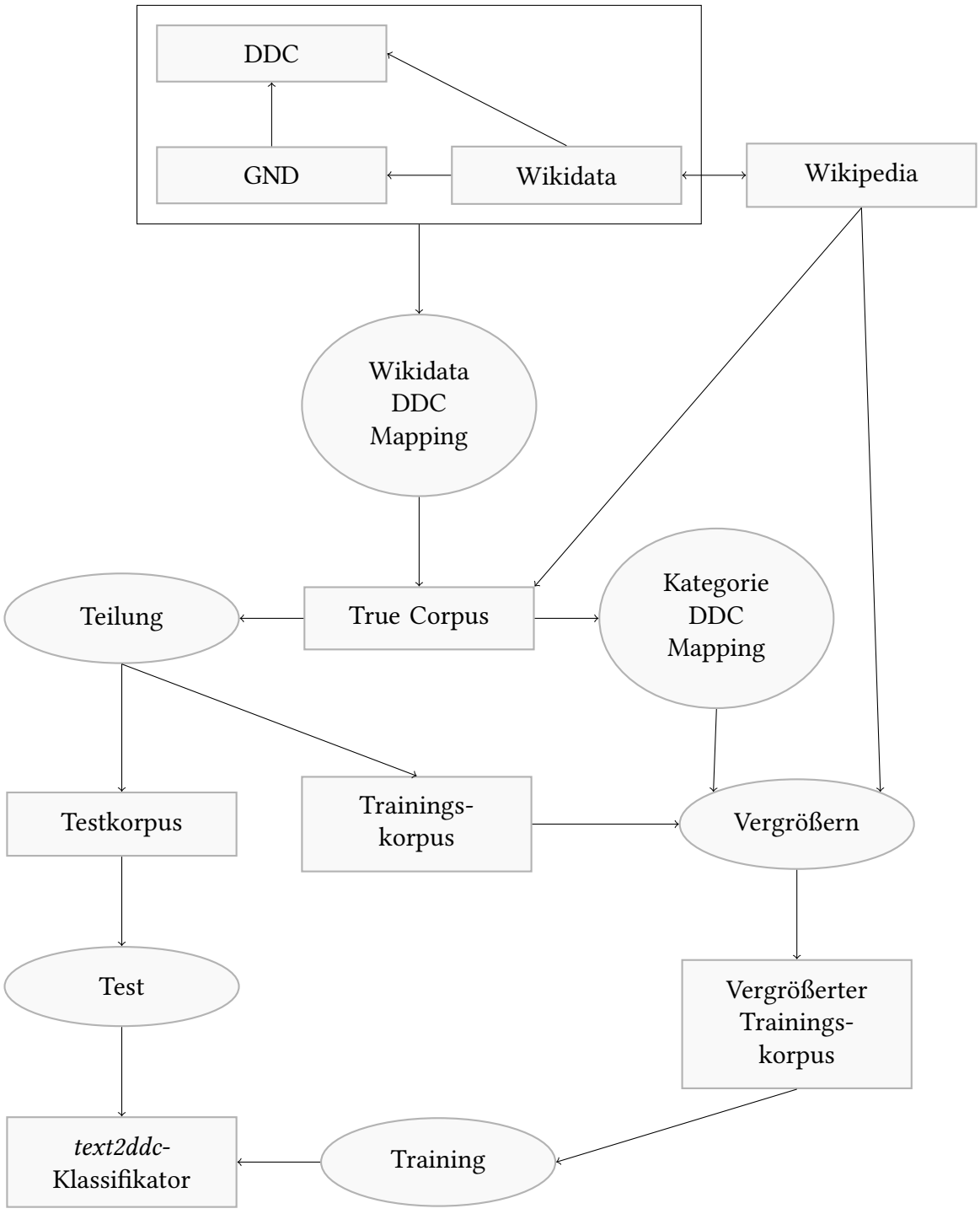

Abbildung 4.1.: Schematische Darstellung des gesamten Prozesses, bestehend aus der Aktualisierung des *True Corpus*, der Vergrößerung des Trainingskorpus und dem Training und Testen des *text2ddc*-Klassifikators.

```
500 Naturwissenschaften
└── 510 Mathematik
    └── 514 Topologie
        └── 514.2 Algebraische Topologie
```

Unter Hinzunahme der Ziffern nach dem Dewey-Punkt ist die Klassifikation bereits sehr spezifisch. *text2ddc* und damit auch diese Bachelorarbeit konzentrieren sich ausschließlich auf die zweite und dritte Ebenen der DDC, sodass wir im vorangegangenen Beispiel die Notation *Mathematik* auf der zweiten, und *Topologie* auf der dritten Ebene erhalten.

Um die Daten zu exportieren wurde auf den Dump im RDF-Format[2] der GND-Datenbank vom 13.06.2019 zurückgegriffen. Dieser wurde in eine lokale Blazegraph-Datenbank geladen und mittels eines Python-Programms über den SPARQL-Endpoint der Datenbank abgefragt. Im RDF-Format sind die DDC-Notationen zu einem Eintrag über das Prädikat `relatedDdcWithDegreeOfDeterminacy*`[3] zugänglich. Dabei werden vier verschiedenen Determiniertheitsgrade unterschieden, die die Vollständigkeit der Abdeckung des DDC-Konzepts beschreiben. Für das *text2ddc*-Korpus wurden alle vier Grade verwendet.

Der RDF-Eintrag des Beispiels liegt im RDF-Dump in folgender Form vor:

```
<https://d-nb.info/gnd/4120861-4> <https://d-nb.info/standards/elementset/gnd#
    ↪ relatedDdcWithDegreeOfDeterminacy3> <http://dewey.info/class/514.2/>
```

Alle Elemente, die mit einer DDC-Notation versehen sind, lassen sich damit über den folgenden, etwas vereinfachten, SPARQL-Query (siehe 4.1) abfragen:

```
SELECT ?gnd ?ddc
WHERE
{
  ?gnd <http://d-nb.info/standards/elementset/gnd#
    ↪ relatedDdcWithDegreeOfDeterminacy1> ?ddc .
}
```

Listing 4.1: SPARQL-Query zur Abfrage der GND-Elemente mit DDC-Notation.

Insgesamt wurden so 156 123 Einträge exportiert und 217 458 Zuordnungen zu DDC-Klassen gefunden. Unter Berücksichtigung aller Ebenen wurden 36 243 unterschiedliche DDC-Notationen mit im Durchschnitt 1,3 DDC-Zuordnungen pro GND-Eintrag gesammelt.

4.1.2. Wikdiata

Aus Wikidata werden insgesamt vier Informationen verwendet: GND-Zuordnungen, DDC-Zuordnungen, Wikipedia-Artikel sowie Wikipedia-Kategorien. Letztere Daten werden erst in Kapitel 4.2 verwendet, die Übrigen dienen dem Zusammenstellen des *True Corpus*.

[2] `https://data.dnb.de/opendata/`

[3] Vollständig ausgeschrieben:
`<http://d-nb.info/standards/elementset/gnd#relatedDdcWithDegreeOfDeterminacy1>`

Aufgrund der Größe der Abfragen und der Einschränkungen der offiziellen Wikidata-API, dessen Ausführungszeit eines Queries auf 60 Sekunden[4] beschränkt ist, wurde eine lokale Kopie der Wikidata eingesetzt. Dazu wurde eine Blazegraph-Datenbank mit dem Wikidata-Dump[5] vom 21.10.2019 aufgesetzt. Die im Folgenden beschriebenen Daten wurden mittels eines Python-Skriptes über die SPARQL-Schnittstelle der Datenbank geladen.

DDC

Viele Objekte in Wikidata verfügen bereits über eine direkte Zuordnung zu einer DDC-Notation über das Prädikat (hier in Wikidata Property genannt) `P1036` (Dewey Decimal Classification)[6]. So ist dem Wikidata-Eintrag zur *Colon-Klassifikation*[7] die DDC-Notation 025.435 zugeordnet:

```
000 Informatik, Informationswissenschaft & allgemeine Werke
└─ 020 Bibliotheks- & Informationswissenschaften
   └─ 025 Tätigkeiten und Betriebsabläufe in Bibliotheken, Archiven,
   │      Informationszentren
   └─ 025.4 Inhaltsanalyse und thematische Kontrolle
      └─ 025.43 Universalklassifikationssysteme
         └─ 025.435 Internationale Dezimalklassifikation
```

Der dazu passende RDF-Eintrag aus dem Wikidata-Dump hat die folgende Form:

```
<http://www.wikidata.org/entity/Q1110558> <http://www.wikidata.org/prop/direct/
   ↪ P1036> "025.435"
```

Aufgrund der Strukturierung der Wikidata-Einträge wird außerdem zwischen sogenannten `Truthy` und `Full` Statements unterschieden[8]. Letztere sind wiederum aufgeteilt in die Gruppen `Preferred`, `Normal` und `Deprecated`. Für den *text2ddc* Export wurden alle `Truthy` Statements (und damit indirekt auch diese mit `Preferred`), sowie die als `Normal` markierten Daten verwendet.

Exemplarisch sei hier der SPARQL-Query zur Abfrage aller Wikidata-Elemente mit zugeordneten DDC-Notationen über alle Statement Varianten gezeigt (siehe Query 4.2), die einfache Abfrage der `Truthy`-Daten erfolgt entsprechend dem Query der GND-Abfrage (siehe Query 4.1):

```
SELECT ?qid ?ddc ?rank
WHERE
{
  ?qid <http://www.wikidata.org/prop/p/P1036> ?stmt .
  ?stmt <http://www.wikidata.org/prop/statement/P1036> ?ddc .
```

[4] https://www.mediawiki.org/wiki/Wikidata_Query_Service/User_Manual#Wikimedia_service
[5] https://dumps.wikimedia.org/wikidatawiki/
[6] https://www.wikidata.org/wiki/Property:P1036
[7] https://www.wikidata.org/wiki/Q1110558
[8] https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format#Statement_types

```
  ?stmt <http://wikiba.se/ontology#rank> ?rank .
}
```

Listing 4.2: SPARQL-Query zur Abfrage der Wikidata-Elemente mit DDC-Notation.

Exportiert werden konnten so insgesamt 8 826 Einträge, denen 18 388 DDC-Klassen (5 187 einzigartige) zugeordnet sind. Im Durchschnitt entspricht dies 2,1 Notationen pro Wikidata-Objekt.

GND

Die GND-Zuordnungen in Wikidata sind vergleichbar aufgebaut wie die in Abschnitt 4.1.2 beschriebenen DDC-Notationen. Die Verbindung erfolgt hier über das Property `P227 (GND ID)`[9]. So ist der Eintrag, der auf das in 4.1.1 vorgestellte Beispiel *Algebraische Topologie* in der GND verweist, wie folgt abgebildet:

```
<http://www.wikidata.org/entity/Q212803> <http://www.wikidata.org/prop/direct/
    ↪ P227> "4120861-4"
```

Damit ist eine Verbindung zwischen Wikidata-Item `Q212803`[10] und der DDC-Notation 514.2 hergestellt.

Der Export lieferte 792 253 Wikidata-Entitäten, die einem von 1 589 745 GND-Einträgen (davon 795 575 eindeutige) zugeordnet sind, mit im Durchschnitt 2,0 GND-Annotationen pro Wikidata-Entität.

Wikipedia

Um die Daten, die in den beiden vorherigen Abschnitten aus der GND und Wikidata gesammelt wurden, für das Training des *text2ddc*-Klassifikators zu nutzen, werden Zuordnungen zu Wikipedia-Artikeln benötigt. Diese können direkt aus Wikidata extrahiert werden, jedes Objekt ist dort dem passenden Artikel zugeordnet. Auf der Webseite ist dies über den Link *Wikidata-Datenobjekt* abgebildet. Über das Prädikat `schema:about`[11] lassen sich diese Daten per SPARQL abfragen, hier im Query beispielshaft gezeigt für alle Artikel der deutschen Wikipedia:

```
SELECT ?article ?name ?qid
WHERE
{
  ?article schema:isPartOf <https://de.wikipedia.org/> .
  ?article schema:name ?name .
  ?article schema:about ?qid .
}
```

Listing 4.3: SPARQL-Query zur Abfrage der deutschen Wikipedia-Artikel aus Wikidata.

[9] `https://www.wikidata.org/wiki/Property:P227`
[10] `https://www.wikidata.org/wiki/Q212803`
[11] `http://schema.org/about`

Insgesamt konnten so 46 801 656 Artikel in 307 Sprachen exportiert werden, mit durchschnittlich 209 867 Artikeln pro Sprache.

Für die Vergrößerung des Korpus (genauer beschrieben in Kapitel 4.2) wurden aus Wikidata außerdem die Namen der Hauptkategorien der verschiedenen Sprachen abgerufen. Dies ist möglich über die Verlinkungen der Hauptkategorie der englischen Wikipedia (`Q1281` Category:Contents[12]) zu den weiteren Sprachvarianten, siehe auch SPARQL-Query 4.4.

```
SELECT ?category ?lang
WHERE
{
  ?category schema:about wd:Q1281 .
  filter(regex(str(?category), ".wikipedia.org")) .
  ?category schema:inLanguage ?lang .
}
```

Listing 4.4: SPARQL-Query zur Abfrage der Hauptkategorien der verschiedenen Wikipedia-Sprachversionen.

Dazu wurden alle Wikipedia-Kategorien extrahiert, die über das Property `P31 (instance of)`[13] als Instanz der Klasse `Q4167836 (Wikimedia category)`[14] identifiziert werden können (siehe Query 4.5). Diese werden verwendet, um gleiche Kategorien über die verschiedene Sprachversionen der Wikipedia hinweg zuordnen zu können.

```
SELECT ?wiki ?name ?cat
WHERE
{
  ?wiki schema:isPartOf <https://de.wikipedia.org/> .
  ?wiki schema:about ?cat .
  ?cat wdt:P31 wd:Q4167836 .
  ?wiki schema:name ?name .
}
```

Listing 4.5: SPARQL-Query zur Abfrage der deutschen Wikipedia-Kategorien aus Wikidata.

4.1.3. Wikipedia

Das Training des Klassifikators basiert auf den Texten der Wikipedia-Artikel. In dieser Bachelorarbeit wird dabei auf das gleiche Wikipedia-Textkorpus zurückgegriffen, mit dem *text2ddc* zuvor in Baumartz, Uslu und Mehler (2018) und Uslu, Mehler und Baumartz (2019) trainiert wurde. Dieser bietet für die deutsche Wikipedia, wie in Kapitel 3.2 beschrieben, unter anderem Features wie Lemmatisierung und Entfernung von Funktionswörtern, alle weiteren Sprachen liegen nur tokenisiert vor.

[12] `https://www.wikidata.org/wiki/Q1281`

[13] `https://www.wikidata.org/wiki/Property:P31`

[14] `https://www.wikidata.org/wiki/Q4167836`

4.1.4. True Corpus

Mittels der Daten aus den vorherigen Abschnitten wird nun in zwei Schritten das finale *True Corpus* erstellt.

Zum Einsatz kommen verschiedene Python-Programme (siehe auch Anhang B zur Softwaredokumentation). Zuerst wurden die Wikidata- und GND-Daten zusammengeführt, dies führt zu einer Zuordnung von Wikidata-Objekten (spezifiziert über ihre `Q-ID`) auf DDC-Klassen. In diesem Schritt wurden die DDC-Notationen auf gültige Einträge gefiltert, sowie direkt die passenden Werte für die erste, zweite und dritte Ebenennotation erstellt. Aufgrund von fehlerhaften oder unvollständigen Daten in Wikidata wurden nur solche DDC-Klassen behalten, die aus Zahlen, einem Punkt oder Leerzeichen bestehen. Außerdem wurden nur die Notationen der Haupttafeln behalten, die der Hilfstafeln wurden entfernt. Von insgesamt 119 406 gesammelten DDC-Notationen wurden so 23 615 gefiltert, damit bleiben 95 791. Die Anzahl eindeutiger DDC-Klassen insgesamt beträgt 17 403, die der dritten Ebene 736, der zweiten 99 und der ersten 10. Diese sind 29 962 Wikidata-Objekten zugeordnet.

Im zweiten Schritt werden diese Wikidata-DDC-Zuordnungen verwendet, um sie auf die Wikipedia-Artikel der verschiedenen Sprachen abzubilden. Dies wird über die Wikipedia Q-IDs ermöglicht. Aus dem Wikipedia-Kategoriebaum (siehe Kapitel 4.2.1 für Details zu Erstellung desselben) werden über die Artikelnamen aus Wikidata die passenden IDs der Wikipedia-Artikel gewonnen. Dabei konnten im Falle des deutschen Korpus von insgesamt 2 788 731 Wikipedia-Artikeln 23 286 passende DDC-Notationen zugeordnet werden. Für die Daten der anderen Sprachen, sowie die Anzahl der DDC-Klassen siehe Tabelle 4.1. Die DDC-Mappings liegen nun in einem JSON-Datenformat vor, hier sei nochmal das Beispiel von weiter oben aufgegriffen:

```
{
  "wp_id": "165042",
  "wp_url": "https://de.wikipedia.org/wiki/Algebraische_Topologie",
  "wp_name": "Algebraische Topologie",
  "wd_qid": "Q212803",
  "ddcs": {
    "full": [ "514.2" ],
    "ddc1": [ "500" ],
    "ddc2": [ "510" ],
    "ddc3": [ "514" ]
  }
},
```

4.2. Vergrößern des Trainingskorpus

Das Ziel dieser Bachelorarbeit ist es, das Trainingskorpus das *text2ddc* zur Verfügung steht zu erweitern, um besonders beim Lernen der Modelle für die dritte DDC-Ebene von deutlich mehr Beispielen zu profitieren. Dazu werden, unter Zuhilfenahme der Wikipedia-Kategorien, bisher nicht-annotierten Artikeln neue DDC-Notationen zugeordnet. In diesem Abschnitt

Tabelle 4.1.: Anzahl der Artikel sowie DDC-Notationen der verschiedenen Ebenen der DDC im *True Corpus*.

Sprache	Artikel	DDC (eindeutige)	DDC 1	DDC 2	DDC 3
de	23 286	38 856 (15 484)	29 233 (10)	32 970 (99)	35 821 (697)
en	19 556	32 949 (14 419)	24 779 (10)	27 960 (97)	30 378 (690)
an	2 023	4 061 (3 160)	2 887 (10)	3 439 (88)	3 820 (451)
es	16 301	27 934 (13 029)	20 878 (10)	23 647 (97)	25 767 (666)
fa	10 563	18 917 (10 218)	13 863 (10)	15 867 (97)	17 426 (625)
fr	17 423	29 738 (13 650)	22 223 (10)	25 172 (97)	27 382 (676)
it	15 304	26 246 (12 568)	19 612 (10)	22 165 (97)	24 164 (666)
li	1 831	3 723 (2 902)	2 594 (10)	3 124 (85)	3 482 (429)
nl	15 640	27 001 (12 822)	20 063 (10)	22 837 (97)	24 875 (661)
pl	15 481	26 683 (12 618)	19 934 (10)	22 579 (98)	24 590 (664)
ru	16 296	27 966 (13 070)	20 885 (10)	23 700 (97)	25 790 (664)
sa	697	1 578 (1 324)	1 064 (10)	1 302 (72)	1 481 (262)
sh	7 738	14 282 (8 662)	10 316 (10)	12 018 (97)	13 229 (601)
ur	3 999	7 742 (5 529)	5 451 (10)	6 510 (94)	7 214 (531)

werden die verschiedenen Methoden und Varianten, die in dieser Arbeit entwickelt und getestet wurden, vorgestellt.

Alle Methoden der Korpus-Erweiterung basieren auf der Annahme, dass Artikel mit gleichen oder mindestens ähnlichen Kategorien auch thematisch verwandt sind, und somit als Trainingsmaterial für *text2ddc* zum Einsatz kommen können. Für die Erstellung dieser neuen Annotationen ergaben sich während der Arbeit die folgenden Varianten:

Kategoriemapping Die verschiedenen Möglichkeiten, wie basierend auf den aktuellen DDC-zu-Artikel-Zuordnungen des *True Corpus* neue Mappings für möglichst viele Klassen der DDC und Wikipedia-Kategorien gewonnen werden können:

- Es werden keine weiteren Mappings generiert.
- Händische Zuordnung der DDC-Notationen zu ausgewählten Wikipedia-Kategorien, dies wurde für die 100 Klassen der zweiten Ebene durchgeführt.
- Im Besonderen für das Mapping der 1 000 Klassen der dritten Ebene, sowie die weiteren Spracheversionen der Wikipedia, wurde ein automatisches Mapping durchgeführt.

Unterkategorien Die Wikipedia Kategorien sind in einem Baum organisiert, zur Erweiterung des Trainingskorpus werden drei Varianten unterschieden, die den Umfang der Dazunahme von Unterkategorien aufzeigen:

- Keine Unterkategorien
- Die direkten Unterkategorien einer Kategorie

- Sämtliche unterhalb einer Kategorie liegenden Unterkategorien

DDC-Annotation Unabhängig des Kategoriemappings sind folgende Methoden zur Annotation von weiteren Wikipedia-Artikeln mit DDC-Klassen entstanden:

- Die einfache Variante, in der direkt basierend auf den Kategoriemappings weitere Artikel ausgewählt wurden.
- Die heuristische Variante, bei der mittels Methoden wie Clustering das Thema eines Artikel stärker fokussiert betrachtet wird und basierend darauf Artikel annotiert werden.
- Eine Methode, die Artikel mittels eines Klassifikators annotiert, der mit den Kategoriezugehörigkeiten der Artikel trainiert wurde.

4.2.1. Wikipedia-Kategoriebaum

Im ersten Schritt wurde der Kategoriebaum der Wikipedia erstellt, basierend auf einem vom *TTLab* für eine bisher nicht veröffentlichte Forschung entwickelten Algorithmus. Mittels eines Wikipedia-Dump-Readers aus WikiDragon (Gleim, Mehler und Song 2018), der von Java nach C++ konvertiert wurde, wurden die Wikipedia-Dumps der verschiedenen Sprachen eingelesen, um den Kategoriebaum zu generieren. Aus den `page`- sowie `categorylinks`-Dumps wurden die Kategorien im Namensraum 14[15] extrahiert und als ein gerichteter Graph weiterverarbeitet. Aufgrund der Wikipedia Kategoriestruktur sind Zyklen in dem Graphen möglich, diese wurden mittels einer Breitensuche, ausgehend von der Hauptkategorie, aufgelöst. In der deutschen Wikipedia ist dies die Kategorie `!Hauptkategorie`, in der englischen `Contents`, diese Informationen wurden für alle Sprachen aus der Wikidata gesammelt (siehe dazu auch Abschnitt 4.1.2). Das hier verwendete Kategoriesystem ist damit ein *gerichteter azyklischer Graph* (Directed Acyclic Graph, DAG) beziehungsweise ein *Kategoriebaum*. Für jede Kategorie wurde anschließend mithilfe des Dijkstra-Algorithmus ihre minimale Höhe berechnet. Im letzten, neu hinzugefügten Schritt wurden die Listen der Unterkategorien (mittels einer Methode ähnlich einer Tiefensuche) und Oberkategorien (ähnlich einer Breitensuche) aller Kategorien gesammelt.

DDC und Wikipedia-Kategorie Zuordnungen

Für die Erweiterung des Korpus werden weitere Wikipedia-Artikel gesucht, denen eine DDC-Notation zugeordnet werden kann. Diese neuen Zuordnungen sollen über die den Artikeln zugewiesene Wikipedia-Kategorien durchgeführt werden. Voraussetzung dafür ist ein möglichst gutes Mapping von DDC-Klassen auf Wikipedia-Kategorien. Während für die zweite DDC-Ebene eine solche Zuordnung gut von Hand erstellt werden kann, ist dies bereits für die dritte Ebene aufgrund der stärkeren Spezialisierung der Klassen schwieriger. Ein wichtiges Ziel von *text2ddc* ist darüber hinaus die Integration möglichst vieler Sprachen, eine automatisierte Lösung ist hier somit die bevorzugte Option.

[15] `https://de.wikipedia.org/wiki/Hilfe:Namensräume`

Händische Zuordnung Für jede Klasse der zweiten Ebene der DDC werden passende Kategorien in der deutschen Sprachvariante der Wikipedia gesucht, die das Thema der Notation möglichst gut abdecken. Dies ist für viele Kategorien leicht umsetzbar, DDC 510 lässt sich so zum Beispiel direkt der Kategorie *Mathematik*[16] zuordnen. Andere Klassen lassen sich durch mehrere Kategorien abbilden, wie 420 über *Englische Sprache* und *Altenglisch*. Als dritte Variante können Klassen wie 490 über alle Unterkategorien der Wikipedia-Kategorie *Einzelsprache als Thema* abgebildet werden, wobei sämtliche bereits verwendeten Elemente dieser Kategorien, wie *Englische Sprache* aus dem vorherigen Beispiel, für diese Klasse entfernt werden. Die im Rahmen dieser Bachelorarbeit erstellten Mappings sind der Tabelle im Anhang A zu entnehmen.

Automatische Zuordnung Aufgrund der Menge von DDC-Klassen auf der dritten Ebene, als auch der zukünftigen Möglichkeit das Mapping auf andere Sprachen auszuweiten, wird ein Algorithmus entwickelt, um automatisiert die Zuordnungen von DDC-Notation auf Wikipedia-Klasse zu generieren. Dieser ist im Folgenden beschrieben, sowie in Algorithmus 1 vereinfacht dargestellt. Viele Einstellungesmöglichkeiten sind parameterisiert und bieten somit die Möglichkeit, die Ergebnisse zu variieren und eine Parameterstudie durchzuführen.

1. Basierend auf dem *True Corpus* wird für jede DDC-Notation berechnet, wie viele Artikel jeder in dieser Notation vorkommenden Wikipedia-Kategorie zugeordnet sind.
2. Diese Liste wird gefiltert, sodass nur Kategorien mit einer Mindestanzahl von Artikeln verwendet werden. Optional können Klassen, die nach dem Filtern leer wären, trotzdem behalten werden.
3. Die Eltern-Kategorien einer jeden Kategorie werden der Liste hinzugefügt, die Anzahl an Ebenen lässt sich anpassen. Das Ziel dieses Schrittes ist es, möglichst allgemeine Kategorien aus dem *True Corpus* zu erhalten. Aufgrund der Wikipedia Kategoriestruktur wurden allerdings die obersten Ebenen, die in der deutschen Sprachversion aus zu allgemeinen Kategorien wie *Zeitliche Systematik*[17] bestehen, ausgespart. Dabei wird die Anzahl der Artikel dieser neu hinzugefügten Kategorien berechnet als die summierte Häufigkeiten der Kinder-Kategorien.
4. Da für die DDC-Klassennamen Übersetzungen in verschiedene Sprachen existieren, werden diese Namen genutzt, um ähnliche und gleich benannte Kategorien in der Wikipedia bevorzugt einander zuzuordnen. Dabei werden verschiedene Varianten unterstützt, bei denen der Vergleich über die zweite und dritte DDC-Ebene durchgeführt werden kann (so können zum Beispiel alle Klassennamen der dritten Ebene auch für Vergleiche auf der zweiten Ebene verwendet werden). Der eigentliche Namensvergleich findet auf der Menge der Worte im Namen statt, je nach Grad der Übereinstimmung wird, zum Ausdruck der Verstärkung, die Anzahl Artikel in dieser Kategorie erhöht. Um die Erkennungsrate zu erhöhen werden außerdem Satzzeichen sowie Stoppwörter mittels NLTK (Natural Language

[16] `https://de.wikipedia.org/wiki/Kategorie:Mathematik`
[17] `https://de.wikipedia.org/wiki/Kategorie:Zeitliche_Systematik`

Input :

- *wiki_category_tree*: Der Wikipedia-Kategoriebaum;
- *tc_ddc_wiki_categories*: Mapping von DDC-Notation auf alle zugehörigen Wikipedia-Kategorien, mitsamt der Anzahl zu diesen Kategorien zugehörigen Artikeln, aus dem *True Corpus*;
- *ddc_names*: DDC-Klassennamen;
- *min_articles*: Minimale Anzahl an Artikeln die in einer Kategorie vorhanden sein sollten;
- *num_parent_layers*: Anzahl an Ebenen oberhalb einer Kategorie die Hinzugefügt werden;
- *limit_method*: Methode um die Anzahl Kategorien zu limitieren, beispielsweise `count`, `mean` oder `height`.

Output : *new_mapping*: Erweitertes Mapping von DDC-Notation auf Wikipedia-Kategorien.

```
1  categories_filtered = filter(tc_ddc_wiki_categories, min_articles);
2  for l = 1; l ≤ num_parent_layers; l++ do
3  |   categories_filtered = add_parent_layer(categories_filtered, wiki_category_tree);
4  end
5  foreach ddc ∈ categories_filtered do
6  |   categories_filtered_ddc = bootstrap_with_names(categories_filtered_ddc, ddc_names);
7  end
8  new_mapping = ∅;
9  foreach ddc1 ∈ categories_filtered do
10 |   foreach category ∈ categories_filtered_ddc1 do
11 |   |   foreach ddc2 ∈ categories_filtered do
12 |   |   |   if ddc1 ≠ ddc2 ∧category ∈ categories_filtered_ddc2 then
13 |   |   |   |   keep_ddc1 = choose(categories_filtered, ddc1, ddc2, category);
14 |   |   |   |   if keep_ddc1 then
15 |   |   |   |   |   new_mapping_ddc1,category = categories_filtered_ddc1,category;
16 |   |   |   |   else
17 |   |   |   |   |   new_mapping_ddc2,category = categories_filtered_ddc2,category;
18 |   |   |   |   end
19 |   |   |   end
20 |   |   end
21 |   end
22 end
23 new_mapping = limit_categories(new_mapping, limit_method);
24 return new_mapping;
```

Algorithmus 1 : Automatisches Mapping der DDC-Notationen zu Wikipedia-Kategorien, vereinfacht dargestellt.

Toolkit) (Bird, Loper und Klein 2009) aus den Kategorienamen entfernt. Sind keine Übersetzungen für eine Sprache verfügbar, werden die Deutschen DDC-zu-Wikipedia-Kategoriemappings, die über die internen Wikidata-Verlinkungen soweit möglich in die passende Sprache übersetzt werden, zu einer Art von Bootstrapping verwendet.

5. Im nächsten Schritt werden optional die Anzahl der Artikel genutzt, um ein eindeutiges $1 : n$ Mapping von Wikipedia-Kategorien zu DDC-Klassen herzustellen. Dabei wird versucht zu vermeiden, dass eine Notation keine Kategorien zugeordnet bekommt.
6. Da die Anzahl an Wikipedia-Kategorien oft in diesem Schritt noch sehr groß ist, wird als letzte Aktion die endgültige Anzahl an Zuordnungen pro Klasse begrenzt. Hier sind wieder verschiedene Varianten möglich, unter anderem die Wahl einer festen Anzahl oder eine Auswahl absteigend nach Artikelanzahl, oder nach der Tiefe der Kategorie im Wikipedia-Kategoriebaum. Auch ein Filtern auf Kategorien, die mehr Artikel als der Durchschnitt über alle Kategorien dieser, oder aller, DDC-Notationen haben ist möglich.

Hinzunahme der Unterkategorien

Die DDC-auf-Wikipedia-Kategoriemappings können nun verwendet werden um weitere Artikel auszuwählen. Aufgrund der Struktur der Wikipedia sind diese Kategorien in einer Baumstruktur angeordnet, was verschiedene Möglichkeit bietet, die Unterkategorien dieses Baums miteinzubeziehen. So sind zum Beispiel alle Artikel, die einer Subkategorie von *Mathematik* zugeordnet sind, thematisch in diesem Bereich einzuordnen und könnten somit als Trainingsmaterial für DDC-Notation 510 dienen. Aufgrund der Menge an möglichen neuen Kategorien werden nicht nur alle rekursiv unter eine Kategorie liegenden Unterkategorien verwendet, sondern drei Varianten unterstützt: Die Hinzunahme keiner Unterkategorien ist die einfachste, gefolgt von der Aufnahme aller direkten Unterkategorien. So würde zum Beispiel *Ungleichung* als direkte Unterkategorie von *Mathematik* verwendet werden, nicht aber *Analytische Funktion*, die erst über die Unterkategorie *Mathematische Funktion* erreichbar wäre. In der dritten Variante hingegen werden alle Unterkategorien, unabhängig von der Entfernung, mit einbezogen.

4.2.2. Artikelauswahl

Nachdem im vorherigen Abschnitt das Erstellen der DDC-zu-Wikipedia-Kategoriemapping gezeigt wurde, widmet sich dieses Kapitel der Vergrößerung des Trainingskorpus, der Auswahl, beziehungsweise der Annotation weiterer Wikipedia-Artikel. Im Folgenden werden die drei verschiedenen Methoden vorgestellt.

Einfacher Algorithmus

Der einfache Algorithmus fügt alle Artikel, der aus den Kategoriemappings gewonnenen Wikipedia-Kategorien, dem Trainingskorpus hinzu, ohne weitere Änderungen. Einzig die

Hinzunahme der Unterkategorien kann variiert werden. So können ausschließlich die Artikel der direkten Kategorie, die auf eine DDC-Notation gemappt wurde, verwendet werden. Alternativ werden noch diejenigen Artikel hinzugefügt, die in einer direkten Unterkategorie der direkt zugeordneten Kategorien zu finden sind. Die dritte Option ist die Hinzunahme sämtliche Artikel, die einer beliebigen Unterkategorie zugeordnet sind. Besonders die letzte Variante führt zwar zu einem großen Anstieg der Dokumente im Trainingskorpus, fügt aber auch viele Artikel mit großer Anzahl an Kategorien und damit auch DDC-Notationen hinzu, was wiederum das Lernen des Klassifikators erschwert.

Heuristische Variante

Der Ansatz dieser Variante (siehe Algorithmus 2) versucht, den großen Nachteil des einfachen Algorithmus aus Abschnitt 4.2.2 auszugleichen. Dazu werden einem Artikel nicht unkontrolliert alle DDC-Notationen aller Kategorien hinzugefügt. Der Kategoriebaum jeder Wikipedia-Kategorie wird ebenenweise nach oben durchsucht, um alle Kategorien zu finden, für die ein DDC-Mapping vorliegt. Die Anzahl aller Notationen, die durch die dem Artikel zugeordneten Kategorien gefunden wurden, werden gesammelt. Ein Artikel über ein zum Beispiel *mathematisches Thema* kann somit über zwei verschiedene Wikipedia-Kategorien zweimal die DDC-Notation 510 erhalten. Ausgewählt werden nun diese DDC-Klassen, die einen Mindest-Prozentsatz der Gesamtanzahl der zugewiesenen Klassen ausmachen. Damit wird ausgenutzt, dass die meisten Artikel mehrere Kategorien besitzen (im Durchschnitt 4,48 und maximal sogar 120), aber für das Training lediglich die wirklich großen vorherrschenden Themen eines Artikels gesucht sind. Optional findet ein Clustering der DDC-Notationen auf der ersten Ebene statt, bevor die Häufigkeiten ausgewertet werden.

Variante mit Klassifikator

Da die Auswahl der Artikel aufgrund ihrer Zugehörigkeit zu den DDC gemappten Wikipedia-Kategorien durchgeführt wird, bietet es sich an, sich dieser Aufgabe nicht nur über heuristische Methoden zu nähern, sondern auch hier die Vorteile des maschinellen Lernens zu nutzen (siehe Algorithmus 3). Zum Einsatz kommt hier ein *Random Forest-Klassifikator* (Random Forest Classificator, RFC) aus scikit-learn (Pedregosa u. a. 2011). Das Trainingskorpus besteht aus Feature-Vektoren der Kategorien: Aus allen Wikipedia-Kategorie-Zuordnungen zu DDC aus dem *True Corpus* wird pro Artikel ein One-Hot-Encoding Feature-Vektor gebildet. Dieser geordnete Vektor hat als Länge die Anzahl aller möglichen Kategorien, jedes Element steht für eine Kategorie. Dabei markiert eine 1 das Vorhandensein einer Kategorie in einem Artikel, die restlichen Elemente des Vektors sind auf 0 gesetzt. Aufgrund der Notwendigkeit der *Multi-Label Multi-Class* Klassifikation wird scikit-learns `MultiLabelBinarizer`[18] zur Erstellung dieser Feautre-Vektoren verwendet. Optional können die erweiterten DDC-Mappings (siehe Abschnitt 4.2.1) mit in das Trainingskorpus aufgenommen werden. Dies erweitert die Anzahl an Trainingsbeispielen deutlich, aufgrund von Speicherbeschränkungen konnte allerdings nur mit insgesamt maximal 200 000 Beispielen trainiert werden. Dabei

[18] `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html`

Input :

- *wiki_category_tree*: Der Wikipedia-Kategoriebaum;
- *wiki_articles_categories*: Liste aller Kategorien der Wikipedia-Artikel;
- *threshold*: Der Schwellenwert zur Auswahl der DDC-Klassen.

Output : *new_ddc_articles*: Vergrößertes Trainingskorpus mit DDC-Notation zu Wikipedia-Artikel.

```
1  category_top_ddc = ∅;
2  foreach category_id ∈ wiki_category_tree do
3  |   category_top_ddc_category_id = get_top_ddc(category_id, wiki_category_tree);
4  end
5  new_ddc_articles = ∅;
6  foreach article_id ∈ wiki_articles_categories do
7  |   ddcs = ∅;
8  |   foreach category_id ∈ wiki_articles_categories_article_id do
9  |   |   ddcs = ddcs ∪ category_top_ddc_category_id;
10 |   end
11 |   foreach ddc ∈ ddcs do
12 |   |   if ddcs_ddc < threshold then
13 |   |   |   delete ddcs_ddc;
14 |   |   end
15 |   end
16 |   new_ddc_articles_article_id = ddcs;
17 end
18 return new_ddc_articles;
```

Algorithmus 2 : Heuristischer Algorithmus, der neue Wikidata-Artikel mit DDC-Klassen annotiert, vereinfacht dargestellt.

Input :

- *true_corpus*: Das *True Corpus*, eine Liste von Wikipedia-Artikeln mitsamt ihren zugewiesenen Kategorien sowie DDC-Notationen;
- *test_size*: Die Größe des Testkorpus des Klassifikators;
- *wiki_articles_categories*: Liste aller Kategorien der Wikipedia-Artikel.

Output : *new_ddc_articles*: Vergrößerter Trainingskorpus mit DDC-annotierten Wikipedia-Artikeln.

```
1  true_categories, true_ddcs = reformat(true_corpus);
2  category_encoder = new MultiLabelBinarizer();
3  true_x = category_encoder.fit_transform(true_categories);
4  ddc_encoder = new MultiLabelBinarizer();
5  true_y = ddc_encoder.fit_transform(true_ddcs);
6  true_x_train, true_x_test, true_y_train, true_y_test = train_test_split(true_x, true_y,
     test_size);
7  classifier = new RandomForestClassifier(n_estimators, max_features);
8  classifier.fit(true_x_train, true_y_train);
9  new_ddc_articles = ∅;
10 foreach article_id ∈ wiki_articles_categories do
11     pred_x = category_encoder.transform(wiki_articles_categories_article_id);
12     if pred_x ≠ 0 then
13         pred_y = classifier.predict(pred_x);
14         ddcs = ddc_encoder.inverse_transform(pred_y);
15         new_ddc_articles_article_id = ddcs;
16     end
17 end
18 return new_ddc_articles;
```

Algorithmus 3 : Annotation neuer Wikidata-Artikel mit DDC-Klassen mittels eines Klassifikators, stark vereinfacht dargestellt.

werden die Beispiele des *True Corpus* immer gewählt und die restlichen freien Plätze mit zufälligen Artikeln aus den erweiterten Mappings aufgefüllt. Eine zufällige Aufteilung in 80 % Trainings- und 20 % Testmengen wurde durchgeführt und anschließend mittels scikit-learns `RandomForestClassifier`[19] trainiert. Die Ergebnisse des Klassifikators werden im Kapitel 4.4 gezeigt.

Der so trainierte Random Forest-Klassifikator wird nun verwendet, um weitere Artikel mit einer DDC-Notation zu versehen, die bisher nicht über Wikidata oder die GND im *True Corpus* annotiert wurden. Nicht alle der 372 887 Kategorien der Wikipedia sind im Trainingskorpus des Klassifikators enthalten, insgesamt konnten 18 985 Feature-Klassen generiert werden. Daraus ergibt sich, dass viele Artikel in der Klassifikationsphase, in der ihnen eine DDC-Notation zugewiesen werden soll, einen Feature-Vektor komplett aus Nullen bestehend besitzen. Diese konnten entsprechend nicht annotiert werden, sodass keine komplette Abdeckung der Wikipedia-Artikel erzielt werden konnte.

4.2.3. Vergrößertes Korpus

Mit denen in diesem Kapitel beschriebenen Methoden wurde somit das ursprüngliche *True Corpus* um weitere Trainingsbeispiele, bestehend aus Wikipedia-Artikeln mit zugeordneten DDC-Klassen, erweitert. Durch die Aktualisierung des *True Corpus* stieg die Anzahl Artikel von ursprünglich 12 655 bereits auf 23 286. Der Evaluation vorweg greifend, konnte die Anzahl mit den oben beschrieben Methoden nochmals deutlich erhöht werden. Dabei konnte nicht nur für den deutschen *text2ddc*-Klassifikator, sondern für alle in dieser Bachelorarbeit evaluierten Sprachversionen Vergrößerungen erzielt werden.

4.2.4. Weitere Sprachen

Entsprechend der Methode, die bei *text2ddc* bisher zum Einsatz kam, werden auch in dieser Bachelorarbeit die Trainingsdaten für weitere Sprachen direkt aus dem Deutschen Korpus abgeleitet. Die über Wikidata abgerufenen Verlinkungen der Sprachversionen der Wikipedias sind die Basis, um den Artikel-Texten die DDC-Notationen zuzuweisen. Dies ist möglich, da von der Annahme ausgegangen wird, dass die jeweiligen Artikel in den verschiedenen Sprachen die gleiche thematische Ausrichtung haben. Wie sich in den Ergebnissen zeigt, konnte der Abstand zwischen den Scores der deutschen Sprachversion und den weiteren Sprachen, der in Uslu, Mehler und Baumartz (2019) beschrieben wird, verringert werden.

4.3. Training der Klassifikatoren

Dieses Kapitel beschreibt das verwendete Wikipedia-Korpus, sowie die zum Training eingesetzte *fastText*-Bibliothek.

[19] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

4.3.1. Trainings- und Testkorpus

Basierend auf den Methoden, die in Kapitel 4.2 beschrieben sind, werden verschiedene Trainingskorpora erstellt. Da ein wichtiger Teil dieser Bachelorarbeit der Vergleich mit den kleineren, bisher genutzten Korpora für *text2ddc* ausmacht, wird die gleiche Aufteilung der Korpora in Trainings- und Testmengen gewählt wie in Baumartz, Uslu und Mehler (2018) und Uslu, Mehler und Baumartz (2019).

Da dieses Wikipedia-Korpus älter ist als die Daten, die für diese Bachelorarbeit aus Wikidata gewonnen wurden, konnten nicht für alle aus Wikidata gesammelten Daten Artikeltexte gefunden werden. So waren zum Beispiel in einem Experiment 2 196 944 Artikel mit DDC-Notation verfügbar, 444 263 davon fehlten allerdings im Wikipedia-Korpus. Die folgenden Zahlen innerhalb der Experimente geben somit immer den Stand bezogen auf den Wikipedia-Korpus aus Baumartz, Uslu und Mehler (2018) und Uslu, Mehler und Baumartz (2019) wieder.

Erste Experimente mit dem vergrößerten Trainingskorpus haben darüber hinaus gezeigt, dass die Ergebnisse der Klassifikation weiter verbessert werden konnten, wenn die POS-Features und Wikipedia-Kategorienamen aus dem Korpus entfernt werden. Damit weist das Korpus der deutschen Sprachversion nun folgende Features auf:

- Lemmatisierung
- Entfernung von Satzzeichen
- Entfernung von Funktionswörtern
- Word-Embeddings

Die Korpora der weiteren Sprachen kommen entsprechend in tokenisierter Form, ohne Wikipedia-Kategorienamen, zum Einsatz.

4.3.2. fastText

Zum Training der Klassifikatoren wird *fastText* (Joulin u. a. 2017) eingesetzt. Ein Bash-Skript (siehe auch die Softwaredokumentation in B) kommt zum Einsatz, um das Training zu starten, sämtliche Parameter und verwendeten Daten festzuhalten und um die Auswertung des Klassifikators durchzuführen. Ausgehend von den Hyperparametern, die in Uslu, Mehler und Baumartz (2019) ermittelt wurden, führte eine kleine Parameter Studie zu den folgenden, auf die größeren Trainingskorpora angepassten Parametern:

- `minCount`: 10
- `wordNgrams`: 2
- `bucket`: 20 000 000
- `minn`: 0
- `maxn`: 0

- `lr`: 0,2
- `lrUpdateRate`: 150
- `dim`: 300
- `ws`: 10
- `epoch`: 200 (DDC-Ebene 2) und 100 (DDC-Ebene 3)
- `neg`: 10
- `loss`: `softmax`

Aufgrund der deutlich gestiegenen Anzahl von Trainingsbeispielen konnten vor allem die Trainingsiterationen von ursprünglich 10 000 Epochen auf 200 (DDC-Ebene 2) und 100 (DDC-Ebene 3) verringert werden, was ein schnelleres Training ermöglicht und die Gefahr des Overfitting auf die Trainingsdaten reduziert.

4.4. Evaluation der Ergebnisse

Dieses Kapitel stellt die Ergebnisse der Bachelorarbeit vor. Das Hauptaugenmerk der Experimente lag auf der Evaluation der Auswirkungen der verschiedenen Methoden zur Vergrößerung des Trainingskorpus, die in Kapitel 4.2 vorgestellt werden, auf die Güte der *text2ddc*-Klassifikatoren. Im ersten Schritt wurden die unterschiedlichen Methoden auf den Daten der deutschen zweiten DDC-Ebene untersucht, gefolgt von der dritten Ebene. Im Anschluss wurden die Erkenntnisse dieser Experimente auf eine Auswahl weiterer Sprachen (darunter Englisch, Spanisch und Französisch) angewendet.

4.4.1. Zweite Ebene der DDC

Bei allen Experimenten der zweiten Ebene der DDC konnten alle 99 Klassen dieser Ebene abgedeckt werden, die fehlende letzte Klasse 450 der alten *text2ddc*-Version steht somit auch mit insgesamt 56 Beispielen zur Verfügung.

Die folgende Tabelle 4.2 zeigt die Ergebnisse des einfachen Algorithmus, dazu wurde das händisch zugeordnete Wikipedia-Kategorien-zu-DDC-Klassenmapping verwendet, um das Trainingskorpus zu vergrößern. Bereits in der einfachsten Variante, in der keinerlei Unterkategorien verwendet wurden, ist ein Anstieg der Trainingsbeispiele von 12 655 auf 73 634 zu erreichen, eine Steigerung auf 581 %. Die Hinzunahme der Unterkategorien führt zu einem weiteren starken Anstieg, im maximalen Fall können mit 1 752 505 Beispielen fast sämtliche 1 760 875 Artikel der Wikipedia zum Training verwendet werden (mehr als 99 %). Dies führt allerdings auch zu einem starken Abfall der *Precision@1*, die mit 0,496 sogar leicht unter 0,5 fällt. Auffällig ist hier die deutlich höhere durchschnittliche Anzahl von DDC-Klassen pro Beispiel mit 6,89, was das Training des Klassifikators erschwert. Die Werte des alten *text2ddc*-Klassifikators mit 87,4 % konnten hier nicht erreicht werden.

Tabelle 4.2.: Ergebnisse der Experimente des einfachen Algorithmus.

Variante	P@1	Train	Klassen/Beispiel	Beispiele/Klasse
Baseline *text2ddc*	**0,874**	12 655	1,42	183,76
Keine Unterkategorien	0,837	73 634	1,20	890,02
Direkte Unterkategorien	0,796	174 360	1,848	3 255,28
Alle Unterkategorien	0,496	1 752 505	6,89	122 006,95

P@1: Precision an Position 1; *Train*: Anzahl Trainingsbeispiele; *Klassen/Beispiel*: Durchschnittliche Anzahl Klassen pro Trainingsbeispiel; *Beispiele/Klasse*: Durchschnittliche Anzahl Trainingsbeispiele pro Klasse.

Die Ergebnisse des heuristischen Algorithmus (wie in Kapitel 4.2.2 beschrieben) zeigt Tabelle 4.3. Auch hier kam das manuell erstellte DDC-zu-Wikipedia-Kategoriemapping zum Einsatz. Die besten Klassifikatoren erreichen hier mit einem Score von 0,831, trotz deutlich

Tabelle 4.3.: Ergebnisse der Experimente des heuristischen Algorithmus.

Variante	Threshold	P@1	Train	Klassen/Beispiel	Beispiele/Klasse
Baseline *text2ddc*		**0,874**	12 655	1,42	183,76
Baseline Experiment 1		0,837	73 634	1,20	890,02
DDC not clustered	50 %	0,745	595 738	1,03	6 183,95
DDC clustered	50 %	0,675	991 459	2,01	20 166,56
DDC not clustered	90 %	0,831	132 600	1,06	1 422,25
DDC clustered	90 %	0,803	206 709	1,46	3 041,60
DDC not clustered	95 %	0,831	122 155	1,07	1 316,75

Clustered: DDC-Notationen auf Ebene 1 gekürzt; *Threshold*: Mindest-Prozentsatz den eine DDC-Klasse pro Artikel erreichen muss um gewählt zu werden; *P@1*: Precision an Position 1; *Train*: Anzahl Trainingsbeispiele; *Klassen/Beispiel*: Durchschnittliche Anzahl Klassen pro Trainingsbeispiel; *Beispiele/Klasse*: Durchschnittliche Anzahl Trainingsbeispiele pro Klasse.

mehr Trainingsbeispielen, allerdings nur ähnliche Werte wie die einfachste Variante aus dem ersten Experiment.

Das letzte Experiment auf der zweiten DDC-Ebene betrachtet den Einsatz eines Klassifikators zur Auswahl, beziehungsweise Annotation der neuen Trainingsartikel und ist in den Tabellen 4.4 und 4.5 abgebildet.

Die erste Tabelle 4.4 zeigt die Ergebnisse der Experimente ohne, sowie mit der Hinzunahme der automatisch erstellten Kategoriemappings. In allen Fällen konnte das Trainingskorpus von den ursprünglichen 12 655 Beispielen deutlich vergrößert werden. Auffällig ist der Unterschied, den der `Max Features` Parameter beim Training des Random Forest-Klassifikators verursacht. Obwohl der Klassifikator in beiden Fällen mit einem F_1-score von 0,57 ähnlich schlecht abschneidet, unterscheidet sich das Ergebnis des DDC-Klassifikators um 10 %. Die Hinzunahme des automatischen Kategoriemappings erhöht den F_1-score des Auswahl-Klassifikators, sowie die Anzahl Trainingsbeispiele zwar deutlich, erzielt aber ähnlich schlechte

Tabelle 4.4.: Ergebnisse der Experimente des Algorithmus mit Klassifikator ohne, sowie mit automatisch erstelltem Kategoriemapping.

	Variante			
	Baseline	RFC1	RFC2	RFC3
Auswahl-Klassifikator				
Features		18 810	18 810	30 658
Trainingsbeispiele		18 572	18 572	80 000
Kategoriemapping		Keines	Keines	Automatisch
Unterkategorien		Keine	Keine	Keine
Estimator Trees		100	100	100
Max Features		`sqrt`	`all`	`sqrt`
Max Samples		100 000	100 000	100 000
F_1-score		0,57	0,57	0,82
Prediction Threshold		50 %	50 %	50 %
DDC-Klassifikator				
Precision@1	**0,874**	0,872	0,76	0,727
Trainingsbeispiele	12 655	224 828	1 011 858	1 202 380
Klassen/Beispiel	1,42	1,24	1,06	1,20
Beispiele/Klasse	183,76	2 824,27	10 787,19	14 518,59

Zum Einsatz kommt ein Random Forest-Klassifikator, die Aufteilung in Trainings- und Testmenge ist immer 80 % / 20 %.
Auswahl-Klassifikator *Features*: Anzahl Features, die über die Kategoriezugehörigkeiten gewonnen wurden; *Estimator Trees*: Anzahl der Bäume; *Max Features*: Maximale Anzahl an Features, die bei jedem Split betrachtet werden; *Max Samples*: Maximale Anzahl an Trainingsbeispielen, aufgrund Speichereinschränkungen wird die verfügbare Anzahl an Trainingsbeispielen begrenzt; *Prediction Threshold*: Minimal nötiger Score der bei der Bestimmung der neuen DDC-Notationen erreicht werden muss.
DDC-Klassifikator *Klassen/Beispiel*: Durchschnittliche Anzahl Klassen pro Trainingsbeispiel; *Beispiele/-Klasse*: Durchschnittliche Anzahl Trainingsbeispiele pro Klasse.

Werte bei der DDC-Klassifikation. Dies deutet auf eine niedrige Qualität des automatischen Kategoriemappings hin.

Bestätigt wird dies durch die Experimente mit den händisch erstellten Kategoriemappings, die in Tabelle 4.5 aufgelistet sind. Bereits der erste Versuch liefert mit 0,875 das beste Ergebnis des DDC-Klassifikators, wenn auch die Verbesserung zum bisherigen Ergebnis 0,874 von *text2ddc* nur marginal ausfallen und durch die zufällige Initialisierung innerhalb des Trainings mit *fastText* erklärt werden kann. Das Trainingskorpus konnte um 334 493 Beispiele vergrößert werden. Während *text2ddc* im Durchschnitt nur 890,02 Beispiele pro Klasse zum Training verfügbar hatte, konnte dies auf nun 4 106,64 erhöht werden, bei durchschnittlich 1,17 DDC-Klassen pro Beispieltext.

In drei weiteren Experimenten wird versucht, die Qualität der neu hinzugenommen Artikel zu verbessern. Dazu wird der `Prediction Threshold` des Auswahl-Klassifikators angepasst: Standardmäßig werden mit diesem Klassifikator für alle Artikel der Wikipedia, die

nicht bereits im *True Corpus* enthalten sind, DDC-Klassen bestimmt, falls die Klasse mit einer Wahrscheinlichkeit von mehr als 50 % gewählt wird. Höhere Werte führen dementsprechend zu weniger DDC-Notationen und auch zu weniger Beispielen im Trainingskorpus, bei einem Schwellenwert von 90 % stehen nur 133 878 Artikel zur Verfügung. Interessanterweise sinkt der Score des DDC-Klassifikators allerdings, je weiter der Schwellenwert sich von 50 % entfernt. Im Detail beträgt er 0,86 bei 75 %, sowie 0,852 bei 90 % und sinkt auf erwartbar niedrigere 0,798 bei 25 %.

Weitere Tests wurden durchgeführt, in denen die Unterkategorien des Wikipedia-Kategoriebaums mit einbezogen wurden. Dies erhöht die Anzahl an möglichen Trainingsdaten des Auswahl-Klassifikators deutlich. Aufgrund von Speicherrestriktionen wurde die maximale Anzahl auf 100 000, sowie in einem zweiten Experiment auf 200 000 begrenzt. Dabei wurden die Daten des *True Corpus* wie bisher verwendet und erweitert mit einer zufälligen Auswahl an weiteren Trainingsbeispielen, die durch die Hinzunahme der Unterkategorien gewonnen werden konnten. Vergleichbar der Ergebnisse der einfachen Algorithmen konnte eine größere Erweiterung des Trainingskorpus erreicht werden, allerdings bei gleichzeitigem deutlichen Anstieg auf durchschnittlich über 6 Klassen pro Beispiel und deutlichem Absinken der Scores des DDC-Klassifikators auf Werte im Bereich von nur 0,5.

Tabelle 4.5.: Ergebnisse der Experimente des Klassifikators mit dem händisch erstelltem Kategoriemapping.

		Variante					
	Baseline	RFC4	RFC5	RFC6	RFC7	RFC8	RFC9
Auswahl-Klassifikator							
Features		18 985	18 985	18 985	18 985	18 913	18 937
Trainingsbeispiele		66 645	66 645	66 645	66 645	80 000	160 000
Kategoriemapping		Manuell	Manuell	Manuell	Manuell	Manuell	Manuell
Unterkategorien		Keine	Keine	Keine	Keine	Alle	Alle
Estimator Trees		100	100	100	100	100	100
Max Features		`sqrt`	`sqrt`	`sqrt`	`sqrt`	`sqrt`	`sqrt`
Max Samples		100 000	100 000	100 000	100 000	100 000	200 000
F_1-score		0,88	0,88	0,88	0,88	0,97	0,98
Prediction Threshold		50 %	90 %	75 %	25 %	50 %	50 %
DDC-Klassifikator							
Precision@1	0,874	**0,875**	0,852	0,86	0,798	0,509	0,518
Trainingsbeispiele	12 655	347 148	133 878	173 807	1 112 544	1 752 745	1 752 811
Klassen/Beispiel	1,42	1,17	1,13	1,14	1,30	6,89	6,89
Beispiele/Klasse	183,76	4 106,64	1 525,47	2 002,05	14 562,01	122 010,84	122 011,49

Zum Einsatz kommt ein Random Forest-Klassifikator, die Aufteilung in Trainings- und Testmenge ist immer 80 % / 20 %.

Auswahl-Klassifikator *Features*: Anzahl Features, die über die Kategoriezugehörigkeiten gewonnen wurden; *Estimator Trees*: Anzahl der Bäume; *Max Features*: Maximale Anzahl an Features, die bei jedem Split betrachtet werden; *Max Samples*: Maximale Anzahl an Trainingsbeispielen, aufgrund Speichereinschränkungen wird die verfügbare Anzahl an Trainingsbeispielen begrenzt; *Prediction Threshold*: Minimal nötiger Score der bei der Bestimmung der neuen DDC-Notationen erreicht werden muss.

DDC-Klassifikator *Klassen/Beispiel*: Durchschnittliche Anzahl Klassen pro Trainingsbeispiel; *Beispiele/-Klasse*: Durchschnittliche Anzahl Trainingsbeispiele pro Klasse.

Damit fällt das Ergebnis der Experimente der zweiten DDC-Ebene gemischt aus. Es konnte eine deutliche Vergrößerung des Trainingskorpus (von 12 655 auf 347 148) bei Beibehaltung des Scores von 87 % des DDC-Klassifikators erreicht werden. Dazu wurde die Anzahl der Beispiele pro Klasse deutlich von 183,76 auf 4 106,64 erhöht. Dies kommt insbesondere den Klassen zugute, die bisher wenige Beispiele zur Verfügung hatten. Die minimale Anzahl an Beispielen konnte von 1 auf 19 gesteigert werden, insgesamt gibt es 21 Klassen mit weniger als 10 Trainingsbeispielen. Ein Vergleich der Verteilung der Anzahl Beispieltexte pro Klasse ist in Grafik 4.2 zu sehen, sowie eine grobe Übersicht der Anzahl neu gewonnener Beispieltexte in Grafik 4.3.

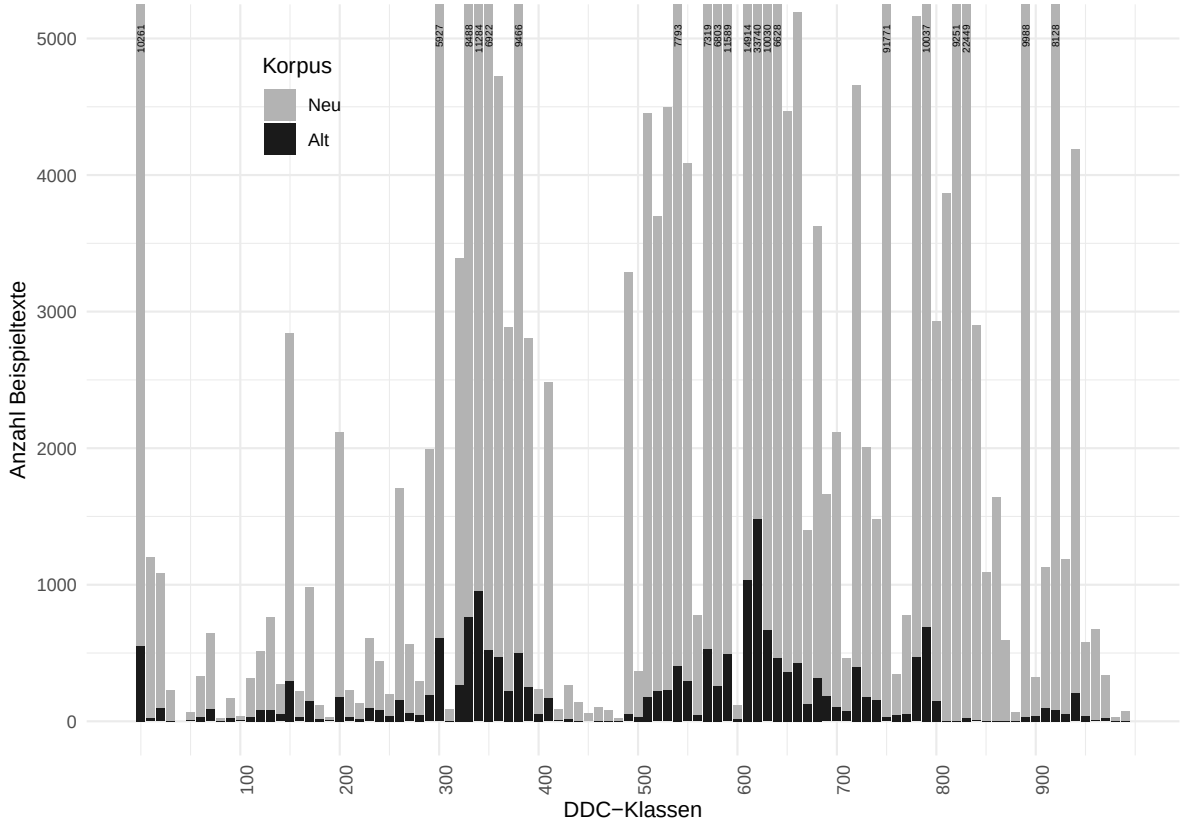

Abbildung 4.2.: Die Grafik vergleicht die Größe des alten und die des erweiterten Trainingskorpus der zweiten Ebene der DDC.

Ein Blick auf die genauen Daten zeigt einen möglichen Grund auf, warum der Score nicht weiter verbessert werden konnte. Betrachten wir die DDC-Klassen, die durch den vergrößerten Korpus mehr als 1 000 Beispiele dazugewonnen haben, so sind in der Testmenge für 11 von diesen 53 Klassen weniger als 10 (und bei 7 Klassen keinerlei) Beispiele verfügbar. Insgesamt stehen auch für die neu erschlossene DDC-Klasse 450, sowie 19 weitere Klassen, die im alten Trainingskorpus weniger als 10 Vorkommnisse aufweisen, gegenüber nun deutlich mehr Trainingsdaten, weiterhin keine Testdaten zur Verfügung.

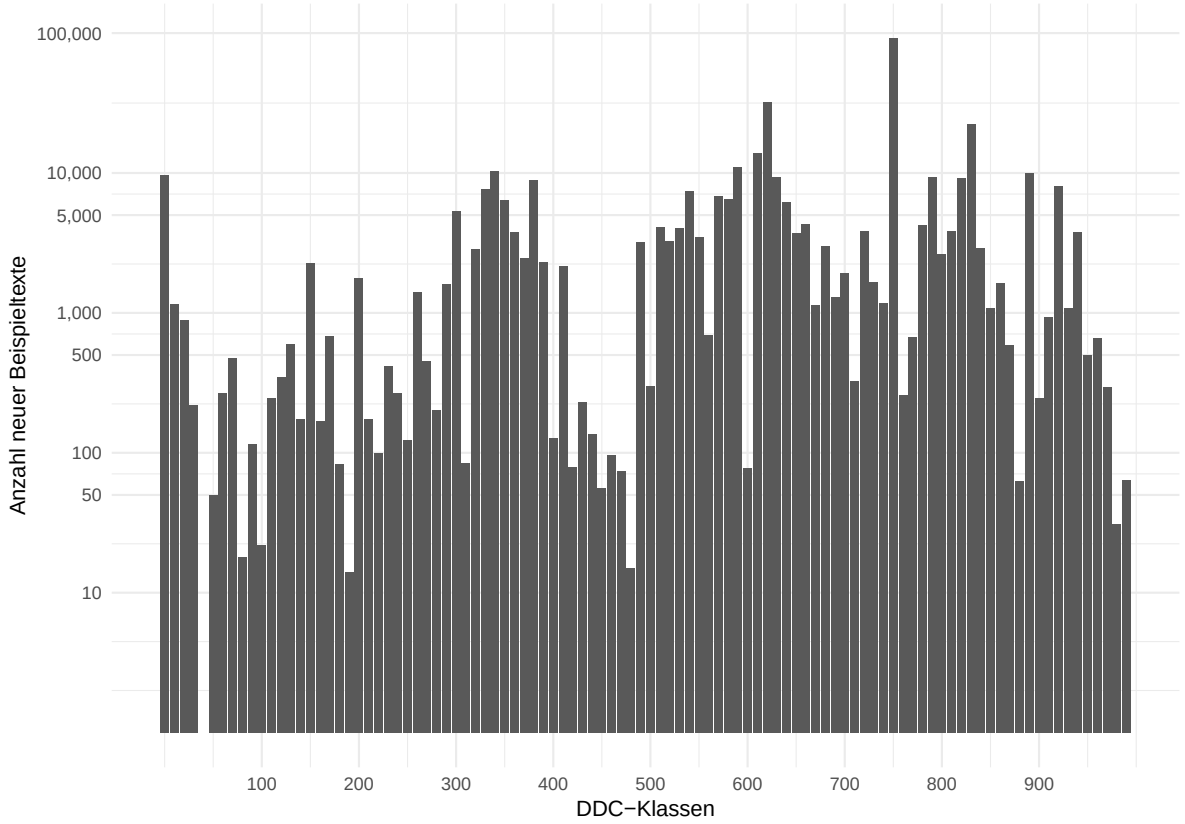

Abbildung 4.3.: Die Grafik zeigt eine grobe Übersicht der Anzahl neu gewonnener Beispieltexte der zweiten DDC-Ebene. Die Y-Achse ist logarithmisch dargestellt.

4.4.2. Dritte Ebene der DDC

Während die Vergrößerung des Trainingskorpus die Ergebnisse der zweiten DDC-Ebene nicht signifikant verbessern konnte, zeigen die Experimente mit der dritten Ebene das Potential des größeren Korpus auf.

Die Ergebnisse in Tabelle 4.6 zeigen, dass bereits mit der einfachsten Version der Score des DDC-Klassifikators von 0,799 auf 0,811 gesteigert werden kann. In dieser annotiert ein Random Forest-Klassifikator, trainiert mit den Kategoriezuordnungen der Wikipedia-Artikel, neue Trainingsdaten (siehe Experiment *RFC6*). Das Trainingskorpus wurde von 13 299 auf 218 504 Beispiele erweitert und umfasst nun 693 statt nur 635 DDC-Klassen der dritten Ebene. Eine grobe Übersicht der Anzahl neu gewonnener Beispieltexte ist in Grafik 4.4 gegeben. Damit steigt die durchschnittliche Anzahl an Beispielen pro Klasse von sehr wenigen 32,56 auf 409,65 Beispiele. Entsprechend den Testbeispielen der zweiten DDC-Ebene sind diese auch auf der dritten Ebene sehr ungleich verteilt, viele neue Trainingsdaten werden somit nicht getestet.

Weitere Experimente, die die automatisch erstellten Kategoriemappings mit einbeziehen (siehe Experimente *RFC1* bis *RFC5*), konnten zwar das Trainingskorpus sowie die Anzahl DDC-Klassen deutlich steigern, erreichen aber mit Scores unter 70 % nicht die erwartete

Tabelle 4.6.: Ergebnisse der Experimente der dritten DDC-Ebene.

		Variante					
	Baseline	RFC1	RFC2	RFC3	RFC4	RFC5	RFC6
Kategoriemapping							
Variante		Auto	Auto	Auto	Auto	Auto	Keine
Oberkategorien		5 Ebenen	1 Ebene	1 Ebene	1 Ebene	1 Ebene	
DDC-Ebene 2		Nein	Nein	Nein	Ja	Ja	
Minimum Artikel		2	2	2	2	10	
Strenges Matching		Nein	Nein	Nein	Nein	Ja	
Auswahl-Klassifikator							
Features		30 389	30 064	30 002	30 036	28 480	18 810
Trainingsbeispiele		80 000	80 000	80 000	80 000	80 000	23 216
Unterkategorien		Keine	Keine	Keine	Keine	Keine	Keine
Estimator Trees		100	100	200	200	200	100
Max Features		`sqrt`	`sqrt`	`sqrt`	`sqrt`	`sqrt`	`sqrt`
Max Samples		100 000	100 000	100 000	100 000	100 000	100 000
F_1-score		0,79	0,79	0,80	0,80	0,80	0,47
Prediction Threshold		50 %	50 %	50 %	50 %	50 %	50 %
DDC-Klassifikator							
Precision@1	0,799	0,658	0,669	0,671	0,671	0,694	**0,811**
Trainingsbeispiele	13 299	1 128 947	1 011 687	1 008 314	1 005 237	1 182 628	218 504
Klassen	635	776	780	780	780	772	693
Klassen/Beispiel	1,55	1,25	1,33	1,33	1,33	1,12	1,30
Beispiele/Klasse	32,56	1 824,58	1 720,84	1 715,96	1 713,32	1 715,42	409,65

Zum Einsatz kommt ein Random Forest-Klassifikator, die Aufteilung in Trainings- und Testmenge ist immer 80 % / 20 %.
Auswahl-Klassifikator *Features*: Anzahl Features, die über die Kategoriezugehörigkeiten gewonnen wurden; *Estimator Trees*: Anzahl der Bäume; *Max Features*: Maximale Anzahl an Features, die bei jedem Split betrachtet werden; *Max Samples*: Maximale Anzahl an Trainingsbeispielen, aufgrund Speichereinschränkungen wird die verfügbare Anzahl an Trainingsbeispielen begrenzt; *Prediction Threshold*: Minimal nötiger Score, der bei der Bestimmung der neuen DDC-Notationen erreicht werden muss.
DDC-Klassifikator *Klassen/Beispiel*: Durchschnittliche Anzahl Klassen pro Trainingsbeispiel; *Beispiele/-Klasse*: Durchschnittliche Anzahl Trainingsbeispiele pro Klasse.

Performance.

4.4.3. Weitere Sprachen

Basierend auf den Erkenntnissen der Experimente mit der deutschen Sprache wurde auch für eine Auswahl von 10 weiteren Sprachen die Auswirkungen des vergrößerten Trainingskorpus untersucht. Dazu wurden, wie in 4.2.4 beschrieben, das deutsche Trainingskorpus als Basis genutzt, um die mittels DDC annotierten Artikel über die Wikidata-Verlinkungen übersetzt. Die Ergebnisse sind in Tabelle 4.7 einsehbar.

Englisch (en) Verbesserungen konnten sowohl auf der zweiten als auch auf der dritten DDC-Ebene erzielt werden. Dazu sind jetzt sämtliche Kategorien der ersten Ebene abgedeckt. Die englische Sprachversion liegt nun auf einem ähnlich hohen Level wie die

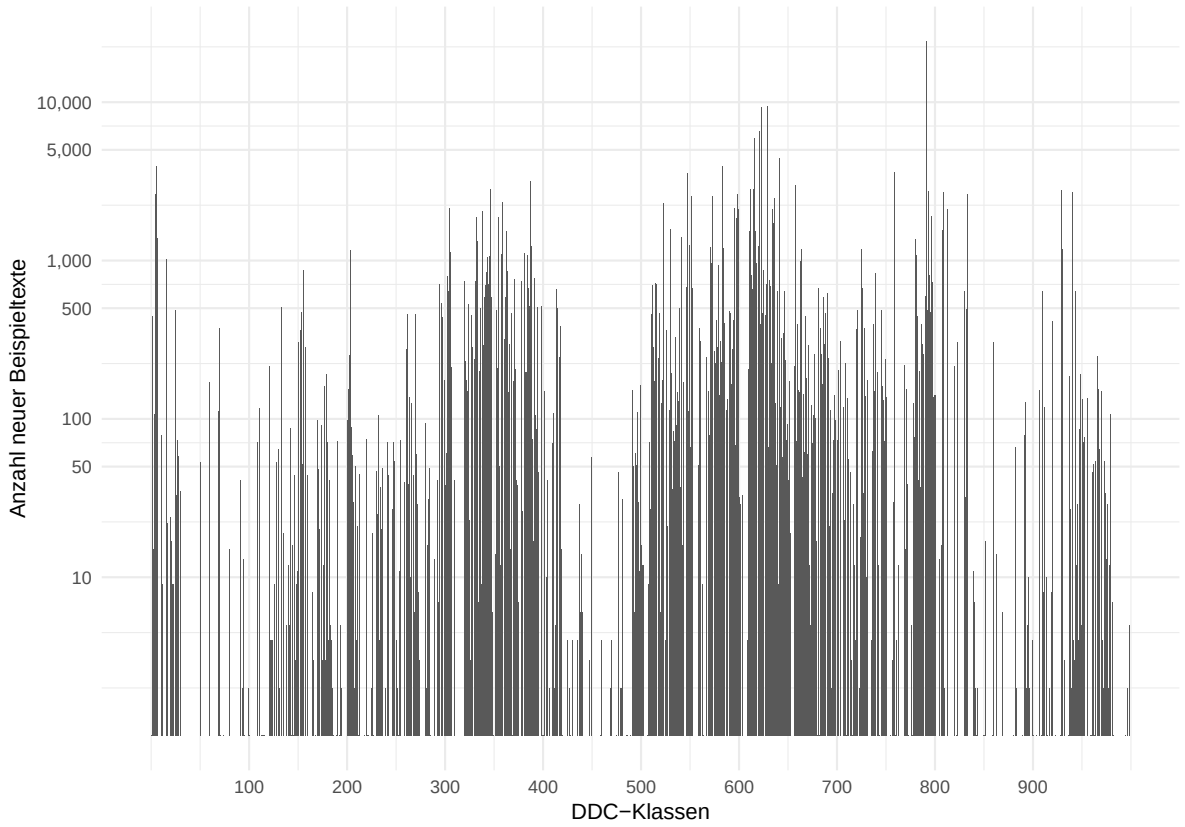

Abbildung 4.4.: Die Grafik zeigt eine grobe Übersicht der Anzahl neu gewonnener Beispieltexte der dritten DDC-Ebene. Die Y-Achse ist logarithmisch dargestellt.

deutsche, mit Scores von 0,874 in der zweiten und 0,807 in der dritten Ebene.

Aragonesisch (`an`) Hier konnte auf keiner der DDC-Ebenen eine Verbesserung erreicht werden. Es zeigt sich allerdings, dass in der zweiten Ebene 49 Klassen keinerlei Testbeispiele besitzen, davon 7 innerhalb der 20 Klassen mit den größten Änderungen durch den erweiterten Trainingskorpus. Auch in der dritten Ebene ist dies zu beobachten, 50 % der 20 meist vergrößerten Klassen sind ohne Beispiele, im Durchschnitt sind nur 0,27 Testbeispiele pro Klasse verfügbar.

Spanisch (`es`) Das vergrößerte Korpus verbessert die Scores der zweiten und dritten DDC-Ebene von 0,797 auf 0,862 und 0,705 auf 0,777.

Französisch (`fr`) Das vergrößerte Korpus verbessert die Scores der zweiten und dritten DDC-Ebene von 0,857 auf 0,860 und 0,681 auf 0,792.

Italienisch (`it`) Eine Verbesserung von 0,738 auf 0,745 war auf der dritten DDC-Ebene möglich.

Limburgisch (`li`) Hier konnte keine Verbesserung erreicht werden. 53 Klassen besitzen keinerlei Testdaten, unter anderem auch die zwei Klassen, die von der Korpuserweite-

rung am meisten profitiert haben. Auch auf der dritten Ebene ist die durchschnittliche Anzahl an Testdaten pro Klasse mit 0,25 sehr gering.

Polnisch (`pl`) Die zweite Ebene der DDC konnte nicht verbessert werden. Auch hier ist sichtbar, dass die am meisten vergrößerte Klasse keine Testbeispiele zur Verfügung hat, obwohl die durchschnittliche Anzahl an Beispielen pro Klasse mit 28,13 auf ähnlichem Niveau ist wie bei erfolgreich verbesserten Sprachen. Da für die dritte DDC-Ebene mit *text2ddc* bisher keine Scores berechnet wurden, wurde dies nun im Rahmen der Bachelorarbeit durchgeführt, dabei konnte ein Score von 0,752 erreicht werden. Dieser konnte mittels des erweiterten Trainingskorpus auf 0,77 verbessert werden.

Russisch (`ru`) Das vergrößerte Korpus verbessert die Scores der zweiten und dritten DDC-Ebene leicht von 0,853 auf 0,857 und 0,761 auf 0,79.

Serbokroatisch (`sh`) Hier konnte keine Verbesserung erreicht werden. Auch hier besitzt die Klasse mit der größten Veränderung auf der zweiten DDC-Ebene im Trainingskorpus, sowie insgesamt 32 Klassen keine Testbeispiele. Ähnlich das Bild auf der dritten Ebene, mit 431 Klassen ohne Beispieltexte.

Urdu (`ur`) Mit 0,351 und 0,234 auf den Ebenen zwei und drei erreichte *text2ddc* hier nur einen sehr geringen Score. Besonders in diesem Fall war die Erwartung hoch, mit einem vergrößerten Korpus deutlich bessere Ergebnisse zu erzielen, was auch erreicht wurde. Der Score konnte auf der zweiten DDC-Ebene auf 0,737 und auf der dritten Ebene auf 0,645 gesteigert werden.

In allen Fällen konnten durch den vergrößerten Trainingskorpus deutlich mehr DDC-Klassen abgedeckt werden. Dies ist besonders im Falle der dritte Ebene interessant, da so der Klassifikator um einige weitere Themenbereiche erweitert werden kann. Die Änderungen der Scores fallen gemischt aus, bei insgesamt 6 Sprachen der zweiten Ebene konnte er verbessert werden, bei 5 sank er. Die Ergebnisse der dritten Ebene zeigen eine Verbesserung bei 8 Sprachen und eine Verschlechterung bei 3. Dabei ist zu sehen, dass sich die dritte Ebene immer verbessert, sofern sich auch die zweite Ebene verbessern konnte. Ein Blick auf die dazugewonnenen Klassen, beziehungsweise das Testkorpus im Ganzen zeigt, dass die Testdaten diese oftmals nicht prüfen, siehe Tabelle 4.8. So finden sich im deutschen Korpus alleine 26 Klassen, die über das erweiterte Korpus neue Trainingsbeispiele gewinnen, allerdings keinerlei Testbeispiele zur Verfügung haben.

4.5. Diskussion und Fazit

Die Aufgabe der Erweiterung des Trainingskorpus von *text2ddc*, einem Klassifikator der beliebige Texte thematisch nach der DDC klassifiziert, konnte umgesetzt werden.

Nach der Aktualisierung des *True Corpus* wurden dazu drei verschiedene Methoden mit mehreren Varianten entwickelt und auf insgesamt 11 Sprachen evaluiert, mit Fokus auf der deutschen Sprachvariante. Es zeigt sich, dass das einfache Hinzufügen von großen Datenmengen problematisch ist, ersichtlich an dem starken Anstieg der durchschnittlichen Klassen pro Trainingsbeispiel. Dies macht es dem Klassifikator deutlich schwieriger, aus den

Beispielen zu lernen. Dazu kommt die ungleiche Verteilung der Trainings- und Testdaten, so werden viele neu gewonnene Beispiele nicht durch Testdaten geprüft. Experimente mit händischer Zuordnung von DDC-Notationen zu Wikipedia-Kategorien zeigen außerdem die Wichtigkeit der Güte dieser Mappings, so konnten bessere Ergebnisse ohne Mappings statt mit automatisch erstellten erzielt werden. Dies ist besonders im Hinblick auf die dritte DDC-Ebene, sowie weitere Sprachvarianten von Bedeutung.

Gute Ergebnisse konnten letztendlich erzielt werden, indem ein Random Forest-Klassifikator die Aufgabe der Erweiterung des Trainingskorpus übernahm (siehe Tabelle 4.7 für alle Ergebnisse). Damit ist eine Verbesserung der Scores von 6 Sprachversionen der zweiten DDC-Ebene, sowie 8 der dritten Ebene umgesetzt worden. Außerdem konnte der Abstand zwischen den Scores der deutschen Sprachversion und den weiteren Sprachen verringert werden.

Tabelle 4.7.: Vergleich der Scores des *text2ddc*-Klassifikators vor und nach der Vergrößerung des Trainingskorpus.

	Alt			Neu		
Sprache	**P@1**	**Train**	**Klassen**	**P@1**	**Train**	**Klassen**
DDC Ebene 2						
Deutsch	0,874	12 655	98	**0,875**	347 148	99
Englisch	0,854	11 200	97	**0,874**	220 191	99
Aragonesisch	**0,818**	1 400	80	0,703	4 641	94
Spanisch	0,797	9 499	94	**0,862**	118 620	99
Französisch	0,857	9 759	94	**0,860**	156 562	99
Italienisch	**0,853**	8 892	93	0,849	116 958	99
Limburgisch	**0,697**	1 255	80	0,561	28 850	97
Polnisch	**0,837**	9 246	95	0,821	103 249	99
Russisch	0,853	9 703	96	**0,857**	116 476	99
Serbokroatisch	**0,820**	5 115	94	0,793	27 286	99
Urdu	0,351	2 243	82	**0,737**	9 588	99
DDC Ebene 3						
Deutsch	0,799	13 299	635	**0,811**	218 504	693
Englisch	0,726	11 794	624	**0,807**	148 100	687
Aragonesisch	**0,772**	1 535	412	0,60	42 720	474
Spanisch	0,705	10 029	599	**0,777**	887 630	668
Französisch	0,681	10 318	604	**0,792**	110 383	678
Italienisch	0,738	9 413	598	**0,754**	87 481	670
Limburgisch	**0,653**	1 381	394	0,532	2 681	451
Polnisch	-[1]	9 775	610	**0,77**	75 295	670
Russisch	0,761	10 210	612	**0,79**	85 071	669
Serbokroatisch	**0,719**	5 482	559	0,710	23 667	615
Urdu	0,234	2 495	455	**0,645**	8 075	554

[1] Da bisher für die polnische Sprachvariante auf der dritten Ebene der DDC kein Klassifikator vorlag, wurde dieser im Rahmen dieser Bachelorarbeit mit den Daten aus dem *True Corpus* trainiert, um einen Vergleich zum erweiterten Trainingskorpus zu ermöglichen, dabei konnte ein Score von 0,752 erreicht werden.

Tabelle 4.8.: Übersicht über das Testkorpus.

Sprache	Testbeispiele	Beispiele/Klasse	Neue ohne
DDC-Ebene 2			
Deutsch	73	39,45	26
Englisch	71	35,08	28
Aragonesisch	45	3,80	37
Spanisch	71	30,02	28
Französisch	73	30,66	26
Italienisch	72	27,68	27
Limburgisch	44	3,17	38
Polnisch	72	28,13	26
Russisch	72	30,25	27
Serbokroatisch	67	15,46	31
Urdu	52	6,78	41
DDC-Ebene 3			
Deutsch	305	4,62	274
Englisch	285	4,08	275
Aragonesisch	46	0,27	178
Spanisch	260	3,38	281
Französisch	262	3,42	286
Italienisch	235	3,19	292
Limburgisch	42	0,25	159
Polnisch	236	3,23	280
Russisch	271	3,54	250
Serbokroatisch	186	1,65	238
Urdu	86	0,49	281

Anzahl Klassen mit Testbeispielen, durchschnittliche Anzahl Beispiele pro Klassen (bezieht auch Klassen mit ein, die keine Beispiele aufweisen), sowie Anzahl Klassen, die über das vergrößerte Trainingskorpus neue Beispiele erhielten, aber keinerlei Testbeispiele für diese Klasse enthalten.

5. Zusammenfassung und Weiterführende Arbeiten

5.1. Zusammenfassung

In dieser Bachelorarbeit wurde die Erweiterung des Trainingskorpus des *text2ddc*-Klassifikators vorgestellt.

Nach der einleitenden Motivation dieser Arbeit, sowie der Erläuterung der Grundlagen, auf denen diese Arbeit basiert, wurde das Klassifikationsmodell sowie der aktuelle Stand von *text2ddc* vorgestellt. Der Mangel an Trainingsdaten als großes Problem zum Lernen des Klassifikators wurde ausgemacht, sowie die sich daraus ergebenden Aufgaben erläutert.

Ausführlich wurde die Aktualisierung der aktuellen Trainingsdaten beschrieben. Dabei wurde im Detail erläutert, wie die Daten aus Wikipedia und der GND abgerufen sowie kombiniert werden, um schlussendlich Wikipedia-Artikel mit DDC-Notationen zu versehen, die *text2ddc* zum Lernen verwenden kann.

Es folgte die Beschreibung der verschiedenen Methoden und Varianten, die zum Einsatz kommen um weitere Texte als Beispiele zu gewinnen. Dies beinhaltet das automatische sowie händische Erstellen von Mappings der Wikipedia-Kategorien auf DDC-Klassen, sowie das Erschließen neuer Artikel mittels drei verschiedener Algorithmen: Einer einfachen Version, einer heuristischen Version sowie eine, sich als die Erfolgreichste herausstellende, Version die einen Random Forest-Klassifikator einsetzt.

Anschließend wurden die Features und Einschränkungen der Trainingstexte vorgestellt, die aus dem Korpus der vorherigen Veröffentlichung von *text2ddc* stammen. Die zum Trainieren verwendete Software *fastText*, sowie die verwendeten Hyperparameter, wurden besprochen, bevor danach eine Beschreibung und Evaluation aller durchgeführten Experimente gegeben wurde.

Das Fazit kam zu dem Schluss, dass die Vergrößerung des Trainingskorpus in vielen der getesteten Fällen die Güte der Klassifikation verbessern kann, und beschrieb abschließend einige Probleme dieses Ansatzes. Aus diesen, sowie aus den durch die Experimente gewonnenen Erkenntnissen, ergeben sich eine Reihe von weiteren möglichen zukünftigen Arbeiten, die im nächsten Abschnitt kurz vorgestellt werden.

5.2. Weiterführende Arbeiten

Das Trainingskorpus basiert auf den Wikipedia-Texten eines Dumps aus 2016, die Nutzung neuerer Artikel-Texte würde es ermöglichen, das Korpus weiter zu vergrößern, siehe auch Abschnitt 4.3.1.

Besonderer Augenmerk wurde auf die Experimente mit der deutschen Sprachversion gelegt, eine Vorverarbeitung der Korpora der anderen Sprachen (mit zum Beispiel Lemmatisierung, POS-Tagging oder Word Sense-Disambiguation) könnte die Ergebnisse weiter verbessern.

Ebenso wurden die Beispiele über die Übersetzungen der deutschen Trainingsdaten gewonnen, was zu einem Verlust an Daten führt, da nicht alle Artikel in den verschiedenen Wikipedia-Sprachversionen verfügbar sind. Es sollte geprüft werden, ob diese Sprachversionen mehr und/oder bessere, Ergebnisse liefern können, wenn die komplette Erweiterung des Korpus basierend auf ihren Kategoriezuordnungen erstellt wird und nicht ausschließlich über die Übersetzungen abgeleitet wird. Die benötigten Daten hierfür wurden bereits im Rahmen dieser Bachelorarbeit vorbereitet.

An den letzten Punkt anschließend sollten weitere Parameterstudien durchgeführt werden, sowohl bei dem Training des *text2ddc*-Klassifikators mit *fastText*, als auch im Besonderen bei der automatischen Berechnung der Wikipedia-Kategoriezuordnungen zu DDC-Klassen, sowie dem Training des Artikelauswahl-Klassifikators.

Anpassungen des Testsets würden die Vergleichbarkeit mit der älteren Veröffentlichung von *text2ddc* erschweren, aber unter Umständen neu gewonnene Klassen mit in die Bewertung des Klassifikators aufnehmen.

Zuletzt könnte versucht werden, bereits den *True Corpus* zu vergrößern, indem weitere DDC-annotierte Quellen mit eingebunden werden, zum Beispiel über weitere Wikidata-Verlinkungen.

Literatur

Aggarwal, Charu C. und Cheng Xiang Zhai (2013). *Mining Text Data*. Bd. 9781461432, S. 1–522. ISBN: 9781461432234. DOI: 10.1007/978-1-4614-3223-4.

Alex, Heidrun (2018). „Die Dewey-Dezimalklassifikation (DDC)“. In: *Klassifikationen in Bibliotheken*. Berlin, Boston: De Gruyter Saur, S. 65–110. DOI: 10.1515/9783110299250-003. URL: https://www.degruyter.com/view/book/9783110299250/10.1515/9783110299250-003.xml.

Baumartz, Daniel, Tolga Uslu und Alexander Mehler (2018). „LTV: Labeled Topic Vector“. In: *Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics: System Demonstrations, August 20-26*. Santa Fe, New Mexico, USA: The COLING 2018 Organizing Committee.

Baykan, Eda, Monika Henzinger, Ludmila Marian und Ingmar Weber (2009). „Purely URL-based Topic Classification Categories and Subject Descriptors“. In: *In Proceedings of the 18th International World Wide Web Conference (WWW 2009)*, S. 1109–1110.

Behrens-Numann, Renate und Barbara Pfeifer (2011). „Die Gemeinsame Normdatei - ein Kooperationsprojekt“. In: *Dialog mit Bibliotheken* 1, S. 37–40.

Bird, Steven, Edward Loper und Ewan Klein (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc.

Blanck, Sven, Andreas Niekler und Marc Kaulisch (2019). „Augmenting a research information system with automatically acquired category and keyword information“. In: *17th International Conference on Scientometrics and Informetrics, ISSI 2019 - Proceedings* 2.4, S. 2002–2013. DOI: 10.31219/osf.io/bxuaf.

Blei, David M. und Jon D. McAuliffe (2010). *Supervised Topic Models*. arXiv: 1003.0783 [stat.ML].

Blei, David M., Andrew Y. Ng und Michael I. Jordan (2003). „Latent Dirichlet Allocation“. In: *Journal of Machine Learning Research* 3, S. 993–1022.

Bracewell, David B., Jiajun Yan, Fuji Ren und Shingo Kuroiwa (2009). „Category Classification and Topic Discovery of Japanese and English News Articles“. In: *Electronic Notes in Theoretical Computer Science* 225.C, S. 51–65. ISSN: 15710661. DOI: 10.1016/j.entcs.2008.12.066.

Breiman, Leo (Okt. 2001). „Random Forests“. In: *Machine Learning* 45.1, S. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

Cortes, Corinna und Vladimir Vapnik (Sep. 1995). „Support-vector networks“. In: *Machine Learning* 20.3, S. 273–297. ISSN: 0885-6125. DOI: 10.1007/BF00994018. URL: http://link.springer.com/10.1007/BF00994018.

Crain, Steven P., Ke Zhou und Hongyuan Yang Shuang-Hong and Zha (2012). „Dimensionality Reduction and Topic Modeling: From Latent Semantic Indexing to Latent Dirichlet Allocation and Beyond“. In: *Mining Text Data*. Hrsg. von Charu C. Aggarwal und ChengXiang Zhai. Boston, MA: Springer US, S. 129–161. ISBN: 978-1-4614-3223-4. DOI: 10.1007/978-1-4614-3223-4_5.

Deutsche Nationalbibliothek (2020). *Gemeinsame Normdatei (GND)*. [Online; Stand 21. Mai 2020]. URL: https://www.dnb.de/DE/Professionell/Standardisierung/GND/gnd.html.

Ferrucci, David und Adam Lally (Sep. 2004). „UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment". In: *Natural Language Engineering* 10.3-4, S. 327–348. DOI: 10.1017/S1351324904003523.

Gleim, Rüdiger, Alexander Mehler und Sung Y. Song (2018). „WikiDragon: A Java Framework For Diachronic Content And Network Analysis Of MediaWikis". In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12*. LREC 2018. Miyazaki, Japan.

Goldstone, Andrew und Ted Underwood (2012). „What can topic models of PMLA teach us about the history of literary scholarship". In: *Journal of Digital Humanities* 2.1, S. 39–48.

Goller, Christoph, Joachim Löning, Thilo Will und Werner Wolff (2000). „Automatic Document Classification: A thorough Evaluation of various Methods". In: *ISI* 2000, S. 145–162.

Golub, Koraljka, Johan Hagelbäck und Anders Ardö (2020). „Automatic Classification of Swedish Metadata Using Dewey Decimal Classification: A Comparison of Approaches". In: *Journal of Data and Information Science* 5.1, S. 18–38. ISSN: 2543-683X. DOI: 10.2478/jdis-2020-0003. URL: https://content.sciendo.com/view/journals/jdis/5/1/article-p18.xml.

Hemati, Wahed, Tolga Uslu und Alexander Mehler (2016). „TextImager: a Distributed UIMA-based System for NLP". In: *Proceedings of the COLING 2016 System Demonstrations*. Federated Conference on Computer Science und Information Systems. Osaka, Japan.

Hunziker, Alex, Hasanagha Mammadov, Wahed Hemati und Alexander Mehler (2019). „Corpus2Wiki: A MediaWiki-based Tool for Automatically Generating Wikiditions in Digital Humanities". In: *INF-DH-2019*. Hrsg. von Manuel Burghardt und Claudia Müller-Birn. Bonn: Gesellschaft für Informatik e.V.

Husby, Stephanie D und Denilson Barbosa (2012). „Topic Classification of Blog Posts Using Distant Supervision". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, S. 28–36.

Ikonomakis, M., S. Kotsiantis und V. Tampakas (2005). „Text Classification Using Machine Learning Techniques". In: *WSEAS TRANSACTIONS on COMPUTERS* 4.8, S. 966–974.

Jacob, Elin K. (2004). „Classification and categorization: A difference that makes a difference". In: *Library Trends* 52.3, S. 515–540.

Jenkins, Charlotte, Mike Jackson, Peter Burden und Jon Wallis (1998). „Automatic classification of Web resources using Java and Dewey Decimal Classification". In: *Computer Networks and ISDN Systems* 30.1-7, S. 646–648. ISSN: 13891286. DOI: 10.1016/s0169-7552(98)00035-x.

Joulin, Armand, Edouard Grave, Piotr Bojanowski und Tomas Mikolov (Apr. 2017). „Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, S. 427–431.

Katakis, Ioannis, Grigorios Tsoumakas und Ioannis Vlahavas (2008). „Multilabel Text Classification for Automated Tag Suggestion". In: *Proceedings of the ECML/PKDD*. Bd. 18, S. 5.

Kett, Attila (2020). „text2city: Räumliche Visualisierung textueller Strukturen". Goethe University of Frankfurt. Bachelor. Goethe University of Frankfurt, Institute of Computer Sci-

ence und Mathematics, Text Technology Lab, S. 61. URL: http://publikationen.ub.uni-frankfurt.de/files/53217/text2City__Raeumliche_Visualisierung_textueller_Strukturen.pdf.

Kuhn, Tamara J. (1999). „Classifying Newspapers Using Dewey Decimal Classification“. In: *Library Resources & Technical Services* 43.2, S. 106–113. DOI: 10.5860/lrts.43n2.106. URL: https://www.journals.ala.org/index.php/lrts/article/view/4946.

La Rosa, Massimo, Antonino Fiannaca, Riccardo Rizzo und Alfonso Urso (2015). „Probabilistic topic modeling for the analysis and classification of genomic sequences“. In: *BMC Bioinformatics* 16.6, S. 1–9. ISSN: 14712105. DOI: 10.1186/1471-2105-16-S6-S2.

Lewis, David D. (1991). „Evaluating Text Categorization“. In: *Proceedings of the Workshop on Speech and Natural Language*. HLT '91. Pacific Grove, California: Association for Computational Linguistics, S. 312–318. DOI: 10.3115/112405.112471.

Liu, Lin, Lin Tang, Wen Dong, Shaowen Yao und Wei Zhou (2016). „An overview of topic modeling and its current applications in bioinformatics“. In: *SpringerPlus* 5.1. ISSN: 21931801. DOI: 10.1186/s40064-016-3252-8.

Lorenz, Bernd (2018). „Zur Theorie und Terminologie der bibliothekarischen Klassifikation“. In: *Klassifikationen in Bibliotheken*. Berlin, Boston: De Gruyter Saur, S. 1–22. DOI: 10.1515/9783110299250-001. URL: https://www.degruyter.com/view/book/9783110299250/10.1515/9783110299250-001.xml.

Lösch, Mathias, Ulli Waltinger, Wolfram Horstmann und Alexander Mehler (2011). „Building a DDC-annotated Corpus from OAI Metadata“. In: *Journal of Digital Information* 12.2.

Madjarov, Gjorgji, Dragi Kocev, Dejan Gjorgjevikj und Sašo Džeroski (2012). „An extensive experimental comparison of methods for multi-label learning“. In: *Pattern Recognition* 45.9, S. 3084–3104. ISSN: 00313203. DOI: 10.1016/j.patcog.2012.03.004.

Marsland, Stephen (2014). *Machine Learning: An Algorithmic Perspective, Second Edition*, S. 457. ISBN: 9781466583337. URL: https://b-ok.cc/book/2543746/ef80cb.

McCallum, Andrew Kachites (1999). „Multi-Label Text Classification with a Mixture Model Trained by EM“. In: *AAAI 99 Workshop on Text Learning*.

Mehler, Alexander, Rüdiger Gleim, Wahed Hemati und Tolga Uslu (2017). „Skalenfreie online soziale Lexika am Beispiel von Wiktionary“. In: *Proceedings of 53rd Annual Conference of the Institut für Deutsche Sprache (IDS), March 14-16, Mannheim, Germany*. Hrsg. von Stefan Engelberg, Henning Lobin, Kathrin Steyer und Sascha Wolfer. In German. Title translates into: Scale-free online-social Lexika by Example of Wiktionary. Berlin: De Gruyter.

Mehler, Alexander, Tolga Uslu, Rüdiger Gleim und Daniel Baumartz (2019). „text2ddc meets Literature - Ein Verfahren für die Analyse und Visualisierung thematischer Makrostrukturen“. In: *Proceedings of the 6th Digital Humanities Conference in the German-speaking Countries, DHd 2019*. DHd 2019. Frankfurt, Germany.

Mehler, Alexander und Ulli Waltinger (2009). „Enhancing Document Modeling by Means of Open Topic Models: Crossing the Frontier of Classification Schemes in Digital Libraries by Example of the DDC“. In: *Library Hi Tech* 27.4, S. 520–539.

Mikolov, Tomas, Kai Chen, Greg Corrado und Jeffrey Dean (2013). „Efficient estimation of word representations in vector space“. In: *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. arXiv: 1301.3781.

Mitchell, Joan S. und Diane Vizine-Goetz (Dez. 2009). „Dewey Decimal Classification (DDC)“. In: *Encyclopedia of Library and Information Sciences, Third Edition*. Bd. 23. 4. CRC Press, S. 1507–1517. DOI: 10.1081/E-ELIS3-120043240. URL: https://www.taylorfrancis.com/books/e/9780203757635.

Möller, Gerhard, Kai-Uwe Carstensen, Bernd Diekmann und Han Wätjen (1999). *Automatic Classification of the World-Wide Web using the Universal Decimal Classification*.

Natarajan, Prem u. a. (2007). „Finding structure in noisy text: topic classification and unsupervised clustering“. In: *International Journal of Document Analysis and Recognition (IJDAR)*. Bd. 10. 3, S. 187–198. DOI: 10.1007/s10032-007-0057-x.

Nigam, Kamal, Andrew Kachites McCallum, Sebastian Thrun und Tom Mitchell (2000). „Text Classification from Labeled and Unlabeled Documents using EM“. In: *Machine Learning* 39.2-3, S. 103–134. DOI: 10.1023/A:1007692713085.

OCLC (2019). „Introduction to the Dewey Decimal Classification“. In: *DDC 23 Introduction*. URL: https://www.oclc.org/content/dam/oclc/dewey/versions/print/intro.pdf.

Pedregosa, Fabian u. a. (2011). „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12, S. 2825–2830.

Quercia, Daniele, Harry Askham und Jon Crowcroft (2012). „TweetLDA: Supervised topic classification and link prediction in Twitter“. In: *Proceedings of the 4th Annual ACM Web Science Conference, WebSci'12*. ISBN: 9781450312288. DOI: 10.1145/2380718.2380750.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: https://www.R-project.org/.

Ramage, Daniel, David Hall, Ramesh Nallapati und Christopher D. Manning (2009). „Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-Labeled Corpora“. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*. EMNLP '09. Singapore: Association for Computational Linguistics, S. 248–256. ISBN: 9781932432596.

Shiri, Ali (Jan. 2004). „Introduction to Modern Information Retrieval (2nd edition)“. In: *Library Review* 53.9, S. 462–463. ISSN: 0024-2535. DOI: 10.1108/00242530410565256.

Sokolova, Marina (2018). „Big Text advantages and challenges: classification perspective“. In: *International Journal of Data Science and Analytics* 5.1, S. 1–10. ISSN: 2364-415X. DOI: 10.1007/s41060-017-0087-5.

Sorower, Mohammad S (2010). „A Literature Survey on Algorithms for Multi-label Learning“. In: *Oregon State University, Corvallis*, S. 1–25.

Uslu, Tolga, Wahed Hemati, Alexander Mehler und Daniel Baumartz (2017). „TextImager as a generic interface to R“. In: *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of the Software Demonstrations*. ISBN: 9781510838604.

Uslu, Tolga, Alexander Mehler und Daniel Baumartz (2019). „Computing Classifier-based Embeddings with the Help of text2ddc“. In: *Proceedings of the 20th International Conference on Computational Linguistics and Intelligent Text Processing, (CICLing 2019)*. CICLing 2019. La Rochelle, France.

Uslu, Tolga, Alexander Mehler, Daniel Baumartz, Alexander Henlein und Wahed Hemati (2018). „fastSense: An Efficient Word Sense Disambiguation Classifier“. In: *Proceedings of*

the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12. LREC 2018. Miyazaki, Japan.

Uslu, Tolga, Alexander Mehler, Andreas Niekler und Daniel Baumartz (2018). „Towards a DDC-based Topic Network Model of Wikipedia“. In: *Proceedings of 2nd International Workshop on Modeling, Analysis, and Management of Social Networks and their Applications (SOCNET 2018), February 28, 2018*.

van der Walt, Stéfan, S. Chris Colbert und Gaël Varoquaux (März 2011). „The NumPy Array: A Structure for Efficient Numerical Computation“. In: *Computing in Science & Engineering* 13.2, S. 22–30. ISSN: 1558-366X. DOI: 10.1109/MCSE.2011.37.

Vega-Carrillo, H. R. u. a. (Mai 2007). „Artificial neural networks technology for neutron spectrometry and dosimetry“. In: *Radiation Protection Dosimetry* 126.1-4, S. 408–412. ISSN: 0144-8420. DOI: 10.1093/rpd/ncm084. eprint: https://academic.oup.com/rpd/article-pdf/126/1-4/408/4563634/ncm084.pdf.

Voß, Jakob u. a. (2014). *Normdaten in Wikidata*.

Wang, Jun (2009). „An extensive study on automated Dewey Decimal Classification“. In: *Journal of the American Society for Information Science and Technology* 60.11, S. 2269–2286. DOI: 10.1002/asi.21147. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/asi.21147. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21147.

Wickham, Hadley (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4. URL: https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry und Kirill Müller (2020). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.0. URL: https://CRAN.R-project.org/package=dplyr.

Wikidata (2020). *Wikidata*. [Online; Stand 21. Mai 2020]. URL: https://www.wikidata.org/w/index.php?title=Wikidata:Main_Page&oldid=1086709037.

Wikipedia (2020). *Wikipedia — Wikipedia, Die freie Enzyklopädie*. [Online; Stand 21. Mai 2020]. URL: https://de.wikipedia.org/w/index.php?title=Wikipedia&oldid=200133825.

Zelikovitz, Sarah (2004). „Transductive LSI for short text classification problems“. In: *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004* 2.Vapnik 1998, S. 556–561.

A. DDC-zu-Wikipedia-Kategoriemapping

Das im Zuge dieser Bachelorarbeit erstellte Mapping von DDC-Klassen der zweiten Ebene auf Wikipedia-Kategorien ist hier einsehbar. Siehe Kapitel 4.2.1 für eine Beschreibung.

Tabelle A.1.: Mapping der DDC-Notationen der zweiten Ebene auf die Kategorien der deutschen Wikipedia.

DDC	Wikipedia-Kategorie
000	Informatik, Wissen
010	Bibliografie
020	Bibliothekswesen, Bibliothek
030	Lexikon_oder_Enzyklopädie
040	[unbesetzt]
050	Sammelwerk, Zeitschrift
060	Organisationen, Museum
070	Verlagswesen, Medien, Journalismus
080	Sammelwerk
090	Handschrift, Antiquariat
100	Philosophie
110	Metaphysik
120	Erkenntnistheorie
130	Okkultismus, Esoterik, Parapsychologie
140	Philosophische_Strömung
150	Psychologie
160	Philosophische_Logik
170	Ethik
180	Philosophie_des_Mittelalters, Philosophie_der_Antike, Östliche_Philosophie
190	Philosophie_der_Moderne, Philosophie_der_Neuzeit
200	Religion
210	Religionswissenschaft, Religionsphilosophie
220	Bibel
230	Christentum, Christliche_Theologie
240	Glaubenspraxis
250	Praktische_Theologie, Ordensgemeinschaft
260	Soziale_Arbeit
270	Christentumsgeschichte
280	Christliche_Konfession

290	Liste_(Religion)
300	Sozialwissenschaft, Soziologie
310	Statistik
320	Politik
330	Wirtschaft
340	Recht
350	Militärwesen, Öffentliche_Verwaltung
360	Soziologie, Sozialstaat
370	Bildung, Pädagogik
380	Handel, Kommunikation, Verkehrswesen
390	Brauch, Umgangsform, Volkskunde
400	Sprache
410	Sprachwissenschaft
420	Englische_Sprache, Altenglisch
430	Deutsche_Sprache, Germanische_Sprachen
440	Französische_Sprache, Romanische_Sprachen
450	Italienische_Sprache, Rumänische_Sprache
460	Spanische_Sprache, Portugiesische_Sprache
470	Latein, Italische_Sprachen
480	Griechische_Sprache
490	{Einzelsprache_als_Thema}
500	Naturwissenschaft
510	Mathematik
520	Astronomie
530	Physik
540	Chemie
550	Geowissenschaft, Geologie
560	Fossil, Paläontologie
570	Biowissenschaften, Biologie
580	Botanik
590	Zoologie, Lebewesen
600	Technik
610	Medizin, Gesundheit
620	Ingenieurwissenschaft
630	Landwirtschaft, Land-_und_Forstwirtschaft,_Fischerei
640	Hauswirtschaft, Familie
650	Management, Öffentlichkeitsarbeit
660	Technische_Chemie
670	Produktion
680	Produktion_nach_Produkt
690	Bauwesen

700	Kunst_und_Kultur
710	Landschaftskunde, Landespflege,Landschaftsplanung, Landschaftsarchitektur, Raumplanung
720	Architektur
730	Bildhauerei, Keramik, Metallkunst
740	Zeichnen, Angewandte_Kunst
750	Malerei
760	Grafik
770	Fotografie, Computerkunst
780	Musik
790	Sport, Spiele, Unterhaltung
800	Literatur, Rhetorik, Literaturwissenschaft
810	Literatur_(Vereinigte_Staaten)
820	Literatur_(Englisch), (Literatur_(Altenglisch))
830	Literatur_(Deutsch)
840	Literatur_(Französisch)
850	Literatur_(Rätoromanisch), Literatur_(Rumänisch), Literatur_(Italienisch)
860	Literatur_(Spanisch), Literatur_(Portugiesisch)
870	Literatur_(Latein)
880	Literatur_(Griechisch)
890	{Literatur_nach_Sprache}
900	Geschichte
910	Geographie, Reise
920	Biografie, Genealogie
930	Archäologie, Altertum
940	Europäische_Geschichte
950	Asiatische_Geschichte
960	Geschichte_(Afrika)
970	Nordamerikanische_Geschichte
980	Südamerikanische_Geschichte
990	{Geschichte_nach_Kontinent}

B. Softwaredokumentation

In diesem Abschnitt werden die entwickelten und eingesetzten Softwarekomponenten kurz vorgestellt. Nicht beschrieben werden kleinere Helfer-Skripte, die zum Beispiel zum Säubern der DDC-Notationen oder zum Erstellen der Graphen und ähnlichen Funktionen eingesetzt werden. Die meisten Funktionen sind über das Python-Programm `cli.py` als *Command Line Interface* (CLI) verfügbar.

B.1. Aktualisierung der Daten

Um einen problemlosen Export der Daten zu gewährleisten, ist es empfehlenswert, die Datenbanken der GND und Wikidata lokal einzurichten.

B.1.1. Datenbanken

scripts/import_gnd.sh Der N-Triples-Dump der GND (siehe Abschnitt 4.1.1) lässt sich direkt ohne Probleme mittels des `Bulk Data Loader`[1] in eine Blazegraph-Instanz importieren.

Wikidata Aufgrund der Größe der Wikidata-Datenbank ist es empfehlenswert, den Import in Blazegraph über die von Wikimedia bereitgestellten Tools zu erledigen, siehe dazu das wikidata-query-rdf-Projekt[2]. Das Aufteilen des Dumps sollte dabei in möglichst kleine Stücke erfolgen, dies verhindert Timeouts.

B.1.2. Laden der Daten

Die folgenden Programme laden die benötigten Daten aus den angelegten Datenbanken sowie Dumps und konvertieren sie in ein passendes Datenformat, meist das von Python sehr gut unterstützte JSON.

blazegraph_loader/gnd.py Lädt die GND-Daten mittels SPARQL-Queries unter Verwendung der SPARQLWrapper-Bibliothek[3].

blazegraph_loader/wikidata.py Lädt alle benötigten Wikidata-Daten, entsprechend dem Vorgehen bei der GND.

wikipedia_category_tree Beinhaltet das C++-Projekt, das die Extraktion und Verarbeitung des Wikipedia-Kategoriebaums aus den Dumps durchführt, unter Verwendung der LEMON-Bibliothek.

[1] `https://wiki.blazegraph.com/wiki/index.php/Bulk_Data_Load`
[2] `https://github.com/wikimedia/wikidata-query-rdf`
[3] `https://rdflib.dev/sparqlwrapper/`

B.2. Vergrößern des Trainingskorpus

Die folgenden Skripte erstellen das Korpus, bestehend aus Trainings- und Testdateien sowie allen benötigten Zwischendaten, wie die kombinierten Daten des *True Corpus*.

ddc/ddc_combine.py Kombiniert die abgerufenen GND-, Wikidata- und Wikipedia-Daten um das *True Corpus* zu erstellen.

ddc/ddc_wikipedia_mapper_manual.py Erstellt das DDC-zu-Wikipedia-Kategoriemapping anhand der händisch ausgewählten Zuordnungen, diese liegen als separate Datei vor und sind im Anhang A einsehbar.

ddc/ddc_wikipedia_mapper.py Erstellt automatisiert das DDC-zu-Wikipedia-Kategorimapping. Zum Einsatz kommen die Bibliotheken NLTK (Bird, Loper und Klein 2009) sowie numpy (van der Walt, Colbert und Varoquaux 2011).

ddc/ddc_infer_wikipedia_simple.py Führt die Erweiterung des Trainingskorpus mit der einfachen Methode durch.

ddc/ddc_infer_wikipedia_article.py Führt die Erweiterung des Trainingskorpus über den heuristischen Algorithmus durch.

ddc/ddc_infer_wikipedia_classifier.py Führt die Erweiterung des Trainingskorpus durch die Bestimmung der DDC-Klassen mittels eines Random Forest-Klassifikators durch, hier wird die scikit-learn-Bibliothek (Pedregosa u. a. 2011) verwendet.

ddc/ddc_translate_inferred.py Übersetzt die Trainingsdaten anhand der Wikipedia-Artikelverlinkung der verschiedenen Sprachversionen.

B.3. Training und Evaluation

traintest/build_traintest.py Kombiniert die verschiedenen Daten (*True Corpus* sowie erweiterte Artikel) um die Trainings- und Testmengen zu erstellen, unter Einhaltung der alten Aufteilung des *text2ddc*-Trainingskorpus.

traintest/make_fasttext.py Die Daten des vorherigen Schrittes werden in ein Format konvertiert, das *fastText* zum Trainieren einsetzt.

fasttext.sh Das Training der Klassifikatoren erfolgt über *fastText* (Joulin u. a. 2017) und wird mit diesem Bash-Skript gesteuert. Es kümmert sich um die Dokumentation der eingesetzten Daten und Hyperparameter und führt automatisch das Testen des Klassifikators durch.

stats_*.py Mehrere Python-Skripte die verschiedene Statistiken der gesammelten und erstellten Daten mit Hilfe der numpy-Bibliothek (van der Walt, Colbert und Varoquaux 2011) berechnen.