Technische Hochschule Wildau
Technical University of Applied Sciences

Masterarbeit

zur Erlangung des akademischen Grades
Master

Technische Hochschule Wildau

Fachbereich Wirtschaft, Informatik, Recht

Studiengang Bibliotheksinformatik (M. Sc.)

Thema (deutsch): Automatisierte Vergabe von Notationen der Regensburger Verbundklassifikation RVK mithilfe des Toolkits Annif

Thema (englisch): Automated subject classification for classes of the Regensburger Verbundklassifikation using the toolkit Annif

Autor/in:	Elisabeth Mecking
Seminargruppe:	BIM/20
Betreuer/in:	Dipl.-Informatiker Sascha Szott
Zweitgutachter/in:	Dr. Frank Seeliger
Spätestmögliche Abgabe:	24.07.2023

Zusammenfassung

Das Toolkit Annif ist ein modulares Werkzeug zur automatisierten Sacherschließung, mit dem über verschiedene Verfahren des maschinellen Lernens Schlagwörter oder Notationen zu Dokumenten vorgeschlagen werden. In dieser Arbeit wird untersucht, ob mit Annif Ergebnisse erzielt werden können, die einen praktischen Einsatz für Bibliotheken im deutschsprachigen Raum rechtfertigen. Getestet wird die Vergabe von Notationen der Regensburger Verbundklassifikation (RVK) an deutschsprachigen monografischen Texten. Dabei wird auch untersucht, wie aufwändig die Beschaffung und Aufbereitung von geeigneten Daten für das Training ist. Es wird eine Auswahl von drei angebotenen Verfahren evaluiert. Hierzu werden Metadaten mit RVK-Notationen der Hauptgruppe S beispielhaft ausgewertet. Zusätzlich wird getestet, ob ein Training mit Volltexten die Qualität der Ergebnisse verbessert. Die Arbeit zeigt auf, welches Verfahren für die ausgewählten Daten am besten geeignet ist und erläutert, welche Herausforderungen es in Bezug auf die Vergabe von RVK-Notationen gibt und welche Aspekte bei der Zusammensetzung der Trainingsdaten besonders berücksichtigt werden müssen.

Abstract

Annif is a modular toolkit for automated subject indexing that is used to suggest subject keywords or class notations for documents using various machine learning methods. This thesis examines whether Annif can be used to achieve results that justify its practical use for libraries in German-speaking countries. An assignment of class notations of the Regensburger Verbundklassifikation (RVK, a common German classification system) to German-language monographic resources will be tested. The effort which is needed to receive data and prepare it for use in Annif will also be assessed. A selection of metadata which contain class notations of topics linked to the letter S in the RVK will be used to evaluate the system. In addition, it will be tested whether a use of full texts for training will increase the quality of the results. The work shows which method is best suited for the selected data and explains the challenges which arise regarding the assignment of RVK class notations and discusses aspects which must be given special consideration in the composition of the training data.

Inhaltsverzeichnis

1 Einleitung

Eine Aufgabe von Bibliotheken besteht darin, Ressourcen formal und inhaltlich zu erschließen, damit sie im Bestand gefunden werden können und es Nutzenden möglich ist, Material zu ausgewählten Themen zu finden. Für die Aufstellung von physischen Ressourcen nutzen viele Bibliotheken Klassifikationssysteme, wie z. B. die Regensburger Verbundklassifikation (RVK). Discovery-Systeme und Online-Kataloge bieten die Möglichkeit der thematischen Suche mithilfe von Notationen, also Systemstellen des jeweiligen Klassifikationssystems. Über diese Suche werden im Idealfall alle Ressourcen angezeigt, die zu einem bestimmten Thema, das, je nach Notation, mehr oder weniger spezifisch ist, im Bestand der Bibliothek vorhanden sind. Die inhaltliche Erschließung von Ressourcen ist jedoch aufwändig und durch Fachkräftemangel in den Bibliotheken haben die dafür zuständigen Personen zusätzlich zur inhaltlichen Erschließung von Ressourcen oftmals viele weitere Aufgaben zu bearbeiten. Die Bearbeitung elektronischer Ressourcen, die unabhängig von Regalflächen lizenziert werden können, ist bei der klassifikatorischen Inhaltserschließung im Vergleich zu anderen Aufgaben niedrig priorisiert, da die Klassifikation hier nicht zur Aufstellung benötigt wird. Ähnliches gilt bei Ressourcen, die beispielsweise in einem Magazin mit platzsparender numerischer Aufstellung aufbewahrt werden.

Um Nutzenden die Möglichkeit zu geben, bei einer thematischen Suche mithilfe von Notationen einen umfassenden Überblick über den gesamten Bestand der Bibliothek zu einem Thema zu erhalten, bietet es sich an, nach automatisierten oder teilautomatisierten Lösungen für die klassifikatorische Erschließung zu suchen. Bei der Suche stößt man schnell auf das Toolkit Annif. Annif ist eine Entwicklung der finnischen Nationalbibliothek, die als Open-Source-Software zur Verfügung steht und in modularer Architektur verschiedene Verfahren des maschinellen Lernens zur inhaltlichen Erschließung bereitstellt. Auch in Deutschland wurde Annif schon von Bibliotheken genutzt und weiterentwickelt; prominente Beispiele hierfür sind die Deutsche Nationalbibliothek (DNB) und die Bibliothek des ZBW − Leibniz-Informationszentrum Wirtschaft (siehe z. B. Mödden, 2020; Suominen et al., 2022, S. 273, 277).

Mithilfe von Annif können Computerprogramme trainiert werden, um Dokumenten jeweils Notationen eines Klassifikationssystems, wie z. B. der RVK, oder Schlagwörter aus

einem kontrollierten Vokabular, wie z. B. der Gemeinsamen Normdatei (GND), zuzuordnen. In dieser Masterarbeit soll untersucht werden, ob die Nutzung von Annif mit RVK-Notationen möglich ist und welche Vorarbeiten hierfür notwendig sind. Es soll bewertet werden, ob die Ergebnisse für einen Einsatz in Bibliotheken, welche die RVK nutzen, insofern ausreichend sind, als dass die von Annif vergebenen RVK-Notationen, ggf. nach Prüfung durch Fachpersonal, für die Erschließung übernommen werden können. Die Arbeit wird sich ausschließlich auf den Trainingsprozess und die Bewertung der Ergebnisse mithilfe einer Testkollektion konzentrieren. Es wird eine Kollektion von Dokumenten mit RVK-Notationen verwendet, die in Trainingsdaten und Testdaten aufgeteilt wird. Die Kollektion enthält Dokumente, die bereits intellektuell von Bibliotheksangestellten oder anderem Fachpersonal klassifikatorisch erschlossen und mit RVK-Notationen versehen wurden. Im Training soll Annif anhand von Beispielen, den so genannten Trainingsdaten, lernen, passende RVK-Notationen zu neu vorgelegten Dokumenten vorzuschlagen. Die Testdaten werden verwendet, um die darin bereits vorhandenen RVK-Notationen mit den von Annif vergebenen zu vergleichen. Die Ergebnisse sollen zeigen, ob ein praktischer Einsatz von Annif gerechtfertigt ist und welche Überlegungen im Vorfeld getätigt werden müssen, um Annif auf den Einsatz vorzubereiten. Dabei wird insbesondere betrachtet, welche Daten benötigt werden und wie sie aufbereitet werden müssen, damit Annif sie verarbeiten kann. Da mein Interesse den Bibliotheken im deutschsprachigen Raum gilt und es in Bezug auf Annif bisher wenig Untersuchungen zu Daten in deutscher Sprache gibt, wird in dieser Arbeit mit deutschsprachigen Texten für das Training und die Auswertungen gearbeitet.

2 Grundlagen

2.1 Maschinelles Lernen: Grundlagen und Begrifflichkeiten

Das maschinelle Lernen ist ein Teilbereich der Künstlichen Intelligenz. Als Teilgebiet der Informatik beschäftigt Künstliche Intelligenz sich mit der Frage, wie Maschinen Aufgaben verrichten können, für die normalerweise menschliche Intelligenz benötigt wird (siehe z. B. Ertel, 2021, S. 1). Beim maschinellen Lernen wird versucht, die Tatsache, dass Menschen aus Erfahrung lernen, auf Computerprogramme zu übertragen. Anders als bei statischer Programmierung, in der das Computerprogramm einen vorgegebenen Weg beschreitet und darüber ein Ergebnis berechnet, wird im maschinellen Lernen ein

Computerprogramm mit Daten versorgt, mithilfe derer es sein Verhalten anpassen soll. Die Daten stellen hierbei die Erfahrung dar, aus denen der Mensch sein Verhalten lernt (Frochte, 2020, S. 14).

Die Verfahren des maschinellen Lernens, die in Annif eingesetzt werden, lassen sich in die Unterkategorie des überwachten Lernens einordnen. Beim überwachten Lernen soll das Computerprogramm mithilfe von Beispieldaten Muster erkennen. Anders als beim unüberwachten Lernen sind hier die zu lernenden Zielwerte, denen das Programm neue Daten zuordnen soll, bereits bekannt. Die Beispieldaten bestehen aus Eingabe-Ausgabe-Paaren. Anhand dieser soll „eine Funktion bestimmt werden, die gegebene Eingabewerte auf bekannte Zielwerte abbildet." (Lanquillon, 2019, S. 96). Die Ausgaben oder Zielwerte werden als Labels bezeichnet. Im Zusammenhang dieser Masterarbeit ist eine RVK-Notation, also eine Systemstelle des Klassifikationssystems Regensburger Verbundklassifikation (RVK), ein Label. Labels werden jeweils so genannten Objekten zugeordnet, d.h. die RVK-Notation ST 300 (Informatik → Künstliche Intelligenz → Allgemeines), könnte z. B. dem Titeldatensatz für das Buch „Artificial Intelligence" von Elaine Rich von 1983 zugeordnet werden. Der Titeldatensatz wird in diesem Zusammenhang als Objekt bezeichnet. Die gelernte Funktion wird auch als Modell bezeichnet (Lanquillon, 2019, S. 96). Das Modell entsteht durch das Training. In diesem werden dem Computersystem Beispielpaare vorgelegt, aus denen es das Modell entwickelt oder „lernt". Die vorgelegten Eingabe-Ausgabe-Paare werden als Trainingsdaten und die Erstellung des Modells als Lernphase oder Trainingsphase bezeichnet (ebd., S. 91). Die so genannten Testdaten bestehen ebenfalls aus Eingabe-Ausgabe-Paaren. Sie werden nicht für das Training genutzt, sondern zurückgehalten, um das gelernte Modell zu evaluieren, indem dieses Vorschläge zu der Eingabe ausgibt und diese mit den schon vorhandenen Ausgaben verglichen werden. Je mehr Übereinstimmungen es zwischen den vorgeschlagenen Ausgaben und den bereits vorhandenen Zielwerten gibt, umso effizienter ist das Modell.

Für die Erstellung eines Modells sind gewisse Vorarbeiten notwendig. Zunächst müssen Trainings- und Testdaten ausgewählt und in eine geeignete Form gebracht werden. Für die Trainings und Tests mit Annif werde ich eine Auswahl an bibliografischen Daten zusammenstellen, also Daten, die eine Ressource eindeutig beschreiben. Bei den Ressourcen handelt es sich im Fall dieser Arbeit um monografische Ressourcen. In dieser Arbeit

wird der Begriff Dokument bzw. Objekt genutzt, um jeweils die ausgewählten bibliografischen Daten zu einer Ressource zu referenzieren.

Die Trainings- und Testdaten werden in Einzelteile, so genannte Terme oder Tokens, zerlegt. Dieser Prozess wird Tokenisierung genannt. Biemann et al. beschreiben Tokens als „die in einem grammatisch und orthographisch korrekten Text stehenden Zeichenketten bestehend aus Buchstaben und Bindestrich, die im Text durch Whitespace oder Satzzeichen getrennt sind“ (Biemann et al., 2022, S. 96). Die Autoren weisen darauf hin, dass dies nur eine gängige Vorgehensweise der Tokenisierung ist und es andere Möglichkeiten zur Zerteilung eines Texts in Tokens gibt (ebd.). Die so genannte Normalisierung wandelt Großbuchstaben in Kleinbuchstaben um. Beim Stemming werden Suffixe entfernt und die Wörter auf ihren vermeintlichen Stamm reduziert. Das Ziel liegt darin, Variationen zu reduzieren, sodass ein Algorithmus die Bedeutung verschiedener Wörter mit gleichem Stamm nicht mehrfach lernen muss (Suominen & Koskenniemi, 2022).

Ein Modell, das erstellt wird, um Objekte zu klassifizieren, bezeichnet man auch als Klassifikator. Auch das Toolkit Annif ist ein Klassifikator. Als Klassifikation wird das Teilgebiet des überwachten Lernens bezeichnet, bei dem einem Objekt ein oder mehrere Labels zugeordnet werden sollen. Es steht eine endliche Anzahl an vorher festgelegten Labels zur Verfügung, die in der Regel aus einem Klassifikationssytem oder einem festgelegten Vokabular aus Begriffen der natürlichen Sprache stammen.

2.2 Das Toolkit Annif

2.2.1 Übersicht

Bei dem Toolkit Annif handelt es sich um eine Software-Entwicklung der Nationalbibliothek Finnlands, die sich Verfahren maschinellen Lernens bedient, um Dokumente automatisiert zu klassifizieren. Ein erster Prototyp wurde 2017 in einem Experiment entwickelt (Suominen et al., 2022, S. 266). Es sollte ein Weg gefunden werden, aus den Daten des finnischen Discovery-Systems Finna Daten zur Inhaltsbeschreibung zu generieren. Statt der Eingabe eines Themas und der Ausgabe dazugehöriger Dokumente sollte hier die Eingabe ein Dokument sein und die Ausgabe eine Auswahl an Begriffen, die das Dokument bestmöglich beschreiben. So entstand der Name Annif. Er ergibt sich, wenn man Finna rückwärts liest. (Suominen, 2019, S. 3).

2018 entschied die Nationalbibliothek Finnlands, eine neue Version von Annif zu entwickeln und diese auf einer soliden technischen Grundlage und einer Festlegung von Prinzipien aufzubauen. Zu den Prinzipien gehören Multilingualität, die Möglichkeit, verschiedene kontrollierte Vokabulare und auch Klassifikationssysteme trainieren zu können, eine Auswahl an statistischen Verfahren und maschinellen Lernverfahren, eine Kommandozeilen- sowie auch eine Webanwendung und schließlich eine Software, die als Open-Source-Software zur Verfügung steht und mit der Gemeinschaft weiterentwickelt wird (Suominen et al., 2022, S. 266f.).

2.2.2 Nutzungsbeispiele

Als Werkzeug zur automatisierten oder teilautomatisierten Erschließung von Texten ist Annif für Bibliotheken und andere Institutionen, die Informationen aufbereiten und bereitstellen, interessant. In der Praxis gibt es bereits Institutionen, die Annif nutzen. Hier sollen einige Beispiele genannt werden. Als Entwicklerin nutzt die Nationalbibliothek Finnlands das Toolkit Annif für das von ihr betriebene Repositorium Varsta. Für hochgeladene Metadaten erhält das katalogisierende Bibliothekspersonal Vorschläge zur Sacherschließung mit Schlagwörtern über eine Schnittstelle zu Annif (Suominen et al., 2022, S. 276).

An der Universität von Jyväskylä in Finnland wurde bereits zu einem frühen Entwicklungsstadium mit dem Prototypen von Annif gearbeitet, um Master- und Doktorarbeiten teilautomatisiert zu erschließen. Hier erhalten Studierende beim Hochladen ihrer Abschlussarbeiten Vorschläge für Schlagwörter, die sie auswählen, aber auch ignorieren können (Suominen, 2019, S. 14ff.). Mit ihrer Arbeit hat die Universität von Jyväskylä zudem zur Evaluation von Annif und somit zur weiteren Entwicklung beigetragen.

Auch in Deutschland gibt es Institutionen, die mit Annif arbeiten, u. a. das ZBW − Leibniz-Informationszentrum Wirtschaft, das auch Beiträge zur Weiterentwicklung von Annif geleistet hat. Hier wird Annif für die Datenbank EconBiz genutzt. Die darin enthaltenen Dokumente werden mit Begriffen aus dem Standardthesaurus Wirtschaft (STW) inhaltlich erschlossen. Mit Annif wurde ein Modell trainiert, das Vorschläge zur Erschließung der Dokumente liefert. Über eine Schnittstelle können die Vorschläge an den Digitalen Assistenten des Gemeinsamen Bibliotheksverbundes (GBV), ein Erschließungstool, das hauptsächlich mit Ähnlichkeiten und Konkordanzen arbeitet, gesendet

werden (Beckmann et al., 2019, S. 9) und für die Erschließung von Ressourcen im Bestand des GBV genutzt werden.

In den bisherigen Beispielen wurde jeweils mit kontrollierten Vokabularen, also einer begrenzten Menge an Labels in natürlicher Sprache, die Objekte beschreiben, gearbeitet, jedoch nicht mit Notationen aus Klassifikationssystemen. Unter einer Notation versteht man eine Kombination von Buchstaben und/oder Zahlen, mit der ein Dokument inhaltlich beschrieben werden kann. Zur Vergabe von Notationen mithilfe von Annif gibt es in der Literatur wenige Berichte, obwohl hin und wieder auch spezifische Fragen zu Klassifikationssystemen und Annif in der Gruppe der Nutzenden gestellt werden[1]. Wie ich im Laufe der Arbeit noch beschreiben werde, gibt es bei der Nutzung von Annif mit Klassifikationssystemen zusätzliche Herausforderungen, die mit kontrollierten Vokabularen in natürlicher Sprache nicht in gleichem Maße beachtet werden müssen, weshalb ich vermute, dass Untersuchungen hierzu bisher seltener gemacht wurden. Man findet lediglich Literatur zu Unternehmungen an der Deutschen Nationalbibliothek DNB. Hier wird Annif als ein Modul der Erschließungsmaschine EMa genutzt, sowohl zur Schlagwortvergabe als auch zur Vergabe von Sachgruppen und Kurznotationen der Dewey Decimal Classification (DDC). Der Einsatz erfolgt für die Vergabe der Kurznotationen vollautomatisiert und die Vergabe von Sachgruppen der DDC für Printpublikationen aus den Reihen B (Neuerscheinungen außerhalb des Verlagsbuchhandels) und H (Hochschulschriften) erfolgt teilautomatisiert. DDC-Kurznotationen und DDC-Sachgruppen werden für Publikationen der Reihe O (Online-Publikationen) und Artikel vollautomatisiert mit Annif erschlossen (Deutsche Nationalbibliothek, 2023; Kähler, 2023).

2.2.3 Architektur

Annif besteht aus verschiedenen Modulen und so genannten Backends. Abbildung 2.1 stellt die Architektur von Annif dar. Da sich diese Arbeit auf das Training und dessen Auswertung konzentriert, werde ich mich nicht mit den REST APIs beschäftigen und nicht alle Backends nutzen. Es wird beispielhaft eine Auswahl evaluiert.

Im Konfigurationsmodul werden die so genannten Projekte definiert. Annif-Projekte werden dazu verwendet, das zu nutzende Verfahren festzulegen und Einstellungen wie z. B. die Sprache, das Werkzeug für die Vorverarbeitung und weitere Parameter zu konfigu-

[1] vgl. https://groups.google.com/g/annif-users/

rieren. Im Vocabulary Modul wird das Vokabular hinterlegt. Unter Vokabular versteht man hier sowohl kontrollierte Vokabulare, also Sammlungen von Bezeichnungen, wie z. B. aus einem Thesaurus oder der Gemeinsamen Normdatei GND, als auch Notationen aus Klassifikationssystemen, wie z. B. der RVK oder der DDC. Im Vokabular sind alle Klassen bzw. Themen enthalten, die für die automatisierte Klassifikation zur Verfügung stehen (The National Library of Finland, 2023c). Das Vokabular enthält somit die Menge an zur Verfügung stehenden Labels.

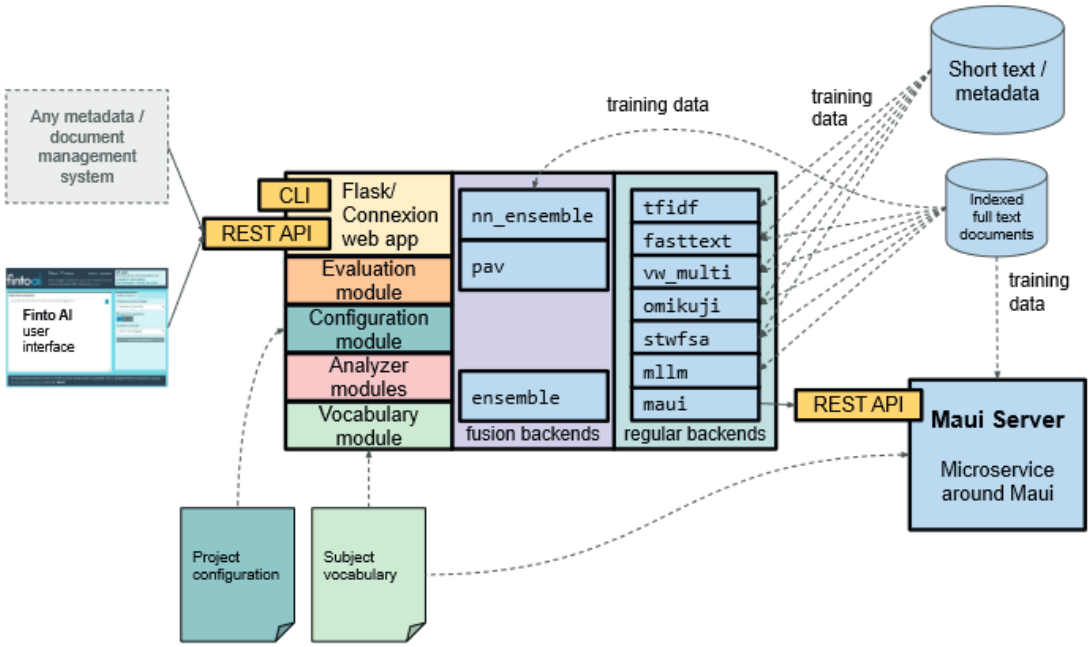

Abbildung 2.1: Annif Architektur, aus Suominen et al., 2022, S. 267

Für die Vorverarbeitung der Daten bietet Annif verschiedene so genannte Analyzer an. Die eingegebenen Daten werden zunächst normalisiert und tokenisiert. Standardmäßig eliminieren alle Analyzer in Annif zunächst Wörter, die aus weniger als drei Zeichen bestehen. Es besteht jedoch die Möglichkeit, die Mindestzeichenlänge anzupassen. Der Simple Analyzer in Annif teilt den Text in einzelne Wörter auf und wandelt diese in Kleinschreibung um. Der Snowball Analyzer basiert auf dem Natural Language Toolkit (NLTK), einer Plattform, mit der Python-Programme zur Verarbeitung von natürlichsprachlichen Texten erstellt werden können (NLTK Project, 2023). Zusätzlich zur

Normalisierung erfolgt mit dem Snowball Analyzer ein Stemming. Der Snowball Analyzer unterstützt mehrere Sprachen, darunter auch deutsch.

Annif bietet außerdem Analyzer an, die eine Lemmatisierung vornehmen. Bei der Lemmatisierung erfolgt die Reduzierung eines Wortes auf seinen Stamm nicht durch das einfache Entfernen von Suffixen, sondern es wird in Wörterbüchern nach der Grundform gesucht und Terme, die zu einer Bedeutung gehören, werden gruppiert. So können auch unregelmäßige Flexionsformen aufgespürt werden und beispielsweise eine Steigerungsform wie „besser“ dem Lemma „gut“ zugeordnet werden. Der im Annif Analyzer-Modul enthaltene Voikko-Analyzer wird nicht betrachtet, da dieser nur die finnische Sprache unterstützt. Der Analyzer Simplemma unterstützt 49 Sprachen, darunter auch deutsch. Er sucht nach Grundformen von flektierten Wörtern (Barbaresi, 2023). Der Spacy-Analyzer basiert auf dem spaCy NLP toolkit und unterstützt ebenfalls eine Vielzahl an Sprachen. Für die Nutzung in Annif ist die zusätzliche Installation eines vortrainierten Modells notwendig. Auch für die anderen Analyzer, die eine Lemmatisierung anbieten, müssen bei der Nutzung von Deutsch als Sprache zusätzliche Pakete installiert werden.

Mit der Auswahl des Backends wird das Verfahren festgelegt, mit dem Annif die Klassifizierung vornimmt. „Research in the field of automatic indexing can be broadly categorized into lexical approaches and associative approaches.” (Toepfer & Seifert, 2020, S. 170). Auch in Annif wird zwischen den so genannten lexikalischen und assoziativen Ansätzen unterschieden (Suominen et al., 2022, S. 268f.). Der lexikalische Ansatz sucht nach Übereinstimmungen von Wörtern im Dokument mit denen im Vokabular. Mithilfe von Heuristiken und Verfahren des maschinellen Lernens werden die Übereinstimmungen gefiltert und die vielversprechendsten Kandidaten ausgewählt (ebd., S. 268). Der lexikalische Ansatz eignet sich für Vokabulare, bei denen jedes Label eine eindeutige, einmalige sprachliche Bezeichnung hat wie z. B. die Schlagwörter in der gemeinsamen Normdatei (GND). Ein Nachteil des lexikalischen Ansatzes liegt darin, dass bei einem direkten Vergleich von Termen keine Synonyme oder ähnliche Konzepte berücksichtigt werden und es zu Fehlern kommen kann, wenn ein Wort mehrere Bedeutungen hat. Da die Benennungen der RVK-Notationen innerhalb des Klassifikationssystems mehrfach vorkommen und die Notationen auf der untersten Ebene, wo sie gebildet werden, oft mit generischen Bezeichnungen versehen sind, ist der lexikalische Ansatz für mein Vorhaben nicht geeignet, sodass ich Ansätze, die auf dem lexikalischen Verfahren beruhen, nicht testen werde.

Beim assoziativen Ansatz werden dem System manuell erschlossene Dokumente für das Training übergeben. Es sollen Wörter oder Ausdrücke gefunden werden, die bestimmte Konzepte beschreiben (Suominen et al., 2022, S. 269). So können auch verwandte Begriffe als Hinweis auf ein bestimmtes Thema erkannt werden. Das Backend TF-IDF ist ein Beispiel für einen assoziativen Ansatz. Weitere Beispiele für Backends mit assoziativem Ansatz in Annif sind Omikuji, Fasttext, und Vowpal Wabbit. Ich werde in dieser Masterarbeit die assoziativen Ansätze TF-IDF, Omikuji und Fasttext untersuchen. Sie werden in Kapitel 4 beschrieben.

Ein weiterer Ansatz, den Toepfer und Seifert beschreiben, ist der so genannte Fusion-Ansatz, der auch Ensemble genannt wird (2020, S. 172). Um ein Ensemble handelt es sich, wenn mehrere Verfahren miteinander kombiniert werden. Annif bietet drei Ensemble-Backends an, (Simple) Ensemble, NN-Ensemble und Pav. Im einfachen Ensemble wird aus den Vorschlägen aus mehreren Quellen, in der Regel aus einem assoziativen und einem lexikalischen Backend, ein gewichteter Durchschnitt berechnet. Die Ergebnisse hieraus werden als Vorschläge ausgegeben. Im Pav-Ensemble werden die Ergebnisse der gewählten Backends neu gewichtet, indem sie in Wahrscheinlichkeiten umgewandelt werden. Ähnliches geschieht im NN-Ensemble über ein neuronales Netzwerk. Das NN-Ensemble kann außerdem während der Nutzung weiter trainiert werden (Suominen et al., 2023).

Ein Ensemble wird oftmals genutzt, um die Vor- und Nachteile verschiedener Ansätze auszugleichen. In der Regel werden assoziative und lexikalische Ansätzen miteinander kombiniert oder Ansätze, die jeweils in unterschiedlichen Bereichen gut abschneiden. Es wird sich herausstellen, dass alle Ansätze, die in dieser Arbeit untersucht werden, in den gleichen Bereichen jeweils gute oder schlechte Ergebnisse erzielen, sodass die Kombination mehrerer genutzter Ansätze im Falle dieser Masterarbeit nicht zu einem Mehrwert führen würde, weshalb ich kein Ensemble detailliert untersuche.

2.3 Die Regensburger Verbundklassifikation (RVK)

> *„Ziel bibliothekarischer Klassifikationen ist die inhaltliche Erschließung von Dokumenten und die dadurch ermöglichte Wissensvermittlung, insbesondere im Hinblick auf monografische Literatur.“* (Lorenz, 2018, S. 2)

Die Vergabe von Notationen aus Klassifikationssystemen zählt im Bibliothekswesen zum Bereich der Sacherschließung, also der inhaltlichen Beschreibung und Erschließung von Ressourcen (Gantert & Hacker, 2016, S. 228). Die traditionelle Sacherschließung gliedert sich in die Schlagwortvergabe, die auch als verbale Sacherschließung bezeichnet wird, und die klassifikatorische Sacherschließung, bei der der Inhalt einer Ressource durch die Vergabe einer oder mehrerer Notationen beschrieben wird. Mithilfe klassifikatorischer Sacherschließung können Publikationen im Kontext ihres Fachgebiets nachgewiesen werden (ebd.).

Klassifikationssysteme, auch Klassifikationen genannt, sind bibliothekarische Ordnungssysteme, die als „klassisches Strukturelement des Bibliothekswesens gelten“ (Lorenz, 2018, S. 3). Viele Klassifikationssysteme waren ursprünglich Haussysteme und wurden somit innerhalb einer Bibliothek anhand des darin befindlichen Bestandes entwickelt und erprobt (ebd., S. 17). Die Regensburger Verbundklassifikation (RVK) ist ein Klassifikationssystem, das in den 1960er Jahren an der Universitätsbibliothek Regensburg entwickelt wurde und bis heute von vielen wissenschaftlichen Bibliotheken, insbesondere auch zur Signaturbildung und somit zur Aufstellung, genutzt wird (Häusler & Werr, 2018, S. 132). Die RVK gehört zu den so genannten Verbundklassifikationen, das bedeutet, sie wird kooperativ weiterentwickelt und optimiert (Lorenz, 2018, S. 17). Da die RVK in allen deutschen Bibliotheksverbünden genutzt wird, können oftmals Daten nachgenutzt werden, was die klassifikatorische Erschließung erleichtert und den Zeitaufwand für einzelne Bibliotheken verringert. Im deutschsprachigen Raum hat die RVK eine hohe Bedeutung.

Im Rahmen der vorliegenden Masterarbeit soll getestet werden, wie zuverlässig die Vergabe von RVK-Notationen durch Annif funktioniert. Die automatische Vergabe von Notationen soll an einem Klassifikationssystem getestet werden, das im deutschsprachigen Raum vielfach genutzt wird und somit typisch ist. So kann sich ein Eindruck zur Übertragbarkeit von Annif speziell auf diesen Sprachraum verschafft werden. Da es wichtig

ist, für das Training eine ausreichende Anzahl von Dokumenten zu finden und es für jede Notation einen Mindestanteil an zugehörigen Dokumenten geben soll, werde ich in dieser Masterarbeit lediglich eine Teilmenge an RVK-Notationen verwenden.

2.3.1 Aufbau

Die RVK ist ein monohierarchisches Klassifikationssystem, das bedeutet, dass jede Klasse nur eine direkt übergeordnete Klasse hat (Lorenz, 2018, S. 6). Sie besteht zum aktuellen Stand aus 34 Hauptgruppen. Unter Hauptgruppe verstehe ich jeweils die Menge der Notationen, die einem Buchstaben des Alphabets zugeordnet sind, der am Anfang jeder Notation steht. Notationen der RVK setzen sich alphanumerisch zusammen und bestehen aus zwei Großbuchstaben, gefolgt von drei bis fünf Ziffern. Einige Buchstaben sind in mehrere Fächer aufgeteilt. So beschreibt z. B. die Hauptgruppe C die Fächer Philosophie (CA bis CK) und Psychologie (CL bis CZ) und die Hauptgruppe S die Fächer Mathematik (SA bis SP) und Informatik (SQ bis SU). Der zweite Großbuchstabe benennt Untergruppen. Durch die Ziffernfolgen werden weitere Untergliederungsebenen abgebildet, die nicht hierarchisch sind (Häusler & Werr, 2018, S. 132).

Die Notation, die auch Systemstelle genannt wird, wird in der RVK auf der jeweils untersten Ebene des Klassifikationssytems gebildet. Abbildung 2.2 zeigt als Beispiel die Notation ST 134 für Algorithmen-, Komplexitätstheorie und ihre Eingliederung innerhalb der Hauptgruppe S. Je nach Fach oder Fächern sind die Hauptgruppen verschieden stark ausdifferenziert und haben unterschiedlich viele Ebenen.

Nach dem Stand von Januar 2023 enthält die RVK insgesamt 674.229 einzelne Notationen. Die Verteilung der Gesamtanzahl der Notationen auf die 34 Hauptgruppen ist sehr ungleichmäßig, was daran liegt, dass die RVK historisch gewachsen ist und sich ursprünglich an dem Bestand der Universitätsbibliothek Regensburg orientierte. Weitere Gründe liegen in Komplexität und Forschungsstand der einzelnen Fächer, die jeweils in den Hauptgruppen repräsentiert sind. Das Spektrum reicht von 881 Notationen für die Wirtschaftswissenschaften (Hauptgruppe Q) bis zu 119.412 Notationen für die Medizin, die sogar auf zwei Hauptgruppen (X und Y) verteilt wird.

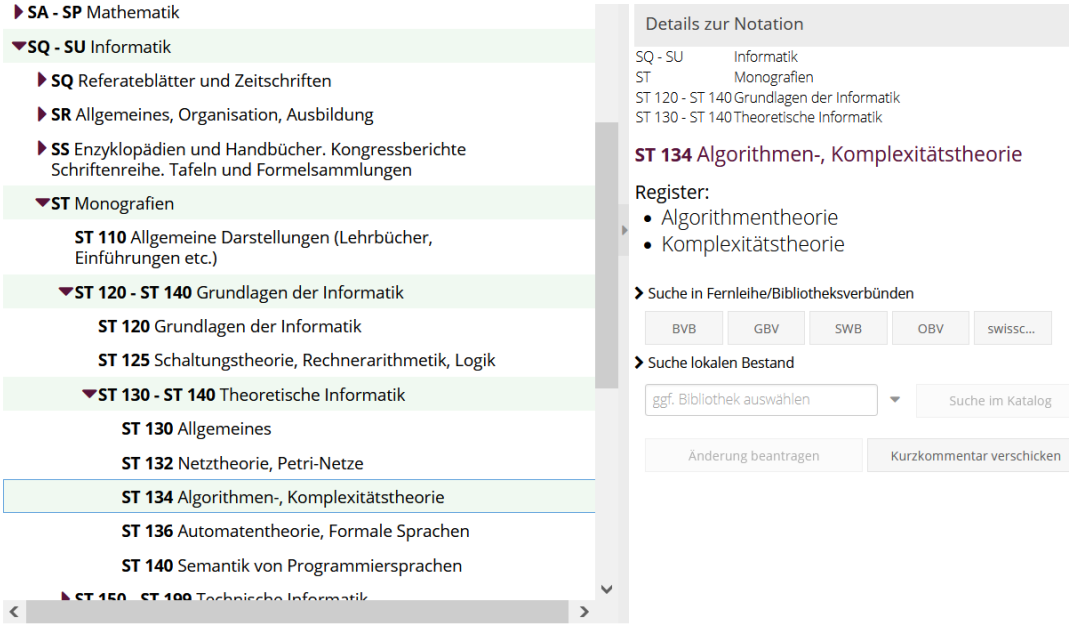

Abbildung 2.2: Notation der RVK innerhalb ihres Kontexts[2]

Bei der Überlegung, welche Teilmenge an Notationen Annif für das Training übergeben werden soll, muss berücksichtigt werden, dass ein Fach mit vielen Notationen vermutlich recht ausdifferenziert ist und Spezialgebiete abbildet, zu denen es weniger Dokumente gibt, die sich für ein Training eignen. Gleichzeitig soll die ausgewählte Teilmenge nicht zu homogen sein und die Dokumente sollen verschiedene Themen beschreiben.

2.3.2 Herausforderungen bei der Klassifikation mit Annif

Durch die bereits beschriebene Entstehung der RVK als Haussystematik und die historische Weiterentwicklung ergeben sich für die Nutzung mit Annif einige Herausforderungen, die sich im Rahmen dieser Arbeit nicht alle lösen lassen, die man jedoch bei der Bewertung der Ergebnisse im Hinterkopf behalten sollte. Zunächst ist zu beachten, dass die vorliegenden Daten in eine Form gebracht werden müssen, die von Annif verarbeitet werden kann. Da die Universität Regensburg die RVK in der Auszeichnungssprache XML zum Download anbietet, ist es mit geringem Aufwand möglich, die Daten in das von Annif geforderte Format zu bringen. Annif kann jedoch keine inhaltlichen Feinheiten erkennen. Auf den untersten Ebenen der RVK finden sich sowohl Notationen, die ein Dokument inhaltlich beschreiben können, als auch welche, die formale Aspekte als Beschreibungsmerkmal nutzen. So gibt es z. B. in der Hauptgruppe Z für „Technik“ auf der unter-

[2] https://rvk.uni-regensburg.de/regensburger-verbundklassifikation-online#notation/ST%20134

sten Ebene die Notation ZM 9060 für „Künstliche Intelligenz und Expertensysteme in der Produktion“, während in der Hauptgruppe V für „Chemie“ auf der unteren Ebene formale Gesichtspunkte für die Notation gewählt werden. Hier stehen die Notationen VC 6150 - VC 6159 für „Künstliche Intelligenz; Expertensysteme“ und teilen sich darunter noch nach Art der Publikation auf, also z. B. VC 6151 für „Lehrbücher“ aus diesem Bereich (siehe Regensburger Verbundklassifikation Online, 2023).

An dem genannten Beispiel ist außerdem zu erkennen, dass verschiedene Notationen ähnliche oder miteinander verwandte Themen beschreiben können und, je nach Fokus des Dokuments, eher zu einer oder zu einer anderen Hauptgruppe gehören können. Da ein Titeldatensatz mehrere Notationen erhalten kann, spielt diese Beobachtung in einer Bibliothek hauptsächlich für die Entscheidung über die Aufstellung einer physischen Ressource eine Rolle. Im Discovery-System werden alle für das jeweilige Dokument vergebenen Notationen angezeigt, sodass Nutzende einen Gesamtüberblick darüber erhalten, in welche Themenbereiche das Dokument gehört. Die Möglichkeit, auf diesem Weg eine Interdisziplinarität darzustellen, hat für Nutzende von Bibliotheken große Vorteile. Bei der Auswahl einer Teilmenge der RVK, wie sie in dieser Masterarbeit mit Annif durchgeführt wird, wird dies jedoch nicht in großem Maße zum Tragen kommen. Wenn man Annif für umfassendere Teilmengen der RVK oder für die gesamte RVK nutzen will, muss man bedenken und untersuchen, ob Annif in bestimmten Fällen den jeweils falschen Fokus erkennt. In Annif werden im Vorhinein die RVK-Notationen hinterlegt, aus denen Annif bei seiner Bewertung von nicht-klassifizierten Dokumenten auswählt. Kommen in den Trainingsdaten Notationen vor, die nicht zu dem in Annif hinterlegten, so genannten Vokabular, gehören, erkennt Annif das und gibt als Warnmeldung aus, dass die Notation nicht bekannt sei. Es besteht hier die Möglichkeit, bei der Transformation der Trainings- und Testdaten diejenigen RVK-Notationen herauszufiltern, die nicht Teil des Vokabulars sein werden. Sollte man in einem späteren Schritt jedoch das Vorhaben ausweiten und ein größeres Vokabular in Annif hinterlegen wollen, könnte man hierfür die bereits transformierten Daten als Teil einer neuen Menge an Trainingsdaten nutzen, wenn man die Notationen in den Daten belässt. Da das Training trotz der Warnmeldungen weitergeführt wird und die nicht-hinterlegten Notationen im weiteren Verlauf ignoriert werden, habe ich mich dazu entschieden, sie nicht herauszufiltern.

Eine RVK-Notation ist eindeutig durch ihre jeweilige Buchstaben-Ziffern-Folge, die im gesamten Klassifikationssystem nur einmal vorkommt, nicht jedoch durch ihre Benennung. So gibt es gerade bei Notationen, die ein Dokument auf der untersten Ebene formal beschreiben, mehrere Notationen mit der gleichen Benennung, z. B. gibt es innerhalb der RVK über 50 Notationen mit der Benennung „Dissertationen“. Annif wird bei den Vorschlägen jeweils die Notation selbst und die Benennung der untersten Ebene anzeigen. Für Fachpersonen in wissenschaftlichen Bibliotheken, die Erfahrung in der klassifikatorischen Erschließung haben, ist dies wenig herausfordernd, da man davon ausgehen kann, dass sie die jeweils von Annif vorgeschlagene Notation anhand der Buchstaben zumindest grob zuordnen können. Dennoch ist es unbefriedigend, dass Annif hier nicht die umliegende Struktur mitliefert. Sollte man Annif als Vorschlagswerkzeug nutzen und in einer ganzen Bibliothek einsetzen wollen, gilt es, hier eine Lösung zu finden. Eine einfache Möglichkeit besteht darin, die URIs für die einzelnen Notationen als Deeplink zur Stelle in der RVK Online zu konstruieren, wie ich es bei meiner Arbeit mit den Daten gemacht habe.

3 Vorarbeiten

3.1 Auswahl einer Teilmenge der RVK für den Test mit Annif

Für den praktischen Teil dieser Arbeit soll eine Teilmenge an RVK-Notationen ausgewählt werden, mit der ein Einblick gewonnen werden kann, ob eine Arbeit mit Annif für Bibliotheken in Deutschland sinnvoll und nützlich sein könnte. Damit die Ergebnisse aussagekräftig sind, werden ausreichend Trainings- und Testdaten benötigt. Hierfür sollen Metadaten genutzt werden, die zumindest zu einem angemessenen Teil auch Abstracts enthalten, da diese den Inhalt zusätzlich zum Titel weiter beschreiben und Annif so eine größere Grundlage hat als mit Metadaten, aus denen nur Titel und Titelzusatz extrahiert werden können. Entsprechende Metadaten werde ich über Schnittstellen aus Bibliothekskatalogen der deutschen Bibliotheksverbünde ziehen. Leider war es in der Katalogisierung lange Zeit nicht üblich, Abstracts mit in die Titeldatensätze aufzunehmen, sodass nicht alle Metadaten Abstracts enthalten werden. Da es zu jeder vergebenen Notation, die in der gewählten Teilmenge der RVK-Notationen enthalten ist, eine Mindestanzahl an Dokumenten geben sollte, soll keine Hauptgruppe oder Hauptgruppen ausgewählt werden, die besonders viele Notationen enthalten.

Um das Modell ausreichend trainieren zu können, werden pro vergebenem Label ausreichend Objekte benötigt. Zwar heißt es, man könne schon mit 50 Objekten pro Label angemessene Ergebnisse erzielen, jedoch sollte das nur als Mindestanforderung gelten (The National Library of Finland, 2021). Da ich ausreichend Trainings- und Testdaten zur Verfügung haben und eine Möglichkeit haben wollte, die Daten zu variieren und auf unterschiedlichen Wegen einzusetzen, habe ich mich dazu entschlossen, mich bei der Auswahl der RVK-Notationen auf eine kleine Hauptgruppe zu beschränken. Ich werde das Training und die Tests mit Dokumenten durchführen, die mit Notationen der Hauptgruppe S versehen sind. Die Hauptgruppe S beschreibt die Fächer Mathematik (SA bis SP) und Informatik (SQ bis SU). Sie enthält insgesamt 1.942 Notationen (s. Regensburger Verbundklassifikation Online, 2023).

3.2 Anpassung der RVK an die Bedürfnisse von Annif

> *"A subject vocabulary defines the subjects available for automated indexing or classification. Typically this will be a thesaurus, a classification or a list of subject headings. Annif doesn't care much about the internal structure of a subject vocabulary, it just needs to know the URIs and preferred labels (a.k.a. terms or descriptors) of each subject/class/concept."* (The National Library of Finland, 2023c)

Annif ist dafür ausgelegt, Objekte nach einem entsprechenden Training mit Labels zu versehen, die den Inhalt einer Ressource, die durch das Objekt repräsentiert wird, beschreiben sollen. Geeignet als Labels für die Sacherschließung mit Annif sind entweder Begriffe aus kontrollierten Vokabularen, wie z. B. der GND, oder aber Notationen aus Klassifikationssystemen, wie z. B. der RVK oder der DDC. Als Bezeichnung für beide Fälle nutzen die Entwickler:innen von Annif den Begriff subject vocabulary oder auch vocabulary, also Vokabular, was ich für diese Arbeit übernehme.

Das Vokabular muss vor dem Trainingsprozess in Annif hinterlegt werden. Mögliche Formate sind TSV, CSV oder SKOS/RDF. Ich habe mich für TSV (tab separated value) entschieden, da eine Transformation der Daten mithilfe der aus der RVK Online herunterladbaren Datei mit wenig Aufwand umsetzbar war. Im Format TSV werden jeweils URIs und Labels spezifiziert. Für jedes Label, also in meinem Fall für jede RVK-Notation, gibt es ein URI (Uniform Resource Identifier), also eine eindeutige, einzigartige Adresse bzw.

einen Identifikator für eine Ressource. Durch Tabulator getrennt folgt die Benennung der Notation und, wieder getrennt durch einen Tabulator, die Notation selbst, also die Buchstaben-Ziffern-Folge der jeweiligen Notation. Ich habe als URI die jeweilige URL der Notation in der RVK Online gewählt. Diese könnte man z. B. bei einem Einsatz mit der Schnittstelle Finto AI, der nicht Bestandteil dieser Arbeit ist, nutzen, um direkt per Mausklick eine Einordnung der vorgeschlagenen Notation innerhalb des Klassifikationssystems RVK zu erhalten.

Die RVK als wird von der Universitätsbibliothek Regensburg gemeinfrei als Download angeboten, sowohl formatiert in der Auszeichnungssprache XML als auch als MARCXML-Datei. Ich habe mir zur Weiterverarbeitung den XML-Abzug heruntergeladen, da dieser weniger Speicherplatz benötigt als die MARCXML-Datei und eine Verarbeitung mit Grundkenntnissen der Programmierung möglich ist. Abbildung 3.1 zeigt einen Ausschnitt aus der XML-Datei.

```
<node notation = "ST 600 - ST 690" benennung = "Datenverarbeitung in Anwendungsgebieten">
 <children>
  <node notation = "ST 600" benennung = "Mathematik, Statistik">
   <register>Anwendung
   </register>
   <register>Computeralgebra
   </register>
   <register>Datenverarbeitung
   </register>
   <register>Informatik
   </register>
   <register>Mathematik
   </register>
   <register>Statistik
   </register>
   <children>
    <node notation = "ST 601" benennung = "Einzelne Systeme (alphabetisch)">
     <register>Datenverarbeitung
     </register>
     <register>Programm
     </register>
     <register>Statistik
     </register>
    </node>
   </children>
  </node>
  <node notation = "ST 610" benennung = "Wirtschaftswissenschaften">
   <register>Anwendung
   </register>
   <register>Datenverarbeitung
   </register>
   <register>Wirtschaftswissenschaften
   </register>
  </node>
```

Abbildung 3.1: Auszug RVK als XML

Die Datei wurde mit Python eingelesen und die einzelnen Notationen mit ihren Benennungen ausgelesen. Da in dem XML-Abzug auch die übergeordneten Gruppen sowie die Hauptgruppen mit ihren Benennungen enthalten sind und diese jeweils keine eigenen Notationen haben, habe ich sie über Ausschluss herausgefiltert. Mit urllib.parse habe ich dann aus Notation und dem Grundteil „https://rvk.uni-regensburg.de/regensburger-verbundklassifikation-online#notation/“ das URI gebildet und den richtigen Umgang mit Leerzeichen gesichert. In Abbildung 3.2 wird der in Abbildung 3.1 gezeigte Ausschnitt in der von Annif erforderlichen Form dargestellt.

```
<https://rvk.uni-regensburg.de/regensburger-verbundklassifikation-online#notation/ST%20600>     Mathematik, Statistik   ST 600
<https://rvk.uni-regensburg.de/regensburger-verbundklassifikation-online#notation/ST%20601>     Einzelne Systeme (alphabetisch) ST 601
<https://rvk.uni-regensburg.de/regensburger-verbundklassifikation-online#notation/ST%20610>     Wirtschaftswissenschaften       ST 610
```

Abbildung 3.2: Auszug RVK, umgewandelt in von Annif gefordertes Format TSV

Wie das Vokabular in Annif hinterlegt wird, wird in Kapitel 3.7 behandelt.

3.3 Trainingsdaten und Testdaten

3.3.1 Auswahlkriterien

> *„Die Trainingsdaten müssen die Wirklichkeit repräsentieren, die sogenannte Grundwahrheit (Ground Truth) abbilden. Daher ist es wichtig vorab zu definieren, wie die Daten erhoben werden, wie groß die Mindestmenge an Daten zum Training ist und wie mit Ausreißern zu verfahren ist. Dies orientiert sich an der Schwierigkeit des Lernproblems sowie an der Bedeutsamkeit des Lernergebnisses bzw. dessen Folgen.“* (Mockenhaupt, 2021, S. 138)

Für das Training nutze ich Ausschnitte aus bibliografischen Titeldatensätzen aus Bibliothekskatalogen, die eine Ressource beschreiben. Es werden Metadaten von Ressourcen benötigt, die bereits klassifikatorisch erschlossen wurden. Da in den meisten der großen wissenschaftlichen Bibliotheken im deutschsprachigen Raum hauptsächlich Monografien klassifiziert werden, liegt der Fokus auf dieser Publikationsart. Die Daten sollen über eine oder mehrere Schnittstellen deutscher Bibliotheksverbünde beschafft werden. Ein Titeldatensatz enthält bibliografische Daten zu einer Ressource sowie in einigen Fällen zusätzliche Metadaten wie Inhaltsverzeichnisse oder Abstracts. Eine Auswahl an Daten, in diesem Fall Titel, Titelzusatz und, wenn vorhanden, Abstract oder Inhaltsverzeichnis, stellt in diesem Zusammenhang ein Objekt dar. Jedes genutzte Objekt muss deutsch-

sprachig sein und über mindestens eine RVK-Notation aus dem festgelegten Vokabular verfügen.

Was eine Mindestanzahl an Titeldatensätzen oder generell die Größe einer Trainingsdatenmenge angeht, gibt es in der Literatur wenig Anhaltspunkte, da dies sehr stark vom eigentlichen Vorhaben und den genutzten Verfahren abhängig ist. Grundsätzlich geht man zunächst davon aus, dass die Labels in der Menge der Trainings- und Testdaten gleichmäßig verteilt sein sollen, also dass es pro Label eine ähnliche Anzahl an Objekten gibt. Das kann im Falle der RVK kaum erreicht werden, da die Verteilung unausgewogen ist. Wenn man sich in Bibliotheken, die nach RVK aufstellen, umschaut, wird man schnell erkennen, dass auch hier nicht für jede RVK-Notation gleich viel Regalplatz reserviert ist. Die Entwickler:innen von Annif empfehlen ca. 50 Objekte pro Label, um akzeptable Ergebnisse zu erhalten, weisen aber auch daraufhin, dass eine größere Menge an Objekten ratsam sei, „it helps to have at least a few hundred“ (The National Library of Finland, 2021). Gleichzeitig wird darauf hingewiesen, dass die Vergabe von Labels subjektiv ist und verschiedene Personen oftmals unterschiedliche Labels vergeben würden, weshalb es auch nicht realistisch sei, von einer Maschine bessere Ergebnisse zu erwarten. Allerdings seien die unterschiedlichen Ergebnisse bei den intellektuell vergebenen Labels oftmals eine Frage der Perspektive, während Algorithmen dazu tendieren, Fehler zu machen, die ein Mensch niemals machen würde (ebd.). So kann eine Bibliothekarin gut erkennen, ob es sich bei einer Ressource über Logik thematisch eher um Mathematik oder Philosophie handelt. Wenn aber z. B. bestimmte Wörter gleich häufig oder selten im Text vorkommen, dann wäre eine falsche Zuordnung durch eine Maschine durchaus möglich. Bei der Evaluierung und insbesondere einem produktiven Einsatz von Annif in einer Bibliothek sollte man diese Tatsachen im Hinterkopf behalten und Möglichkeiten der regelmäßigen Überprüfung der Ergebnisse durch Fachpersonal bereitstellen.

Die Aufgabe, Objekte aus einem festgelegten Klassifikationssystem zu klassifizieren, wenn pro Objekt eine unbestimmte Anzahl an Labels vergeben werden können, wird im maschinellen Lernen als Multi-Label Classification bezeichnet (siehe z. B. Sebastiani, 2002, S. 3). Anders als bei einer binären Aufgabe, bei der einem Objekt genau ein Label von zwei zugeordnet wird und das Objekt damit automatisch nicht dem anderen Label angehört, gibt es diese Eindeutigkeit im Fall der Multi-Label Classification und somit auch bei der Vergabe von RVK-Notationen nicht. Von einem so genannten Extreme

Multilabel Classification-Problem (XMC Problem) spricht man, wenn Textdokumente mit feststehenden Labels verknüpft werden und die Menge der passenden Labels pro Dokument nicht beschränkt ist (Kähler, 2023). Zusätzlich ist die Menge der Labels, die vergeben werden könnten, in der Regel extrem groß und besteht oftmals aus mehreren Hunderttausenden von Labels (Babbar & Schölkopf, 2019, S. 1330). Einhergehend damit ist die so genannte Long-Tail-Problematik, die besagt, dass ein großer Teil der Labels sehr selten oder nie vorkommt. So erkennt man in einem Histogramm einen „langen Schwanz“ (englisch „long tail“). Abbildung 3.3 stellt die Häufigkeiten einzelner RVK-Notationen der Hauptgruppe S innerhalb einer Kollektion aus 108.753 Dokumenten dar, wobei nur ca. jede 15. Notation abgebildet ist und die Abbildung aus Platzgründen abgeschnitten wurde. Nach der letzten Notation folgen weitere 20 mit einmaligem Vorkommen. Insgesamt wurden in dieser Kollektion 802 Notationen mit Vorkommen zwischen 1 und 7.640 vergeben. 1.140 Notationen aus der Hauptgruppe S wurden in dieser Kollektion überhaupt nicht vergeben.

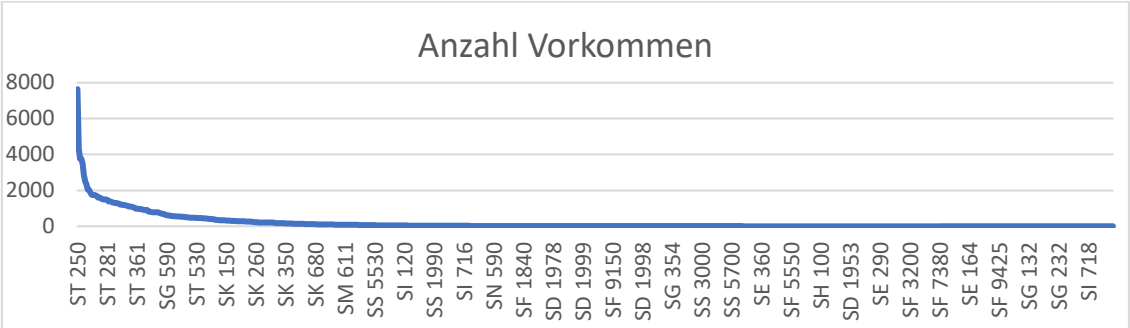

Abbildung 3.3: Veranschaulichung Long-Tail-Problematik

Es wird somit deutlich, dass eine vollkommen gleichmäßige Verteilung von Labels kaum zu erreichen ist, es sei denn, man schreibt ein Programm, das keine weiteren Objekte zu einem Label in eine Kollektion aufnimmt, sobald jeweils die Höchstanzahl an Vorkommen der am wenigsten vorkommenden RVK-Notation erreicht ist. Dazu müsste man vorher eine Mindestanzahl an Vorkommen festlegen, damit nicht jedes Label nur einmal vorkommt. In einer anderen Herangehensweise, die bei Ertel (2021, S. 249) beschrieben wird, werden Objekte mit Labels, die seltener vorkommen, dupliziert oder multipliziert, um auf die benötigte Menge an Objekten zu kommen. Da die Verteilung der Häufigkeiten die Realität widerspiegelt und man davon ausgehen kann, dass selten vorkommende Labels mit großer Wahrscheinlichkeit auch in Zukunft selten gewählt werden, wird die

ungleichmäßige Verteilung gelegentlich in Kauf genommen. Man konzentriert sich inzwischen darauf, Algorithmen zu entwickeln, die das Problem adressieren (siehe Kähler, 2023).

Bei der Beschaffung der Trainings- und Testdaten liegen die Schwierigkeiten in der ungleichen Verteilung der RVK-Notationen. Dessen muss man sich bewusst sein und schauen, ob die gefundenen Dokumente zumindest zu einem gewissen Maß eine gleichmäßige Verteilung bieten. Eine vollkommen gleichmäßige Verteilung der RVK-Notationen wird schwer erreichbar sein, muss aber auch nicht angestrebt werden. Idealerweise soll zumindest ein Teil der gefundenen Metadaten neben dem Titel auch ein Abstract oder ein Inhaltsverzeichnis enthalten, da diese die Ressource zusätzlich beschreiben.

Für das Training und die Tests werden bibliografische Daten wie Titel und Titelzusatz ausgelesen sowie als weitere beschreibende Daten Abstracts bzw. Inhaltsverzeichnisse, sofern vorhanden. Hierfür wird das MARC-Feld 520, in dem Angaben zum Inhalt, wie z. B. Abstracts oder Inhaltsverzeichnisse, erfasst werden, ausgelesen. Nicht zu allen Dokumenten findet man Titeldatensätze, die auch Abstracts und Inhaltsverzeichnisse enthalten. Es besteht die Möglichkeit, sich auf Datensätze zu konzentrieren, die nur Metadaten von elektronischen Ressourcen enthalten. Da die Metadaten von elektronischen Ressourcen oft von Verlagen geliefert werden, enthalten sie meistens zusätzliche Informationen wie Abstracts oder Inhaltsverzeichnisse. Man könnte also zunächst diesen Datenbestand durchsuchen und danach weitere Titeldatensätze aus dem Printbestand extrahieren. Problematisch ist hierbei jedoch, dass viele Monografien sowohl in Print als auch als E-Ressource vorliegen. Nach den Vorgaben des aktuellen Regelwerks Resource Description and Access (RDA) zur formalen Erschließung von Dokumenten werden Titel auf Manifestationsebene erschlossen, das bedeutet, es gibt für einen Titel jeweils eigene Titeldatensätze für die Printausgabe und die E-Ressource (siehe American Library Association et al., 2022). Bei der beschriebenen Vorgehensweise müssten also Dubletten in Kauf genommen und ein Umgang damit gefunden werden.

3.3.2 Testdaten

Aus der Gesamtheit der Trainings- und Testdaten wird eine große Menge der Objekte für das Training genutzt und eine kleinere Menge für die Tests zurückgehalten. Die bereits

klassifizierten Testdaten werden dem Modell unklassifiziert übergeben und die Vorschläge des Modells mit den intellektuell vergebenen Labels verglichen.

Die Testdaten werden als Goldstandard bezeichnet, da sie im Idealfall einem Standard entsprechen, nach dem jedes Dokument vollständig und korrekt klassifiziert wurde (vgl. Suominen et al., 2022, S. 270). Die Testdaten sollen einem Ideal entsprechen, gegen das das Modell gemessen wird. Da es kostenintensiv und zeitaufwändig ist, einen Goldstandard zu kreieren, werden für die Bewertung vorhandene, von Fachpersonal klassifizierte Dokumente für die Evaluation genutzt (ebd.). Für die Klassifizierung mit RVK-Notationen gibt es keinen allgemeingültigen Standard und die Entscheidung über die Vergabe von RVK-Notationen wird oft subjektiv getroffen und von verschiedenen Faktoren beeinflusst. Subjektivität ist hier nicht wertend zu verstehen, sondern beschreibt die Herangehensweise verschiedener Akteurinnen und Akteure, die beispielsweise auch von Überlegungen über die Zielgruppe beeinflusst wird. So ist es für eine Spezialbibliothek nicht unbedingt notwendig, durch die Angabe aller passenden RVK-Notationen aufzuzeigen, zu welchen anderen Sachgebieten ein Objekt noch zugehörig sein kann. Andererseits ist es für eine Universität, deren Fakultäten eine Vielzahl an Fächern abbilden, wichtig, dass bei der Klassifikation alle möglichen Zuordnungen berücksichtigt werden. Faktoren können auch in Verabredungen innerhalb der Institution oder in der generellen Verfügbarkeit von Fachpersonal liegen. Da ich für diese Masterarbeit die Trainings- und Testdaten aus Bibliothekskatalogen generieren werde, gehe ich davon aus, dass die Dokumente von ausgebildetem Bibliothekspersonal fachgerecht analysiert und klassifiziert wurden.

Für das maschinelle Lernen gibt es verschiedene Möglichkeiten, Testdaten zu erstellen bzw. Objekte aus einer Gesamtkollektion an Trainings- und Testdaten als Testdaten zu bestimmen. Für manche Projekte wird in Betracht gezogen, eigene Testdaten zu entwickeln, also Objekte speziell für die Tests mit Labels zu versehen. Dies wird allerdings nur selten gemacht, da der finanzielle sowie der personelle Aufwand hoch ist. In manchen Projekten wird das Training mit klassifizierte Titeldatensätzen durchgeführt und für die Tests Volltexte eingesetzt (siehe z. B. The National Library of Finland, 2021).

Eine weitere Möglichkeit für die Auswahl von Testdaten besteht darin, für die Tests aus der Gesamtkollektion diejenigen Objekte auszuwählen, die am aktuellsten sind, also im Falle von Texten diejenigen mit dem neuesten Erscheinungsdatum. Diese Methode wird

auch für Annif als eine Möglichkeit vorgeschlagen (ebd.). Dabei geht man davon aus, dass bei dem Einsatz eines automatisierten Verfahrens und im bibliothekarischen Tagesgeschäft hauptsächlich Neuerscheinungen erschlossen und das System somit mit den relevantesten Daten getestet werden sollte.

Ich habe zunächst eine Gesamtkollektion aus Metadaten zu 108.753 Ressourcen heruntergeladen und bearbeitet, um sie danach in Trainings- und Testdaten aufzuteilen. Die genaue Vorgehensweise hierzu ist in Kapitel 3.3.3 beschrieben. Durch die nachträgliche Aufteilung in Trainings- und Testdaten besteht die Möglichkeit, dass Testobjekte vereinzelt Labels enthalten, die im Training nicht vorkommen, was die Fehlerquote erhöhen kann.

Um das Potential für eben beschriebenen Fehler zu verringern, habe ich mich bei der Aufteilung in Trainings- und Testdaten für eine Methode entschieden, in der Trainings und Tests mehrfach durchgeführt und die Daten dafür jeweils unterschiedlich aufgeteilt werden. Über ein Zufallsprinzip wurde die Gesamtkollektion vier Mal jeweils unterschiedlich in 80 % Trainingsdaten und 20 % Testdaten aufgeteilt. Eine Aufteilung von 80 % Trainingsdaten und 20 % Testdaten entspricht der gängigen Praxis (siehe z. B. Géron, 2022, S. 56). Aus den Ergebnissen aller Tests werden die arithmetischen Mittelwerte gebildet und ausgewertet.

3.3.3 Zusammenstellung der Trainings- und Testdaten

Da die DNB nicht nach RVK klassifiziert, habe ich es ausgeschlossen, mit Metadaten aus den Katalogen der DNB zu arbeiten. Der K10plus ist eine Verbunddatenbank, die Metadaten zu den Beständen der Mitgliedsbibliotheken sowohl des GBV als auch des SWB enthält, da diese Verbünde vor einigen Jahren ihre Kataloge zusammengelegt haben und die Mitgliedsbibliotheken seit 2019 gemeinsam im K10plus katalogisieren. Die Datenbank enthält „Bibliotheksdaten aus zehn deutschen Bundesländern, der Stiftung Preußischer Kulturbesitz und aus weiteren Einrichtungen“ (Bibliotheksservice-Zentrum Baden-Württemberg, 2023). Da ich mir erhoffe, aus dem K10plus eine annehmbare Menge an Daten zu erhalten, werden die Metadaten für die Trainings- und Testdaten von dort beschafft. Der K10plus bietet eine SRU-Schnittstelle an, über die die benötigten Daten heruntergeladen werden können. SRU steht für „Search/Retrieve via URL“ und ist ein technischer Standard für die Abfrage von bibliografischen Daten. Ich werde die Daten aus der

Gesamtkollektion nach verschiedenen Kriterien unterschiedlich zusammenstellen und untersuchen, mit welchen Trainings- und Testdaten welche Werte besonders gut ausfallen und ob und wo es Unterschiede gibt. So kann erkannt werden, mit welchen Daten das Modell trainiert werden sollte, damit die Ergebnisse einen Einsatz von Annif rechtfertigen.

Für das Erstellen der Trainings- und Testdaten habe ich ein Python-Skript erstellt, um passende Daten aus dem K10plus auszulesen und aufzubereiten. Es wurde nach Titeldatensätzen gesucht, die deutschsprachig und mit mindestens einer RVK-Notation aus der Hauptgruppe S versehen sind. Da für die Arbeit mit Annif nur diejenigen Daten benötigt werden, die die jeweiligen Texte inhaltlich beschreiben und somit nicht die vollständigen Titeldatensätze benötigt werden, habe ich die Titeldatensätze zunächst vollständig ausgelesen und dann mithilfe des Python-Moduls Pymarc die Inhalte aus den Feldern für Titel, Titelzusatz und Abstract bzw. Inhaltsverzeichnis extrahiert. Im MARC-Feld 245 ist der Titel verzeichnet; der Haupttitel in Unterfeld a und der Titelzusatz in Unterfeld b. Gemäß RDA ist das Feld 245a ein Kernelement, also ein Pflichtfeld, das in jedem Datensatz vorkommt, sodass keine Vorkehrungen für einen Fall getroffen werden muss, in dem das Feld nicht belegt ist (American Library Association et al., 2022). Das Feld ist innerhalb eines Titeldatensatzes nicht wiederholbar, kann also nicht mehrfach vorkommen, d.h. auch hier muss keine Ausnahmebehandlung geschrieben werden. Titelzusätze kommen nicht in jedem Datensatz vor, sodass sie gemäß dem Skript nur bei Vorkommen in die Gesamtkollektion von Trainings- und Testdaten aufgenommen werden konnten. Da mich zunächst Titeldatensätze interessiert haben, in denen auch ein Abstract oder Inhaltsverzeichnis vorhanden war, wurden diese Daten aus dem Feld 520a extrahiert und der Datensatz nur aufgenommen, wenn das Feld vergeben war, ansonsten wurde der Datensatz verworfen. Da die Inhalte aus Titel, Titelzusatz und Abstract bzw. Inhaltsverzeichnis während der Vorverarbeitung durch den Analyzer tokenisiert werden, spielt es keine Rolle, in welcher Form die Feldinhalte voneinander getrennt sind, solange keine Tabulatoren genutzt werden, die laut der von Annif vorgegebenen Struktur für die Trennung von Titeldaten und RVK-URI verwendet werden müssen. Auszüge aus einem Titeldatensatz im Schema MARCXML sind als Beispiel in Abbildung 3.4 dargestellt. Auslassungen wurden von mir durch *[...]* gekennzeichnet.

```
[...]
<datafield tag="084" ind1=" " ind2=" ">
<subfield code="a">EDV 90</subfield>
<subfield code="2">sfb</subfield>
</datafield>
<datafield tag="084" ind1=" " ind2=" ">
<subfield code="a">ST 250</subfield>
<subfield code="2">rvk</subfield>
<subfield code="0">(DE-625)rvk/143626:</subfield>
</datafield>
<datafield tag="084" ind1=" " ind2=" ">
<subfield code="a">ST 301</subfield>
<subfield code="2">rvk</subfield>
<subfield code="0">(DE-625)rvk/143651:</subfield>
</datafield>
<datafield tag="084" ind1=" " ind2=" ">
<subfield code="a">Programmier- und Skriptsprachen</subfield>
<subfield code="2">bicssc</subfield>
</datafield>
<datafield tag="084" ind1=" " ind2=" ">
<subfield code="a">54.72</subfield>
<subfield code="2">bkl</subfield>
[...]
<datafield tag="245" ind1="1" ind2="0">
<subfield code="a">Neuronale Netze programmieren mit Python</subfield>
<subfield code="c">Joachim Steinwendner, Roland Schwaiger</subfield>
</datafield>
[...]
<datafield tag="520" ind1=" " ind2=" ">
<subfield code="a">
Neuronale Netze stehen im Mittelpunkt, wenn es um Künstliche Intelligenz und Machine Learning geht. Sie revolutionieren Bild-
und Spracherkennung, Spiele-KIs und vieles mehr. Zum Glück lassen sich die genialen Ideen dahinter einfach erklären.
Um sie zu verstehen und einzusetzen, programmieren Sie verschiedene Netztypen selbst nach! Und zwar in Python, der Hauptsprache der KI-Welt.
Sie werden sich dabei mit Mathematik und Programmierung befassen, brauchen aber keine konkreten Vorkenntnisse. (Verlagstext)
</subfield>
</datafield>
[...]
```

Abbildung 3.4: Auszug Titeldatensatz in MARCXML, aus SRU-Schnittstelle des K10plus

Für die RVK-Notationen wird in den Trainingsdaten das gleiche URI vergeben wie in dem in Annif hinterlegten Vokabular. Da die Suchanfrage so gestellt war, dass nur Datensätze mit mindestens einer RVK-Notation der von mir betrachteten Hauptgruppe S ausgegeben wurden, mussten keine Datensätze ohne RVK-Notation übersprungen werden. Das MARC-Feld 084a enthält Notationen aus verschiedenen Klassifikationssystemen. Das jeweilige Klassifikationssystem ist in Unterfeld 2 angegeben, sodass hierüber die Notationen, die der RVK angehören, ausgelesen und alle anderen ignoriert werden konnten. Die angegebenen RVK-Notationen wurden mithilfe des Skripts in das URI eingebettet. Die erstellten URIs wurden nach Vorgabe von Annif durch einen Tabulator vom Rest der extrahierten Daten des jeweiligen Titeldatensatzes getrennt. Mehrere URIs zu einem Objekt wurden voneinander durch ein Leerzeichen getrennt. Abbildung 3.5 zeigt den Titeldatensatz aus Abbildung 3.4 in konvertierter Form mit den beiden RVK-Notationen ST 250 und ST 301.

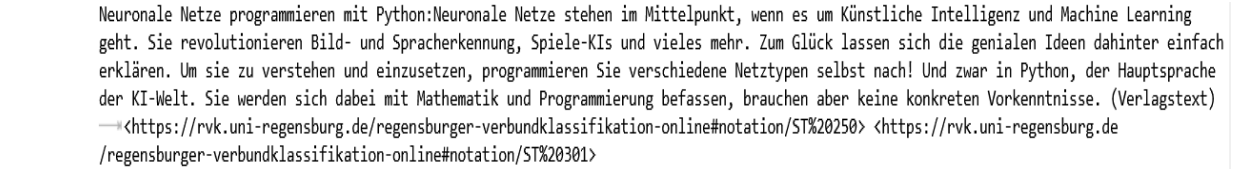

Abbildung 3.5: Objekt mit Labels, vorbereitet für die Verarbeitung in Annif

Damit die Trainings und Tests wie beschrieben variiert werden können, habe ich das Skript später auf verschiedene Weisen abgewandelt und eine Datenmenge erstellt, in der ich alle vorhandenen Objekte belassen habe, also sowohl welche mit zusätzlichen Angaben zum Inhalt als auch diejenigen ohne Abstract oder Inhaltsverzeichnis. Außerdem habe ich eine Datenmenge zusammengestellt, in der ich nur diejenigen Objekte behalten habe, in denen Labels enthalten waren, die mindestens 50-mal in der Gesamtheit der Datenmenge vorkommen, um herauszufinden, wie sich das Modell ohne einen „Long Tail“ verhält. Jedes zu testende Verfahren wird mit diesen drei unterschiedlich zusammengesetzten Daten trainiert und getestet. Die Tabelle in Abbildung 3.6 zeigt die verschiedenen Zusammensetzungen der Trainings- und Testdaten.

Kurzbezeichnung	**Beschreibung**	**Gesamtkorpus**	**Trainingsdaten**	**Testdaten**
Abstracts	Titeldatensätze, die Abstract oder Inhaltsverzeichnis enthalten	52.239 Objekte	41.791 Objekte	10448 Objekte
Abstracts ohne Long Tail	Titeldatensätze, bei denen jedes Label mindestens 50 Titeldatensätzen zugeordnet ist	48.282 Objekte	38.625 Objekte	9.657Objekte
Gemischt	Titeldatensätze, die nur zum Teil Abstract oder Inhaltsverzeichnis enthalten	108.753 Objekte	87.002 Objekte	21.751 Objekte
Volltexte	Monografische Ressourcen	498 Volltexte	398 Volltexte	100 Volltexte

Abbildung 3.6: Übersichtstabelle Trainings- und Testdaten

3.3.4 Sampling mit Volltexten

Es wird zusätzlich untersucht, ob die Effektivität des Verfahrens durch Einbeziehung von Volltexten erhöht werden kann. Da ein großer Teil an Volltexten urheberrechtlich geschützt ist und der Zugriff aus diesem Grund nicht auf eine große Menge an Volltexten möglich war, habe ich ein Sampling mit einer kleinen Auswahl an Labels durchgeführt. Ich habe 14 verschiedene Labels aus der RVK-Gruppe S (Informatik und Mathematik), von denen ich aus vorherigen Berechnungen wusste, dass sie häufig vorkommen, ausgewählt und jeweils ca. 50 Volltexte pro Label heruntergeladen. In der Nacharbeit musste ich einige Volltexte wieder entfernen, da eine Textextraktion nicht möglich war, sodass für jedes gewählte Label ca. 41 Volltexte übrig blieben. Durch meine Vorgehensweise war außerdem pro Volltext jeweils nur ein Label vorhanden, sodass nicht untersucht werden konnte, wie effektiv das Verfahren mehrere Labels pro Volltext vorhersagen würde.

Die heruntergeladenen Volltexte hatten alle das Format PDF (Portable Document Format). Damit Annif sie verarbeiten kann, wurden sie über ein Skript in ein reines Textformat umgewandelt. Korrespondierend zu den Textdateien benötigt Annif zu jeder Datei jeweils eine weitere, so genannte Schlüsseldatei, die das passende Label in Form seines URIs enthält. Hierzu habe ich Dateien mit der Dateiendung `.tsv` erstellt, in denen das jeweils passende URI und die RVK-Notation durch Tabulator voneinander getrennt enthalten sind. In einem gemeinsamen Verzeichnis waren daraufhin Textdateien im Format `document[n].txt` und die jeweils dazugehörigen Schlüsseldateien im Format `document[n].tsv` abgelegt. Wie bei den Untersuchungen zuvor, habe ich jeweils vier Durchläufe absolviert, in denen die Daten nach Zufallsprinzip unterschiedlich in Trainings- und Testdaten aufgeteilt waren.

3.4 Bewertungsmetriken

Zur Bewertung von Projekten stellt Annif die Berechnung bestimmter Metriken, die im maschinellen Lernen üblicherweise genutzt werden, zur Verfügung. Mit den gewonnenen Werten kann ein Grad der Übereinstimmung zwischen den automatisiert von Annif und den intellektuell von Menschen vergebenen Labels bestimmt werden. Das bedeutet, je öfter Annif Labels vorschlägt, die auch in den Testdaten von sacherschließenden Personen vergeben wurden, umso höher ist der Wert. Werte zur Bestimmung der Effektivität von

Lernverfahren werden im maschinellen Lernen als Score-Werte bezeichnet. Der errechnete Score-Wert liegt zwischen 0 und 1, wobei 1 die vollständige Übereinstimmung der Vorschläge durch Annif mit den intellektuell vergebenen Labels darstellt. In Annif implementierte Berechnungen existieren u.a. für Precision, Recall und den F1-Score sowie den NDCG (Normalized Discounted Cumulative Gain).

Die Precision ist der Anteil von richtig vorhergesagten Labels an allen vorhergesagten Labels. Richtig vorhergesagte Labels werden als True Positives (TP) bezeichnet und sind in meinem Fall diejenigen Labels, die Annif vorhergesagt hat und die in den Testdaten für die gleichen Objekte manuell vergeben wurden. False Positives (FP) sind diejenigen Labels, die Annif vorgeschlagen hat, obwohl diese nicht von Menschen intellektuell in den Testdaten für die jeweiligen Objekte vergeben wurden. Die Formel für Precision lautet `TP/(TP+FP)` (siehe z. B. Grandini et al., 2020, S. 2). Eine hohe Precision ist dann wichtig, wenn falsche Vorschläge vermieden werden sollen, man aber ggf. akzeptieren kann, dass nicht alle zutreffenden Labels vorgeschlagen werden.

Der Recall wird berechnet, indem man False Negatives (FN) betrachtet, das sind diejenigen Labels, die in den Testdaten vergeben wurden, die aber nicht als Vorschlag von dem Modell genannt wurden. Recall berechnet den Anteil der True Positives an der Summe der True Positives und False Negatives. Die Formel für Recall lautet `TP/(TP+FN)` (ebd., S. 3). Ein hoher Recall ist wichtig für Situationen, in denen nach einem vollständigen Ergebnis gestrebt wird, in dem nach Möglichkeit alle zutreffenden Labels vorgeschlagen werden. Da False Positives bei Recall unerheblich sind, muss man beachten, dass bei einem hohen Recall auch falsche Labels in den Vorschlägen enthalten sein können. Abbildung 3.7 zeigt eine grafische Darstellung für die Berechnungen von Precision und Recall.

Um die Werte von Precision und Recall miteinander zu kombinieren, nutzt man den F1-Score. Hierbei wird der harmonische Mittelwert von Precision und Recall berechnet. Dies geschieht mit der Formel `2×(Precision×Recall)/(Precision+Recall)` (Shmueli, 2020). Annif berechnet den F1-Score auf mehrere Weisen, zum einen als arithmetischen Mittelwert aus den F1-Scores für jedes Dokument (F1_score_doc_avg). Außerdem wird der F1-Score als arithmetischer Mittelwert der F1-Scores für jede im Vokabular enthaltene RVK-Notation (F1_score_subj_avg) berechnet. Diese Werte fallen

erwartungsgemäß niedrig aus, da eine beträchtliche Anzahl an Notationen nur im Vokabular vorkommen, aber in den Trainings- und Testdaten selbst gar nicht vergeben wurden.

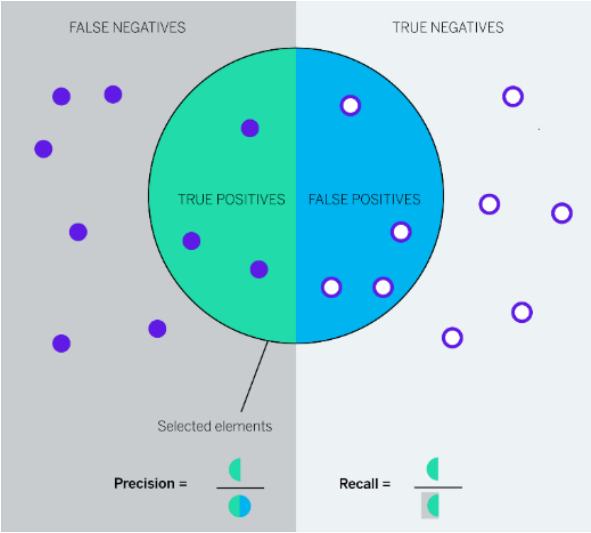

Abbildung 3.7: Darstellung Precision und Recall[3]

Annif berechnet außerdem Werte für Precision, Recall und F1 mit einer Gewichtung der Labels (F1_score_weighted_subj_avg) und außerdem Werte für Precision bei ausschließlicher Betrachtung eines, 3 oder 5 vorgeschlagener Labels (Precision@).

Der Normalized Discounted Cumulative Gain (NDCG) ist ein Evaluationsmaß, das auf einem Ranking beruht. Das bedeutet, dass die Reihenfolge, in der die Labels vorgeschlagen werden, eine Rolle spielt. Annif berechnet für jeden Vorschlag einen Konfidenzwert (suggest score), der auf einem Ranking beruht[4]. Die Vorschläge werden in der Reihenfolge des Rankings angezeigt. Der NDCG-Wert steigert sich, wenn Annifs erster Vorschlag eine Übereinstimmung mit einem intellektuell vergebenen Label hat (The National Library of Finland, 2021).

Es gilt zu überlegen, welche Bewertung bei einer automatisierten Vergabe von RVK-Notationen am sinnvollsten erscheint. Ein hoher Wert für Precision hat einen geringen Anteil an False Positives, was insbesondere für Szenarien wünschenswert ist, bei denen eine Vergabe vollautomatisiert erfolgt, also keine intellektuelle Bewertung der Vorschläge durch Fachpersonal erfolgen soll und die RVK-Notationen direkt in ein Bibliotheksmanagementsystem überführt werden. Unzutreffende Labels wären hier ungünstig, während

[3] https://www.qualtrics.com/experience-management/research/text-analysis/
[4] siehe https://groups.google.com/g/annif-users/c/98XevWGZY3w

das Fehlen einer zutreffenden RVK-Notation aus bibliothekarischer Sicht eventuell verkraftbar wäre. Bei einer teilautomatisierten Vergabe von Notationen, bei der das System Vorschläge abgibt, die von Fachpersonal bewertet werden und z. B. nach Vorgaben oder Praktiken der eigenen Bibliothek für die Erschließung ausgewählt werden, scheint Recall wichtiger als Precision. Die Personen, die die Ergebnisse prüfen, erhalten dann viele richtige Vorschläge und können diejenigen auswählen, die nach ihrer professionellen Einschätzung am passendsten sind. Jedoch ist bei einer hohen Anzahl an False Positives abzuwägen, inwieweit die Vorteile einer teilweisen Automatisierung den Aufwand, den die Erschließenden weiterhin hätten, rechtfertigt. Häufig wird der F1-Score zur Beurteilung herangezogen, da dieser sowohl Precision als auch Recall betrachtet und daraus ein harmonisches Mittel berechnet. In den Auswertungen werde ich mich in dieser Arbeit mit den Werten von Precision und Recall befassen und außerdem die Ergebnisse von NDCG und F1-Score betrachten.

3.5 Systemvoraussetzungen und Installation

Als Betriebssystem für die Entwicklung von Annif wird Ubuntu 18.04 und 20.04 verwendet (The National Library of Finland, 2022b). Für meine Masterarbeit wurde mir von der TH Wildau ein virtueller Server mit dem Betriebssystem Ubuntu 22.04.2 zur Verfügung gestellt. Grundvoraussetzung für Annif ist die Installation eines Python 3.7 oder höher. Ich habe Python 3.10.6 installiert. Für die Installation von Annif folgt man den Schritten, die im Annif Tutorial im Bereich Installation aufgeführt sind (Suominen et al., 2023). Zum Testen, ob die Grundinstallation erfolgreich war, kann man in der Kommandozeile den Befehl `annif` eingeben. Wenn die Installation funktioniert hat, erhält man eine Liste von möglichen Optionen und Kommandos (siehe Abbildung 3.8).

```
annif@annif:~$ annif
Usage: annif [OPTIONS] COMMAND [ARGS]...

Options:
  -e, --env-file FILE   Load environment variables from this file. python-
                        dotenv must be installed.
  -A, --app IMPORT      The Flask application or factory function to load, in
                        the form 'module:name'. Module can be a dotted import
                        or file path. Name is not required if it is 'app',
                        'application', 'create_app', or 'make_app', and can be
                        'name(args)' to pass arguments.
  --debug / --no-debug  Set debug mode.
  --version             Show the version and exit.
  --help                Show this message and exit.

Commands:
  clear          Initialize the project to its original, untrained state.
  eval           Suggest subjects for documents and evaluate the results...
  hyperopt       Optimize the hyperparameters of a project using...
  index          Index a directory with documents, suggesting subjects...
  learn          Further train an existing project on a collection of...
  list-projects  List available projects.
  list-vocabs    List available vocabularies.
  load-vocab     Load a vocabulary from a subject file.
  optimize       Suggest subjects for documents, testing multiple limits...
  routes         Show the routes for the app.
  run            Run a development server.
  shell          Run a shell in the app context.
  show-project   Show information about a project.
  suggest        Suggest subjects for a single document from standard input.
  train          Train a project on a collection of documents.
annif@annif:~$
```

Abbildung 3.8: Annif Optionen und Kommandos

3.6 Projekte erstellen

Um das Training durchzuführen, müssen zunächst ein Projekt angelegt und das Vokabular hinterlegt werden. Ein Projekt wird standardmäßig in einer Textdatei mit dem Namen `projects.cfg` definiert. Die Datei für ein Projekt kann wie folgt aussehen:

```
[rvk_s-tfidf-ger.1]
name=RVK_S TFIDF project
language=german
backend=tfidf
vocab=rvk_s
analyzer=snowball(german)
```

Abbildung 3.9: Datei projects.cfg in Annif

Es wird jeweils eine eindeutige, frei wählbare ID für jedes Projekt vergeben, in diesem Fall `rvk_s-tfidf-ger.1`. Außerdem wird in `name` ein Name festgelegt, die Sprache in `language` definiert und in `vocab` das Vokabular angegeben. Der zu nutzende Analyzer für die Vorverarbeitung wird unter `analyzer` ausgewählt. Ein einmalig hinterlegtes Vokabular kann auch in weiteren Projekten genutzt werden. In der Datei `projects.cfg` können mehrere parallele Projekte definiert werden. Je nach Projekt können hier weitere Parameter festgelegt werden. Um sich die Projekte anzeigen zu lassen, nutzt man den Befehl `annif list-projects` und bekommt angezeigt,

welche Projekte vorhanden sind und ob für das jeweilige Projekt schon ein Training absolviert wurde.

```
annif@annif:~$ annif list-projects
Project ID               Project Name                                   Language  Trained
-------------------------------------------------------------------------------------------
rvk_s-tfidf-ger.1        RVK_S TFIDF project                            german    True
```

Abbildung 3.10: Angelegtes Projekt in Annif

3.7 Hinterlegen des Vokabulars

Das Vokabular, also eine Liste der Labels, die von Annif vergeben werden sollen, wird in Annif mit dem Befehl `annif load-vocab` hinterlegt. Zusätzlich werden Bezeichnung und der Pfad zu der Datei, in der das Vokabular im geforderten Format hinterlegt ist, angegeben. Annif verarbeitet die Dateien und erstellt daraus die Dateien `subjects.ttl` und `subjects.csv`, die in einem gesonderten Ordner abgelegt werden.

4 Trainings und Auswertungen

4.1 TF-IDF

Als erstes Backend habe ich das Backend TF-IDF getestet. TF steht für Term Frequency und IDF für Inverse Document Frequency. Bei TF-IDF handelt es sich um eine Gewichtungsfunktion für Objekte, bei der davon ausgegangen wird, dass Wörter bzw. Terme, die innerhalb eines Dokuments häufig vorkommen, von hoher Bedeutung für das Thema des Dokuments sind. Die Häufigkeit des Vorkommens eines Terms innerhalb eines Dokuments nennt man Term Frequency. Das Vorkommen eines Terms innerhalb einer Kollektion von Dokumenten wird Document Frequency genannt. Es wird angenommen, dass ein Term, der innerhalb einer ganzen Kollektion selten vorkommt, für ein einzelnes Dokument, in dem er häufig vorkommt, eine hohe Relevanz hat, weswegen man die Document Frequency invertiert und die Inverse Document Frequency bestimmt. Terme, die in der Gesamtheit der Trainingsdaten häufig vorkommen, werden also geringer gewichtet als diejenigen, die insgesamt seltener vorkommen, aber häufig in Verbindung mit einem bestimmten Dokument auftauchen (The National Library of Finland, 2023b). Für die Vorverarbeitung werde ich im Projekt TF-IDF zunächst den Snowball Analyzer nutzen, bei dem keine weiteren Einstellungen notwendig sind.

4.1.1 Titel mit Abstracts als Trainingsdaten

Für die ersten Trainings sollen Titeldatensätze mit Abstracts genutzt werden. Ich habe auf dem im letzten Kapitel beschriebenen Weg mithilfe eines Python-Skripts eine .tsv-Datei mit insgesamt 52.239 Objekten und dazugehörigen Labels erstellt. Die Hauptgruppe S der RVK enthält insgesamt 1.942 Notationen, von denen in dieser Datei jedoch nur 363 vergeben wurden. Durchschnittlich sind das ca. 143 Objekte pro Label. Die Verteilung ist jedoch nicht so gleichmäßig. Bei näherer Betrachtung ist festzustellen, dass nur 123 Labels in jeweils mehr als 50 Objekten vergeben wurden. Meine Vermutung ist, dass die selten vergebenen Notationen von Annif vermutlich gar nicht oder nur sehr selten vorgeschlagen werden. Ich werde sie im ersten Durchlauf im Korpus belassen und in einem späteren Durchlauf prüfen, ob die Ergebnisse mit einem Training ohne den „Long Tail“ verbessert werden können.

Für das Backend TF-IDF müssen in der Konfiguration keine weiteren Einstellungen vorgenommen werden als diejenigen, die ich im Kapitel 3.6 beschrieben habe. Für ein Training gibt man als Befehl `annif train` mit der Projekt-ID und dem Pfad zu den Trainingsdaten ein. Wie erwartet sind während des Trainings Warnhinweise erschienen für jedes URI, das nicht im Vokabular enthalten ist. Zum Testen, ob das Training funktioniert hat und Annif Vorschläge für Labels ausgibt, kann man zunächst mit dem Befehl `echo` einen beliebigen Satz oder eine Reihe von Wörtern eingeben und getrennt mit einer Pipe den Befehl `annif suggest` und die Projekt-ID eingeben. Als Ausgabe bekommt man eine Liste mit RVK-URIs aus dem Vokabular und deren Bezeichnung sowie dazu ein von Annif berechneten Konfidenzwert, den suggest score.

Für eine Evaluierung nutzt man den Befehl `annif eval` mit der Projekt-ID und dem Pfad zu den Testdaten. Mit der Option `--metrics-file` und der Angabe eines Pfads und Dateinamens kann man festlegen, dass eine Auswertung mit allen Bewertungsmetriken gespeichert wird. Mit der Option `--results-file` und Pfad- und Dateiangabe speichert Annif eine Übersicht über die Werte für jedes einzelne Label, das im Vokabular vorhanden ist. Nachdem ich die erste Evaluierung mithilfe der Testdaten durchgeführt habe und die berechneten Bewertungswerte gesehen habe, wurde deutlich, dass sehr viele False Positives angezeigt wurden, was daran liegt, dass Annif standardmäßig 10 Labels für ein Dokument vorschlägt. Meiner Erfahrung nach ist es in

der klassifikatorischen Sacherschließung eher unüblich, derart viele Notationen für ein Dokument zu vergeben. Bei der Vergabe von Notationen orientieren sich Fachleute in Bibliotheken an den *Regeln für die Schlagwortkatalogisierung*, die sich zum Teil auf die klassifikatorische Erschließung anwenden lassen. Hier heißt es, dass das „Dokument als Ganzes, nicht einzelne Teile oder besonders relevante Aspekte“ (Scheven et al., 2017, S. 48) erschlossen werden. Daraus ergibt sich, dass auch zu viele Notationen für eine Ressource keine präzise Erschließung des Inhalts ergeben können. Annif bietet die Möglichkeit, die Anzahl der Vorschläge, die gemacht werden, zu begrenzen. Dies kann in der Datei `projects.cfg` mit der Angabe `limit=` festgelegt werden. Ich habe für die weiteren Betrachtungen ein Limit von 3 Vorschlägen pro Dokument gesetzt, da ich selten beobachtet habe, dass mehr als 3 Notationen aus einer Hauptgruppe vergeben werden. Nach meinen Berechnungen wurden in meiner Kollektion der Trainings- und Testdaten pro Dokument im Durchschnitt nur 1,3 Labels vergeben. Wenn überhaupt mehr als 3 Notationen vergeben werden, sind sie in der Regel nicht alle aus derselben Hauptgruppe, sondern es werden zusätzlich Notationen aus einer oder mehreren anderen Hauptgruppen vergeben, um eine gewisse Überschneidung von Themen deutlich zu machen. Ich habe die erste Bewertung nach Anpassung des Limits erneut gestartet und festgestellt, dass sich die Anzahl der False Positives durch meine Anpassung des Limits fast halbiert hat.

Nachdem ich Trainings und Tests mit den vier zufällig aufgeteilten Sätzen an Trainings- und Testdaten durchgeführt hatte, habe ich aus den Werten in den Evaluationsdateien der jeweiligen Tests die arithmetischen Mittelwerte gebildet und daraus einen neuen Report erstellt (siehe Abbildung 4.1). Die Abbildung zeigt den Ausschnitt mit den Werten, die in dieser Arbeit betrachtet werden sollen. Bei Auswertungen aus mehreren Durchläufen werden jeweils die arithmetischen Mittelwerte angezeigt.


```
"Precision_doc_avg": 0.3423701505870342,
"Recall_doc_avg": 0.8136115706811056,
"F1_score_doc_avg": 0.46697260700419196,
"NDCG": 0.7922636475966595,
"True_positives": 10731.25,
"False_positives": 20612.75,
"False_negatives": 2709.25,
"Documents_evaluated": 10448.0
```


Abbildung 4.1: Abstracts, Backend TF-IDF, Analyzer Snowball, vier Durchläufe

Die Werte für die einzelnen Labels, die hier nicht abgebildet sind, fallen sehr unterschiedlich aus. Es ist zu beobachten, dass diejenigen Labels, die sehr häufig in der Gesamtkollektion aller Trainingsdaten auftauchen, bessere Werte erreichen als diejenigen, die nur selten vorkommen. Eine genauere Untersuchung einzelner Labels war im Rahmen dieser Arbeit nicht mehr möglich, könnte sich aber als hilfreich herausstellen, bevor ein praktischer Einsatz von Annif erwogen wird.

4.1.2 Training ohne „Long Tail“

Um weitere Einblicke zu bekommen, welche Rolle die Häufigkeit des Auftretens einzelner Labels spielt, habe ich für die nächsten Trainings Trainingsdaten verwendet, bei denen ich als Grundlage diejenigen des vorherigen Durchlaufs verwendet habe, jedoch nur Dokumente mit Labels einbezogen habe, die in der Gesamtheit der Daten an mindestens 50 Dokumenten vergeben wurden. Dazu habe ich zunächst ein Skript geschrieben, das die zuvor genutzte Gesamtkollektion an Trainings- und Testdaten mit Abstracts durchläuft und diejenigen Dokumente in eine neue Datei schreibt, die nur Labels enthalten, die an mindestens 50 Dokumenten vergeben sind. Diese habe ich in gleicher Weise wie zuvor vier Mal unterschiedlich in jeweils 80 % Trainingsdaten und 20 % Testdaten aufgeteilt. Es wurden die gleichen Parameter genutzt wie bei den letzten Trainings, sodass in Annif kein neues Projekt erstellt werden musste. Nach den vier Trainings- und Testdurchläufen habe ich wieder eine Übersicht mit den arithmetischen Mittelwerten der Ergebnisse erstellt. Diese wird in Abbildung 4.2 dargestellt.


```
"Precision_doc_avg": 0.3598253425839632,
"Recall_doc_avg": 0.8558271633702669,
"F1_score_doc_avg": 0.49112329600835347,
"NDCG": 0.8389025796423379,
"True_positives": 10424.5,
"False_positives": 18546.5,
"False_negatives": 1854.5,
"Documents_evaluated": 9657.0
```


Abbildung 4.2: Abstracts ohne Long Tail, Backend TF-IDF, Analyzer Snowball, vier Durchläufe

Es fällt auf, dass die Werte sich verbessert haben, obwohl insgesamt weniger Dokumente für das Training verwendet wurden (siehe Tabelle in Abbildung 3.6), was darauf schließen lässt, dass Annif mit TF-IDF bessere Ergebnisse erzielt, wenn in den Trainingsdaten keine Labels vorhanden sind, die nur sehr selten vorkommen. Diese Erkenntnis ist für den Bibliotheksalltag nur bedingt nützlich, da prinzipiell alle in der RVK vorkommenden

Notationen für die klassifikatorische Sacherschließung zur Verfügung stehen. Wenn selten vorkommende Labels in den Trainingsdaten gar nicht vorkommen, können sie von Annif nicht berücksichtigt werden und würden nach und nach keine Rolle mehr spielen. Es passiert in regelmäßigen Abständen, dass RVK-Notationen aus der RVK entfernt werden, was für Bibliotheken mit einem großen Bestand an Exemplaren mit diesen Notationen Mehraufwand zur Folge hat, da eine Umarbeitung erforderlich wird. Es stellt sich die Frage, ob es sinnvoll ist, die RVK im Zuge der Automatisierung an neue Bedürfnisse anzupassen. Möglicherweise ist die RVK an sich nicht mehr das geeignetste Mittel, mit dem Texte inhaltlich erschlossen werden sollten, da sie viele Notationen enthält, die zu selten Anwendung finden.

4.1.3 Training mit Dokumenten, die nur zum Teil Abstracts enthalten

Da man die größte Masse an Dokumenten erhält, wenn sämtliche zur Verfügung stehenden Titeldatensätze in die Trainings- und Testdatenmenge aufgenommen werden, unabhängig davon, ob auch Abstracts oder Inhaltsverzeichnisse enthalten sind, habe ich noch untersucht, wie Annif auf ein Training mit Dokumenten reagiert, die nicht alle ein Abstract oder Inhaltsverzeichnis enthalten. Mit einer Precision von gerundet 0,23 und Recall von gerundet 0,52 lagen die Ergebnisse deutlich unter denen, bei deren Training alle Titeldatensätze mit Abstracts bzw. Inhaltsverzeichnissen versehen waren. Demnach ist es nicht empfehlenswert, bei der Auswahl an Trainingsdaten mit allen Mitteln Masse zu generieren.

Die bisherigen Untersuchungen haben gezeigt, dass im Backend TF-IDF ein Training mit Titeldatensätzen mit Abstracts oder Inhaltsverzeichnissen die besten Ergebnisse erzielt, wenn diejenigen Datensätze für das Training ausgelassen werden, die Labels enthalten, die weniger als 50-mal im gesamten Korpus auftreten. Genau genommen geht es aber darum, ein Modell zu schaffen, das auch selten vorkommende Notationen zuverlässig vergeben kann. Außerdem fällt auf, dass die Precision im Vergleich zum Recall niedrig ist. Annif kann in der Regel eine Notation gut vorhersagen, liefert aber bei mindestens drei Notationen schlechtere Ergebnisse. Dabei muss bedacht werden, dass nicht alle Titeldatensätze der Trainings- und Testdaten mehr als eine Notation enthalten, sodass Annif bei dieser Einstellung gezwungen ist, falsche Notationen vorzuschlagen. Um bessere Werte zu erzielen, müsste man das Limit auf 1 setzen, damit Annif nur genau eine

Notation vorschlägt. Gleichzeitig stellt dies eine Begrenzung dar, die dazu führen kann, dass Bewertende aus bibliothekarischem Fachpersonal kaum eine Erleichterung verspüren, weil sie eine vollständige Sacherschließung vornehmen müssten, wenn sie zusätzliche Notationen vergeben wollen. Eine bessere Lösung besteht darin, einen Schwellwert zu setzen und nur Ergebnisse zu berücksichtigen, die über diesem liegen, die also mit einer Mindestwahrscheinlichkeit zutreffend sind. Ich werde später noch die Möglichkeiten prüfen, die Annif in diesem Zusammenhang anbietet.

4.1.4 Nutzung verschiedener Analyzer

Da die beste Effektivität bisher aus dem Training mit Trainingsdaten, in denen jedes Label an mindestens 50 Objekten vergeben war, erreicht wurde, habe ich diese Daten verwendet, um mit dem Backend TF-IDF verschiedene Analyzer zu testen. Es sollte geprüft werden, ob eine zusätzliche Lemmatisierung insgesamt bessere Ergebnisse hervorbringt.

Ich habe zunächst den Simplemma Analyzer verwendet und festgestellt, dass der F1-Score sich hier im Gegensatz zur Nutzung des Snowball Analyzers verschlechtert. Die Ergebnisse sind in Abbildung 4.3 dargestellt.


```
"Precision_doc_avg": 0.3359134997066031,
"Recall_doc_avg": 0.7975195022608815,
"F1_score_doc_avg": 0.45824966069218936,
"NDCG": 0.7829296860591883,
"True_positives": 9731.75,
"False_positives": 19239.25,
"False_negatives": 2547.25,
"Documents_evaluated": 9657.0
```


Abbildung 4.3: Abstracts ohne Long Tail, Backend TF-IDF, Analyzer Simplemma, vier Durchläufe

Auch mit dem Spacy Analyzer habe ich die entsprechenden Trainings und Tests durchgeführt. Betrachtet man die Werte, wird mit dem Spacy Analyzer eine bessere Effektivität erreicht als mit dem Simplemma Analyzer, aber die Werte sind schlechter als bei Nutzung des Snowball Analyzers. Die Werte sind in Abbildung 4.4 dargestellt. Dadurch zeigt sich, dass eine Lemmatisierung für das vorliegende Problem nicht notwendigerweise eine Verbesserung darstellt. Ich werde noch untersuchen, ob die anderen zu prüfenden Verfahren von der Lemmatisierung profitieren.

```
"Precision_doc_avg": 0.34994477235856547,
"Recall_doc_avg": 0.8303575126850988,
"F1_score_doc_avg": 0.4772359499370993,
"NDCG": 0.8104171517502724,
"True_positives": 10138.25,
"False_positives": 18832.75,
"False_negatives": 2140.75,
"Documents_evaluated": 9657.0
```


Abbildung 4.4: Abstracts ohne Long Tail, Backend TF-IDF, Analyzer Spacy, vier Durchläufe

4.1.5 Zusammenfassung TF-IDF

Die Ergebnisse aus den Tests mit dem Verfahren TF-IDF waren nicht besonders vielversprechend. Insbesondere der Wert für die Precision liegt weit unter den Erwartungen. Die Herausforderung liegt darin, einen Weg zu finden, dass Annif mehr als ein Label für ein Dokument vorzuschlagen, wenn alle zutreffend sind. Gleichzeitig sollte es nicht die Pflicht geben, eine Mindestanzahl an Vorschlägen zu unterbreiten, wenn die Wahrscheinlichkeit, dass sie passen, gering ist. Was die unterschiedliche Zusammenstellung der Trainings- und Testdaten angeht, hat sich gezeigt, dass mit TF-IDF die besten Ergebnisse erzielt werden, wenn man darauf achtet, dass zu jedem Dokument ein Abstract oder Inhaltsverzeichnis zur Verfügung steht und dass nur Labels berücksichtigt werden, die an mindestens 50 Objekten vergeben sind. Abbildung 4.5 stellt die Ergebnisse mit unterschiedlichen Trainingsdaten dar.

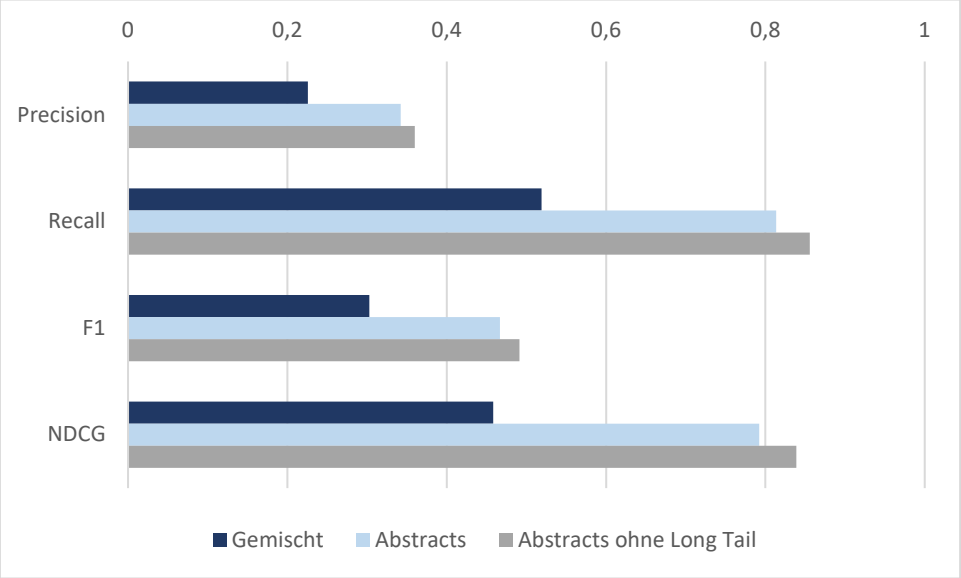

Abbildung 4.5: Vergleich mit unterschiedlichen Trainingsdaten, Backend TF-IDF, Analyzer Snowball, vier Durchläufe

Trainings und Auswertungen mit den Analyzern Simplemma und Spacy haben ergeben, dass die Ergebnisse sich durch die Nutzung von Analyzern, die zusätzlich eine Lemmatisierung vornehmen, nicht verbessern lassen. Es wird noch zu prüfen sein, ob die Anpassung der Vorverarbeitung bei anderen Verfahren eine Rolle spielt. Abbildung 4.6 zeigt den Vergleich der Ergebnisse nach Trainings und Tests mit den verschiedenen Analyzern.

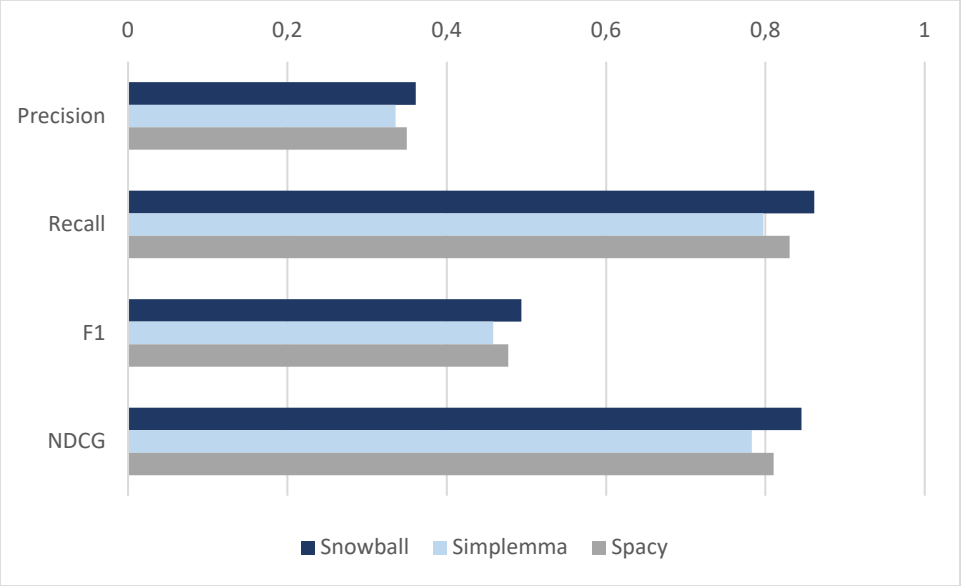

Abbildung 4.6: Vergleich Analyzer, Abstracts ohne Long Tail, Backend TF-IDF, vier Durchläufe

Die Trainings und Tests mit TF-IDF geben einen ersten Einblick in den Umgang mit Annif und nehmen die Scheu vor dem Werkzeug. Die erzielten Ergebnisse haben gezeigt, dass man mit Annif bereits vielversprechende Ergebnisse erreichen kann, aber ein Projekt mit TF-IDF nur als Einstieg dient, oder wie das Entwicklungsteam von Annif TF-IDF feststellt:

> *"The „Hello World" algorithm of automated subject indexing: quick to set up, train and test, but not the final say!"* (Suominen et al., 2023)

4.2 Omikuji

Bei dem Backend Omikuji handelt es sich um ein Verfahren, das mit Entscheidungsbäumen arbeitet. Ein Entscheidungsbaum beginnt an der Wurzel und bietet an jedem Knoten Abzweigungen, über deren Wahl man sich der Vergabe eines Labels nähert. Am Blatt stehen die auszuwählenden Labels. Omikuji nutzt Parabel und Bonsai, zwei Algorithmen,

die sich besonders für die Multilabel Classification eignen. In Annif beträgt die Tiefe des Baumes bei Omikuji 20, d.h. es gibt 20 Entscheidungen (The National Library of Finland, 2023a). Wie für das Projekt TF-IDF habe ich mit den unterschiedlich zusammengestellten Daten jeweils ein Modell trainiert und ausgewertet. Als Analyzer wurde zunächst der Snowball Analyzer verwendet und das Limit der vorzuschlagenden Labels wurde auf 3 gesetzt. Es wurden die gleichen Daten für die Trainings und Tests verwendet wie bei TF-IDF und wieder jeweils vier Durchgänge absolviert, aus deren Ergebnissen die arithmetischen Mittelwerte gebildet wurden.

4.2.1 Titel mit Abstracts als Trainingsdaten

Für das erste Training wurden die Trainingsdaten mit den Titeldatensätzen verwendet, die neben dem Titel auch ein Abstract oder Inhaltsverzeichnis enthalten. Die Auswertung zeigt, dass die Effektivität im Vergleich zu TF-IDF besser ist; der Wert für die Precision ist aber auch hier niedrig. Abbildung 4.7 zeigt die zu betrachtenden Werte.


```
"Precision_doc_avg": 0.39990109749872377,
"Recall_doc_avg": 0.945015820981915,
"F1_score_doc_avg": 0.5438862619854197,
"NDCG": 0.9397992236437254,
"True_positives": 12534.5,
"False_positives": 18809.5,
"False_negatives": 904.5,
"Documents_evaluated": 10448.0
```


Abbildung 4.7: Abstracts, Backend Omikuji, Analyzer Snowball, vier Durchläufe

Durch die niedrige Precision wird auch der F1-Score niedrig. Die Vermutung ist, dass die Problematik zum einen in der ungleichen Verteilung der Labels zu suchen ist und zum anderen in der Einstellung, durch die selbst mit gesetztem Limit von 3 eine Angabe von Vorschlägen forciert wird, die mit hoher Wahrscheinlichkeit unzutreffend sind. Letzteres werde ich an späterer Stelle noch genauer untersuchen.

4.2.2 Training ohne „Long Tail“

Für ein nächstes Training habe ich wieder diejenigen Daten verwendet, bei denen nur die Labels betrachtet werden, die innerhalb der Gesamtmenge aus Trainings- und Testdaten an mindestens 50 Dokumenten vergeben wurden. Wie erwartet sind die Werte mit diesen Daten höher, die Werte für Precision und F1 jedoch nach wie vor niedrig. Die Werte werden in Abbildung 4.8 gezeigt.

```
"Precision_doc_avg": 0.40591453522487997,
"Recall_doc_avg": 0.9653424976700838,
"F1_score_doc_avg": 0.5538346919094046,
"NDCG": 0.9617816751325853,
"True_positives": 11759.75,
"False_positives": 17211.25,
"False_negatives": 519.25,
"Documents_evaluated": 9657.0
```


Abbildung 4.8: Abstracts ohne Long Tail, Backend Omikuji, Analyzer Snowball, vier Durchläufe

Interessant an dieser Stelle ist die Beobachtung, dass die Anzahl der False Negatives, also Labels, die nicht vorgeschlagen wurden, obwohl sie zutreffen, sich im Vergleich zu den Daten mit Long Tail beträchtlich verringert haben. Daran zeigt sich, dass Annif in der Tat besser lernt, wenn zum Lernen eine Mindestanzahl an Objekten für ein Label zur Verfügung steht. Das bestätigt die Ausführungen in der Literatur, die besagen, dass neben der Qualität der Daten auch die Häufigkeit der Objekte pro Label entscheidend ist, „denn ein Algorithmus kann nur Zusammenhänge lernen, die in dem Trainingsdatensatz vorhanden sind“ (Aust, 2021, S. 47f.). So ist es für Annif schwierig bis unmöglich, eine RVK-Notation zu lernen, die in den Trainingsdaten nur mit einem einzigen Dokument verknüpft ist.

4.2.3 Training mit Dokumenten, die nur zum Teil Abstracts enthalten

Auch für das Projekt Omikuji wurde ein Training mit einer größeren Datenmenge durchgeführt, in der nur ein Teil der Titeldatensätze mit Abstracts versehen war. Nach den Ergebnissen mit diesen Daten und TF-IDF habe ich davon nicht viel erwartet, wollte aber der Vollständigkeit halber auch in diesem Projekt alle ausgewählten Daten auswerten. Die Ergebnisse sind in Abbildung 4.9 dargestellt. Meine Erwartung, dass die Effektivität dieses Trainings im Vergleich zu den Trainings mit den zuvor getesteten Trainingsdaten schlechter ist, wurde bestätigt.


```
"Precision_doc_avg": 0.343715997731905,
"Recall_doc_avg": 0.7856790109267006,
"F1_score_doc_avg": 0.46011067164232533,
"NDCG": 0.7442181060526514,
"True_positives": 22428.5,
"False_positives": 42594.25,
"False_negatives": 7134.75,
"Documents_evaluated": 21751.0
```


Abbildung 4.9: Gemischt, Backend Omikuji, Analyzer Snowball, vier Durchläufe

4.2.4 Vergleich TF-IDF und Omikuji

Im Vergleich mit TF-IDF schneidet das Backend Omikuji zwar besser ab, aber die Problematik der niedrigen Precision von weit unter 0,5 ist nach wie vor vorhanden. Wie bereits mehrfach erwähnt, scheinen die voreingestellten Werte unpassend für ein Vorhaben mit den vorliegenden Daten. Der Vergleich der Werte aus den beiden Backends TF-IDF und Omikuji in Abbildung 4.10 veranschaulicht, dass alle dargestellten Werte sich mit Omikuji verbessert haben, aber durch die niedrige Precision weiterhin der F1-Score gering bleibt. Im nächsten Abschnitt werde ich versuchen, die Werte durch die Anpassung von Einstellungen zu optimieren.

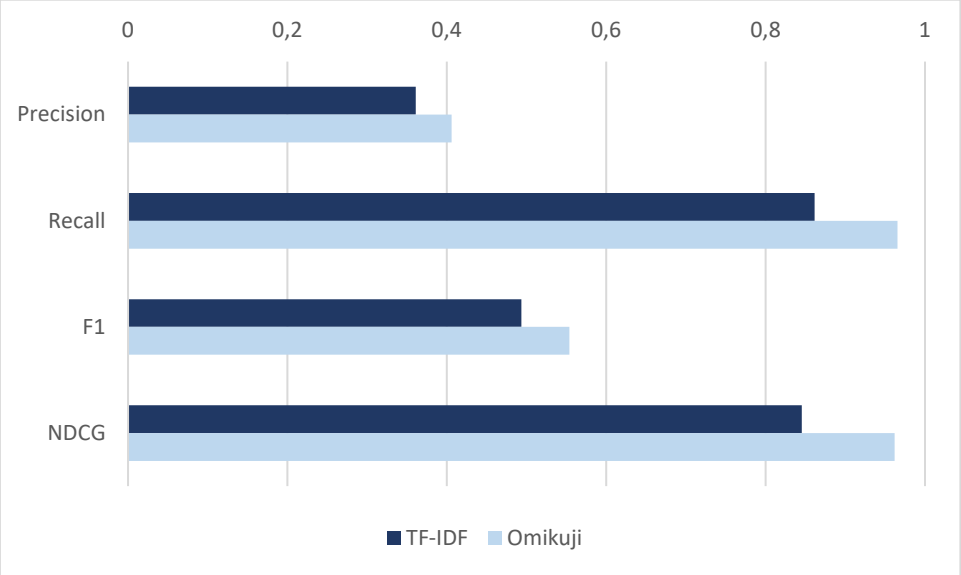

Abbildung 4.10: Ohne Long Tail, Backends TF-IDF und Omikuji, Analyzer Snowball, vier Durchläufe

4.2.5 Optimierung

Da der Wert für Precision auch im Projekt Omikuji nicht zufriedenstellend ist und die Vermutung nahe liegt, dass ein Grund hierfür in den Einstellungen liegt, die verlangen, dass Annif eine Mindestanzahl an Ergebnissen ausgibt, soll versucht werden, die Einstellungen so zu optimieren, dass nur Labels vorgeschlagen werden, bei denen eine Mindestwahrscheinlichkeit für eine Übereinstimmung mit dem Goldstandard vorliegt. Standardmäßig werden in Annif pro Objekt 10 Vorschläge für passende Labels ausgegeben. Wie bereits gezeigt, kann in der Projektkonfiguration über die Einstellung `limit` eine Höchstanzahl an Vorschlägen festgelegt werden. Diese Einstellung wird in dieser Arbeit als Limit bezeichnet. Das Limit kann auch für die Evaluierung zusammen mit dem

Befehl `annif eval` übergeben werden und so die Produktkonfiguration überschreiben. Zusätzlich kann hier ein Schwellwert (threshold) angegeben werden. Dieser gibt an, welcher Konfidenzwert (suggest score, siehe Kapitel 3.4) mindestens erreicht werden muss, damit Annif einen Vorschlag ausgibt.

Über den Befehl `optimize` wird in Annif die Kombination verschiedener Limits und Schwellwerte getestet. Ausgegeben wird eine Liste von Limits und Schwellwerten mit den dazugehörigen Werten für Precision, Recall und F1-Score. So kann entschieden werden, welche Einstellungen für das eigene Projekt sinnvoll sind (University of Helsinki, 2017).

Für diesen Test habe ich den Korpus an Dokumenten mit Abstracts und einem Vorkommen der einzelnen Labels an mindestens 50 Objekten neu aufgeteilt. Zunächst wurden die Daten über ein Zufallsprinzip in 80 % Trainingsdaten und 20 % Testdaten aufgeteilt. Dann habe ich die Testdaten erneut unterteilt, um eine Hälfte der Testdaten für die Optimierung zu nutzen und die andere Hälfte, also Daten, die Annif während des Trainings- und Optimierungsprozesses nicht verarbeitet hat, für die Tests zu nutzen. So wird die Aufteilung in unterschiedliche Größen auch im Annif Wiki für die Arbeit mit dem Befehl `optimize` empfohlen (The National Library of Finland, 2021).

Nachdem das Training durchgeführt wurde, nutzt man den Befehl `optimize` in Verbindung mit dem Teil der Testdaten, die für die Optimierung zurückgelegt wurden. Am Ende der Berechnungen wird ausgegeben, welche Kombination als optimale Einstellung für Precision, Recall und F1 verwendet werden kann. Die Abbildung zeigt die Ausgabe.

```
Best Precision (doc avg): 0.9460        Limit: 1        Threshold: 0.00
Best    Recall (doc avg): 0.9668        Limit: 15       Threshold: 0.00
Best  F1 score (doc avg): 0.9289        Limit: 15       Threshold: 0.10
Documents evaluated:    4829
```

Abbildung 4.11: Ausgabe nach Optimierungsdurchlauf

Nicht überraschend reicht es aus, für optimale Werte bei Precision und Recall jeweils das Limit für Precision auf 1 und für Recall auf 15 zu setzen. Ein Schwellwert ist für eine gute Precision dann nicht mehr notwendig, da die Wahrscheinlichkeit für eine Übereinstimmung mit dem Goldstandard bei einem einzigen Vorschlag ohnehin hoch ist.

Gleichzeitig ist es nicht verwunderlich, dass man einen hohen Recall erreicht, wenn man das Limit hochsetzt, wie hier auf 15, da nun ein großer Teil der „korrekten“ Labels vorgeschlagen werden können. Aus diesen Gründen ist die Betrachtung von Precision und Recall an dieser Stelle nicht zielführend. Interessant ist jedoch zu sehen, mit welchen Einstellungen ein hoher F1-Score erreicht wird, da ein hoher F1-Score eine Ausgeglichenheit zwischen Precision und Recall erzielt. Annif hat berechnet, dass der F1-Score mit einem Limit von 15 und einem Schwellwert von 0,10 bei den vorliegenden Daten am höchsten liegt.

Um zu prüfen, ob die von Annif vorgeschlagenen Einstellungen auch bei Nutzung neuer Daten gute Werte hervorbringen, werden diejenigen Daten, die weder für das Training noch für die Optimierung genutzt wurden, für die Evaluation genutzt. In der Evaluation kann man die Optionen `--limit` und `--threshold` nutzen und mit dem Befehl `annif eval` kombinieren. Abbildung 4.12 zeigt die Ergebnisse aus dem Test.


```
"Precision_doc_avg": 0.9241231703949186,
"Recall_doc_avg": 0.9447459265396297,
"F1_score_doc_avg": 0.9271215528464907,
"NDCG": 0.9459323320908087,
"True_positives": 5705,
"False_positives": 594,
"False_negatives": 392,
"Documents_evaluated": 4828
```


Abbildung 4.12: Omikuji mit optimize, Limit 15, Threshold 0,10, Analyzer Snowball, ein Durchlauf

Der Test hat ergeben, dass die vorgeschlagenen Einstellungen den F1-Score verbessern. Im Vergleich mit den Werten bei der Evaluation ohne Limit und Schwellwert lässt sich feststellen, dass der Recall nach der Optimierung geringfügig gesunken ist. Das liegt daran, dass ich den Fokus auf den F1-Score gelegt habe, der das harmonische Mittel zwischen den beiden anderen Werten darstellt. Das Herstellen einer Ausgeglichenheit zwischen Recall und Precision geht hier auf Kosten des Recall. Der Unterschied ist jedoch so gering, dass es meiner Einschätzung nach in diesem Fall vertretbar ist. In Abbildung 4.13 sind die Unterschiede vor und nach der Optimierung grafisch dargestellt.

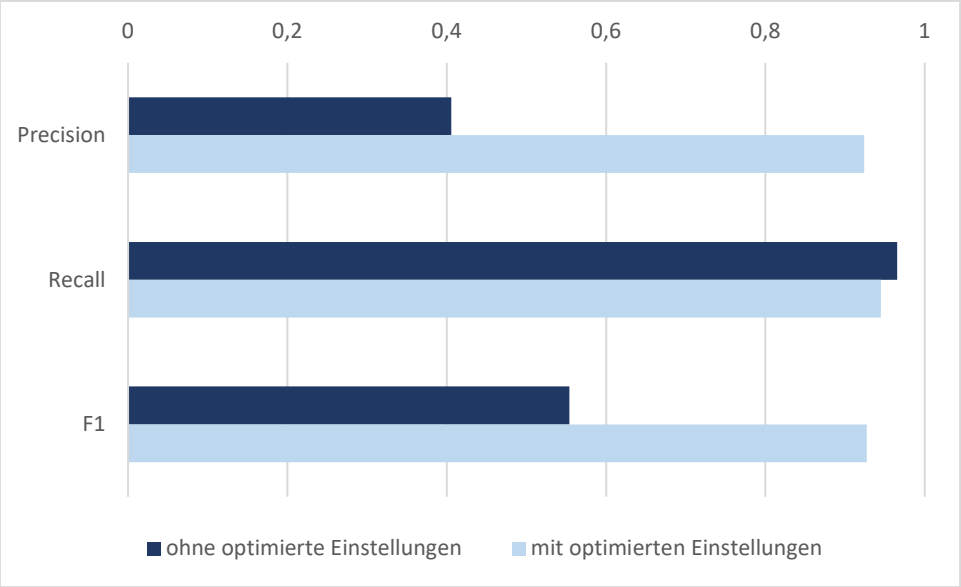

Abbildung 4.13: Vergleich mit und ohne optimize, Abstracts ohne Long Tail, Backend Omikuji, Analyzer Snowball, ein Durchlauf mit optimize, vier Durchläufe ohne optimize

Da die Werte nach diesem Test hoch waren und ich ausschließen wollte, dass es sich hier um Ausreißer handelt, die durch die Verteilung der Daten zustande gekommen sein könnten, habe ich die Daten erneut nach dem Zufallsprinzip aufgeteilt und das Training und die Optimierung damit wiederholt. Auch hier hat Annif mit der Optimierung berechnet, dass der beste F1-Score mit einem Limit von 15 und einem Schwellwert von 0,1 zu erreichen ist. Wie Abbildung 4.14 zeigt, lagen die Werte in einem ähnlichen Bereich.


```
"Precision_doc_avg": 0.9856738470035901,
"Recall_doc_avg": 0.9917564208782104,
"F1_score_doc_avg": 0.9871128733183413,
"NDCG": 0.9982764646443522,
"True_positives": 6089,
"False_positives": 115,
"False_negatives": 48,
"Documents_evaluated": 4828
```


Abbildung 4.14: Abstracts ohne Long Tail, Backend Omikuji, Analyzer Snowball, ein Durchlauf

Für einen weiteren Test habe ich Dokumente genutzt, die ich erneut über die Schnittstelle des K10plus heruntergeladen habe. Hier sind also zum Teil Dokumente enthalten, die in den letzten ca. 3 Monaten neu hinzugekommen sind, sodass Annif einen Teil davon bisher nicht verarbeitet hat. Hierbei lagen Precision und Recall wieder etwas weiter auseinander, wie die Abbildung 4.15 zeigt. Insgesamt zeigen die zusätzlichen Tests, dass die Ergeb-

nisse gut ausgefallen sind und es sich bei den ersten Ergebnissen nicht um ungewöhnliche Ausreißer handelt.


```
"Precision_doc_avg": 0.8691580287324967,
"Recall_doc_avg": 0.966357519549009,
"F1_score_doc_avg": 0.8985244070350453,
"NDCG": 0.9555514501197477,
"True_positives": 1962,
"False_positives": 488,
"False_negatives": 73,
"Documents_evaluated": 1833
```


Abbildung 4.15: Abstracts ohne Long Tail, neue Daten, Backend Omikuji, Analyzer Snowball, ein Durchlauf

4.2.6 Nutzung verschiedener Analyzer

Auch für das Backend Omikuji hatte ich in den ersten Durchläufen den Analyzer Snowball genutzt und später Durchläufe mit Simplemma und Spacy durchgeführt. Wie auch im Backend TF-IDF hat sich herausgestellt, dass die Nutzung von Lemmatisierung keine Verbesserung bringt. Alle Analyzer erzeugen fast identische Ergebnisse. Der Vergleich ist in Abbildung 4.16 dargestellt.

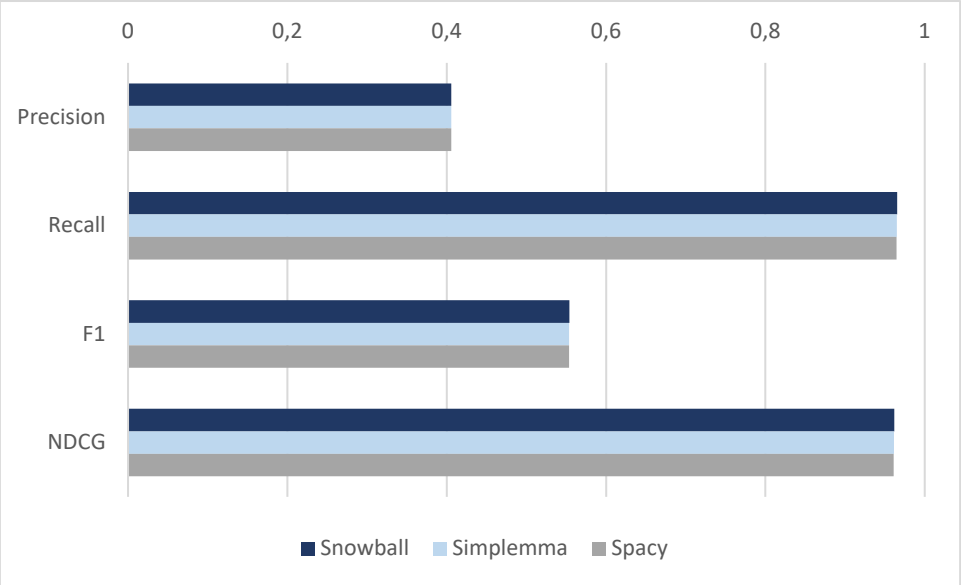

Abbildung 4.16: Vergleich Analyzer, Abstracts ohne Long Tail, Backend Omikuji, vier Durchläufe

4.2.7 Zusammenfassung Omikuji

Es konnte gezeigt werden, dass mit dem Backend Omikuji im Vergleich zu TF-IDF bessere Effektivitätswerte erzielt werden konnten. Gleichzeitig ist deutlich geworden, dass

in beiden Backends die Problematik verbleibt, dass mit den Grundeinstellungen zu viele Vorschläge verlangt werden und Annif gezwungenermaßen unzutreffende Labels in die Vorschläge aufnehmen muss, um die Erwartung zu erfüllen. Dadurch erhöht sich die Anzahl der False Positives, wodurch die Precision gesenkt wird.

Mit dem Befehl `optimize` bietet Annif die Möglichkeit, die Einstellungen zu finden, mit denen die Werte optimiert werden. Eine Limitierung der Vorschläge und der Einsatz eines Schwellwerts ergeben Sinn, da sie den bibliothekarischen Alltag widerspiegeln, in dem ein Dokument mit einer begrenzten Anzahl an RVK-Notationen erschlossen wird. Die Optimierungsroutine in Annif stellt hier einen guten Weg dar, um die passenden Einstellungen mit wenig zusätzlichem Aufwand zu finden.

4.3 Fasttext

Das Backend Fasttext basiert auf Word Embeddings. Unter Word Embeddings versteht man Vektorrepräsentationen von Wörtern im mehrdimensionalen Raum. Das bedeutet, jedem Wort bzw. jedem Term werden Werte zugeordnet, die einem Vektor entsprechen. Beziehungen zwischen Termen können durch ihre Nähe zueinander im Raum erfasst werden. Terme, die eng beieinander liegen, ähneln sich beispielsweise durch ihre Bedeutung oder ihren Kontext (Heidenreich, 2018). Fasttext bedient sich der von Facebook Research entwickelten fastText library und gleicht einem neuronalen Netzwerk (The National Library of Finland, 2022a). In neuronalen Netzen werden die Eingabedaten über mehrere Schichten geleitet und sich jeweils durch Anpassung von Gewichtungen dem Zielwert angenähert (Biemann et al., 2022, S. 81f.). Im Vergleich zu den bisher getesteten Backends ist Fasttext rechenintensiver und erfordert die Installation eines zusätzlichen Paketes. In der Projektkonfiguration müssen weitere Einstellungen vorgenommen werden (The National Library of Finland, 2022a). Das Annif Wiki bietet zwei Beispielkonfigurationen, von denen ich diejenige gewählt habe, die die Grundeinstellungen verwendet und weniger Trainingszeit benötigt. Wieder habe ich jedes Training vier Mal durchgeführt und aus den Ergebnissen die arithmetischen Mittelwerte gebildet.

Die Einstellungsmöglichkeiten für Fasttext sind sehr komplex. Da Annif die Einstellungen direkt an Fasttext übergibt, gibt es keine Möglichkeit, sie auf komfortablem Weg durchzutesten, wie es für Limit und Schwellwert über den Befehl `optimize` möglich ist. Beispielsweise kann man festlegen, wie viele Durchläufe es durch die Schichten des

neuronalen Netzes geben soll oder wie viele Sätze als ein Chunk betrachtet werden sollen. Chunks werden jeweils einzeln ausgewertet und aus den Ergebnissen ein Durchschnitt gebildet (The National Library of Finland, 2022a) Weitere Informationen findet man in der Fasttext Dokumentation (Facebook Open Source, 2022).

4.3.1 Titel mit Abstracts

Die Ergebnisse aus den Tests mit den Trainings- und Testdaten, von denen alle Titeldatensätze Abstracts enthalten und in denen auch die selten vorkommenden Labels enthalten sind, spiegeln das wider, was in den Tests mit den Backends TF-IDF und Omikuji schon erkannt wurde. Durch die geringe Anzahl an zutreffenden Labels pro Objekt ist der Recall hoch, da die Wahrscheinlichkeit hoch ist, dass alle zutreffenden Labels in der Menge der Vorschläge enthalten ist. Ebenso ist die Wahrscheinlichkeit hoch, dass in der Liste der Vorschläge Labels enthalten sind, die nicht zutreffen, also False Positives, da von Annif eine Mindestanzahl an Vorschlägen erwartet wird. Abbildung 4.17 zeigt die Ergebnisse.


```
"Precision_doc_avg": 0.28152716628381824,
"Recall_doc_avg": 0.8892331296032961,
"F1_score_doc_avg": 0.4151440674244082,
"NDCG": 0.8754458173241483,
"True_positives": 11761.75,
"False_positives": 30021.25,
"False_negatives": 1677.25,
"Documents_evaluated": 10448.0
```


Abbildung 4.17: Abstracts, Backend Fasttext, Analyzer Snowball, vier Durchläufe

Der Test zeigt auch, dass Fasttext mit den vorliegenden Einstellungen einen niedrigeren F1-Score liefert als TF-IDF und Omikuji. Da in Fasttext viele zusätzliche Einstellungen individuell vorgenommen werden müssen, vermute ich, dass man hier bessere Ergebnisse erzielen kann, wenn man verschiedene Einstellungen testet und sich nach und nach einer optimalen Konfiguration annähert. Im Rahmen dieser Masterarbeit konnte ich hier nicht weiter in die Tiefe gehen und habe nur die im Annif Wiki vorgeschlagenen Grundeinstellungen getestet.

4.3.2 Training ohne „Long Tail“

Auch im Backend Fasttext habe ich das Training mit den Trainings- und Testdaten durchgeführt, bei denen ich nur Labels aufgenommen habe, die an 50 oder mehr Dokumenten in der Kollektion vergeben sind. Die Ergebnisse werden in Abbildung 4.18 gezeigt.

```
"Precision_doc_avg": 0.3833316074695384,
"Recall_doc_avg": 0.914036018777398,
"F1_score_doc_avg": 0.5236047287771426,
"NDCG": 0.9079199041947381,
"True_positives": 11105.0,
"False_positives": 17865.0,
"False_negatives": 1174.0,
"Documents_evaluated": 9657.0
```


Abbildung 4.18: Abstracts ohne Long Tail, Backend Fasttext, Analyzer Snowball, vier Durchläufe

Auch hier bewahrheitet sich die Vermutung, dass ein Training mit Labels, die an 50 oder mehr Dokumenten vergeben sind, zu höheren Werten führt. Der Wert für Precision und somit auch der F1-Score sind durch die bereits erläuterten Umstände zu niedrig, um Annif für eine Arbeit in der Bibliothek einzusetzen. Gleiches gilt, wenn man das Training mit gemischten Daten durchführt, also zwar eine größere Menge an Objekten nimmt, die aber nur zum Teil Abstracts oder Inhaltsverzeichnisse enthalten.

4.3.3 Optimierung

Um zu sehen, inwieweit sich die problematischen Werte, die durch die geringe Anzahl an zutreffenden Labels zustande kommen, verbessern lassen, habe ich mit dem Backend Fasttext ebenfalls den Befehl `optimize` getestet. Damit wollte ich prüfen, ob sich auch in diesem Backend die Ergebnisse mit einem Limit und einem Schwellwert optimieren lassen. Ich habe für das Training die gleichen Trainingsdaten genommen, die ich für die Optimierung in Omikuji schon zusammengestellt hatte.

Mit den Vorschlägen für den besten F1-Score habe ich dann die Evaluierung durchgeführt. Die Ergebnisse (in Abbildung 4.19 dargestellt) zeigen, dass mit den Einstellungen, die mit dem Befehl `optimize` errechnet wurden, eine Ausgeglichenheit zwischen Precision und Recall erreicht werden kann. Allerdings sind die Werte geringer als bei Omikuji oder TF-IDF.

```
"Precision_doc_avg": 0.89529826014913,
"Recall_doc_avg": 0.9032553162109913,
"F1_score_doc_avg": 0.8926628397838009,
"NDCG": 0.9066411661129323,
"True_positives": 5423,
"False_positives": 590,
"False_negatives": 674,
"Documents_evaluated": 4828
```


Abbildung 4.19: Abstracts ohne Long Tail, Backend Fasttext, Analyzer Snowball, Limit 15, Threshold 0,10, ein Durchlauf

4.3.4 Nutzung verschiedener Analyzer

Auch in dem Backend Fasttext wurden die Ergebnisse nach Trainings mit den verschiedenen Analyzern verglichen. Hierzu habe ich die Trainings- und Testdaten genutzt, in denen jedes Label mit mindestens 50 Objekten verknüpft ist, da diese im Vergleich zu den anderen Trainings- und Testdaten am besten abgeschnitten haben. Abbildung 4.20 zeigt die Ergebnisse im Vergleich.

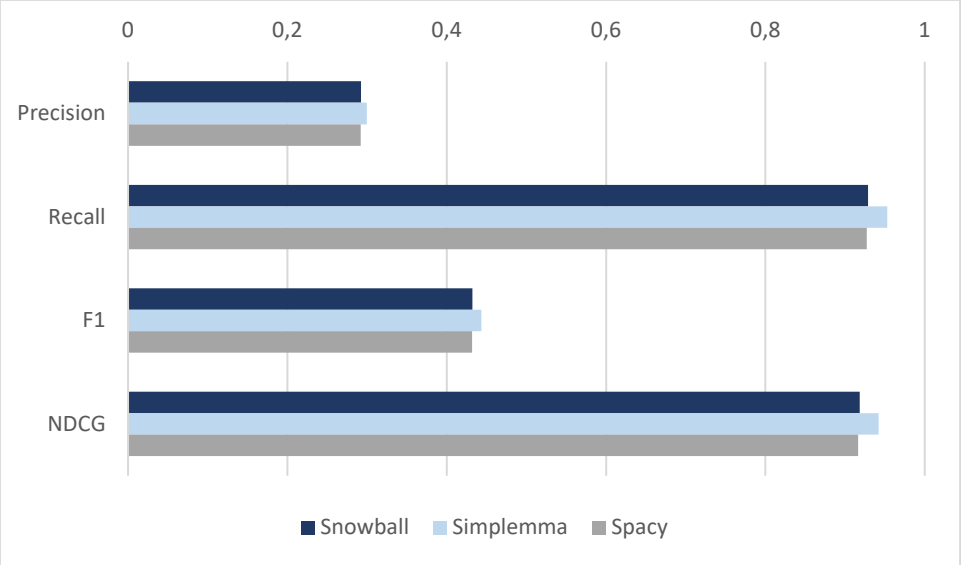

Abbildung 4.20: Vergleich Analyzer, Abstracts ohne Long Tail, Backend Fasttext, vier Durchläufe

Die Ergebnisse zeigen, dass Fasttext mit dem Analyzer Simplemma die höchsten Werte erreicht. Zwar sind die Unterschiede minimal, aber es würde sich in einem nächsten Schritt lohnen, die weiteren Tests der Einstellungen im Backend Fasttext mit dem Analyzer Simplemma fortzuführen. Da in Fasttext mit Word Embeddings gearbeitet wird, vermute ich, dass die Lemmatisierung hier eine größere Rolle als z. B. in TF-IDF oder in

Omikuji spielt, da die Lemmata verschiedene Wörter, die alle von einem Wort mit der gleichen Bedeutung stammen, zusammenfassen.

4.3.5 Zusammenfassung Fasttext

Meine Tests mit dem Backend Fasttext haben zunächst ergeben, dass die Ergebnisse zwar vielversprechend sind, aber hinter denen, die sich aus dem Backend Omikuji ergeben haben, zurückbleiben. Dennoch kann ich mir vorstellen, dass sich eine genauere Untersuchung hier lohnen könnte. Dazu müssten jedoch die einzelnen Einstellungen ausgetestet und verglichen werden, was viel Zeit erfordert und mir deshalb im Rahmen dieser Arbeit nicht möglich ist. Für eine Bibliothek wäre es hier von Vorteil, jemanden mit Erfahrung im Bereich maschineller Lernverfahren um Hilfe zu bitten. Ich vermute außerdem, dass die Auswahl der Trainingsdaten für Fasttext von besonderer Bedeutung ist und möglicherweise hier eine Nutzung von Volltexten weitere Aufschlüsse über die Eignung des Backends geben kann.

4.4 Vergleich TF-IDF, Omikuji und Fasttext

Wenn man die Ergebnisse aller getesteten Backends vergleicht, stellt man fest, dass mit dem Backend Omikuji in allen Bereichen die besten Ergebnisse erzielt werden (s. Abbildung 4.21).

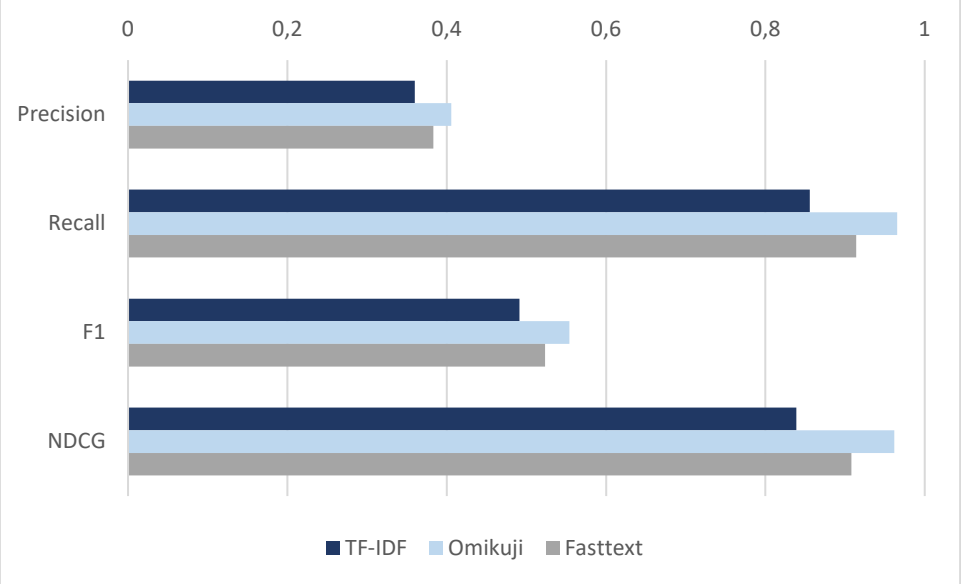

Abbildung 4.21: Vergleich Backends, Analyzer Snowball, vier Durchläufe

Da sich die Optimierung über den Befehl `optimize` und die darauffolgende Auswertung mit den von Annif vorgeschlagenen zusätzlichen Einstellungen als hilfreich heraus-

gestellt hat, habe ich für eine bessere Vergleichbarkeit der Verfahren im Nachhinein auch noch einen Optimierungsdurchlauf mit dem Backend TF-IDF vorgenommen. Im Vergleich der Ergebnisse aller genutzten Backends nach der Optimierung zeigt sich, dass Omikuji bei allen Werten die besten Ergebnisse hervorbringt. Abbildung 4.22 zeigt die Ergebnisse im Vergleich.

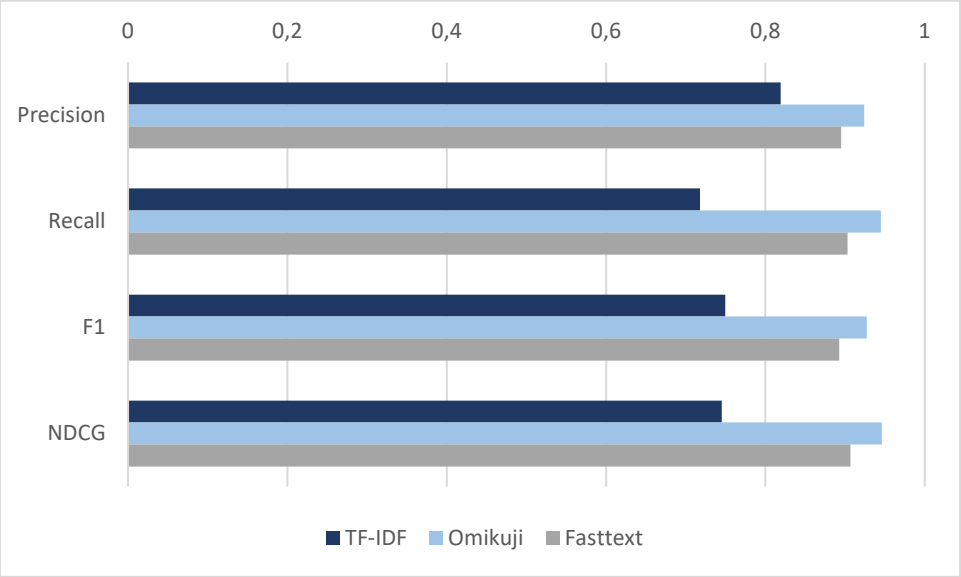

Abbildung 4.22: Vergleich Backends nach Optimierungsroutine, Analyzer Snowball, ein Durchlauf

Während TF-IDF einen guten Einstieg bietet, um das Programm kennen zu lernen und sich mit den Grundfunktionen vertraut zu machen, fällt es in der Effektivität hinter den anderen Verfahren zurück. Die Ergebnisse zeigen, dass man mit Omikuji für eine Vergabe von Labels, wie ich sie evaluiert habe, eine gute Wahl trifft. Die Ergebnisse mit Fasttext sind geringer, allerdings muss man bedenken, dass hier noch einige Einstellungen verändert werden können und ich nur die im Annif Wiki vorgeschlagenen Grundeinstellungen ausgewertet habe.

4.5 Sampling mit Volltexten

Die Effektivität der getesteten Backends soll zusätzlich durch Trainings mit Volltexten getestet werden. Hierfür wird ein Sampling genutzt. Die Zusammenstellung der Volltexte wird in Kapitel 3.3.4 erläutert. Wie dort bereits beschrieben, gibt es in der Kollektion der Volltexte, die für Trainings und Tests zusammengestellt wurden, jeweils pro Objekt nur ein zutreffendes Label, sodass das Limit in der Konfiguration von Vorneherein auf 1 gesetzt wurde.

Die Ergebnisse waren hier im Vergleich zu den vorherigen Tests nicht gleichermaßen vielversprechend. Die Trainings- und Testdaten wurden für diese Untersuchung per Zufallsprinzip vier Mal in Trainingsdaten und Testdaten aufgeteilt und die Trainings und Auswertungen mehrfach durchgeführt. Es hat sich herausgestellt, dass die Ergebnisse in allen Durchgängen zwar leicht abwichen, aber doch ähnlich genug waren, dass ich eine ungünstige Aufteilung der Daten in Trainings- und Testdaten als Grund ausschließen konnte. Für die Auswertung habe ich aus den Ergebnissen aller Durchläufe die arithmetischen Mittelwerte gebildet.

Für jedes Backend habe ich neben den Trainings und Auswertungen mit dem Analyzer Snowball außerdem Trainings und Auswertungen mit den Analyzern Simplemma und Spacy absolviert. Der Spacy Analyzer begrenzt die Anzahl der Zeichen pro Volltext auf 1.000.000. Wie ich in von einem Entwickler von Annif herausgefunden habe, lässt sich diese Einstellung nicht ändern (J. Inkinen, persönliche Kommunikation per E-Mail, 11.07.2023), sodass ich die ohnehin schon kleinen Kollektion an Volltexten weiter reduziert habe und am Ende mit insgesamt 498 Volltexten gearbeitet habe. Um die Ergebnisse besser vergleichen zu können, wurden alle Trainings mit der reduzierten Kollektion durchgeführt und ausgewertet. Durch meine Vorgehensweise bei der Zusammenstellung der Volltexte enthielt jeder Volltext nur ein Label. Somit wurde davon ausgegangen, dass diese Texte auch von den Fachpersonen jeweils nur ein Label erhalten haben.

Da bei einem Limit von einem Label die Werte für Precision und Recall und somit auch für F1 identisch sind, habe ich mich bei der Auswertung auf den F1-Score konzentriert. Die Ergebnisse sind in Abbildung 4.23 grafisch dargestellt.

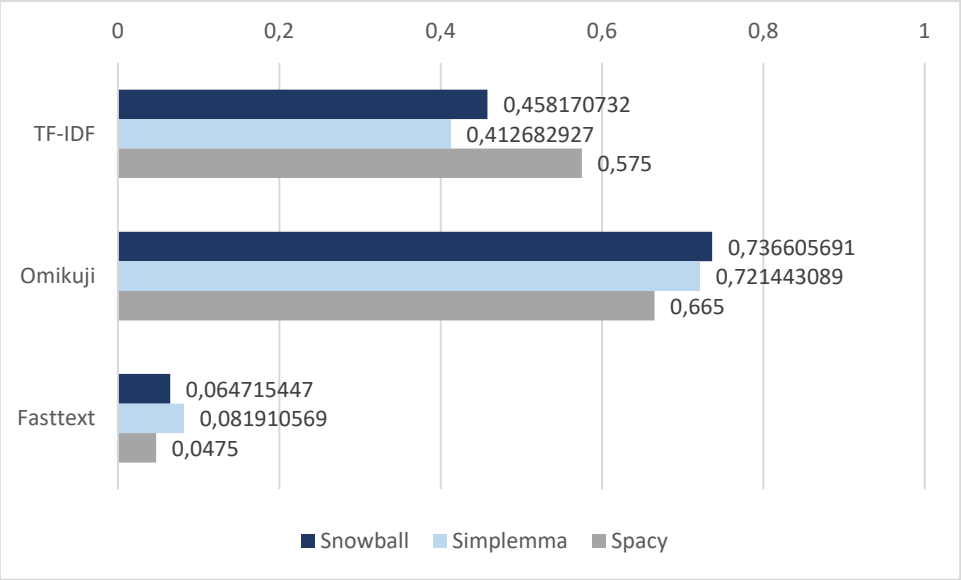

Abbildung 4.23: Vergleich F1-Score Backends und Analyzer, Volltexte, vier Durchläufe

Bei Betrachtung der Ergebnisse sind mehrere Beobachtungen auffällig. Alle Modelle, die unter Verwendung von Volltexten trainiert wurden, schneiden schlechter ab als mit den zuvor für die Trainings verwendeten Dokumenten, die aus Titeln und Abstracts bestanden. Dabei muss zum einen bedacht werden, dass bei den Volltexten nur mit der Mindestmenge an Vorkommen gearbeitet werden konnte. In der Gesamtmenge aller Volltexte, also Trainingsdaten und Testdaten zusammen, gab es zu jedem Label nur jeweils ca. 41 Volltexte, während bei den vorigen Tests jedes Label mit mindestens 50 Objekten verknüpft war, in der Realität hier aber viele Labels 100 oder mehr Dokumenten zugewiesen waren. Als Analyzer erreicht Snowball bei Omikuji die besten Ergebnisse, bei TF-IDF werden mit Spacy die besten Ergebnisse erzielt und für das Backend Fasttext wird nach der Vorverarbeitung durch Simplemma eine Verbesserung der Ergebnisse erzielt. Da die Effektivität, insbesondere mit dem Backend Fasttext, aber auch mit TF-IDF und Omikuji im Vergleich zu den Trainingsdaten, die aus Metadaten bestanden, insgesamt gering ist, würde man einen praktischen Einsatz von Annif mit diesem Setup nicht in Betracht ziehen bzw. zunächst weitere Tests mit größeren Mengen an Volltexten und Labels, die mit mehr Volltexten verknüpft sind, durchführen.

Ein weiterer Grund für die niedrigen F1-Scores könnte darin liegen, dass die genutzten Volltexte Abweichungen in den Daten enthalten können, die nicht zum Lernen des Modells beitragen. Solche Abweichungen werden in der Literatur auch als Rauschen bezeichnet (siehe z. B. Alpaydin, 2022, S. 34; Ertel, 2021, S. 220). Das können z. B. Abbildungen

sein, die das System nicht erkennt, oder auch Text, der nicht zum Inhalt beiträgt. Bei Volltexten von Monografien können dies allgemeine Informationen auf den ersten Seiten, die z. B. den Verlag beschreiben, oder etwas Allgemeines zu einer Reihe sein, in der der vorliegende Text enthalten ist. Genauso können am Ende des Textes andere Titel des Verlages o.ä. beschrieben werden. Auch könnte am Beginn des Textes noch eine Beschreibung in einer anderen Sprache gegeben sein, obwohl dies bei Monografien eher unüblich ist.

Zur Verringerung von Abweichungen bietet Annif zwei Möglichkeiten über die Option `transform`. Hiermit kann der Text angepasst werden (The National Library of Finland, 2023d). Mit der Angabe eines Wertes, der die maximale Anzahl an Zeichen angibt, die verarbeitet werden sollen, wird mögliches Rauschen am Ende des Textes abgeschnitten. Dies bei monografischen Texten anzuwenden ist schwierig, da die Länge der Texte sehr unterschiedlich sein kann. Zusätzlich bietet `transform` einen Sprachfilter, über den Sätze, die nicht der Sprache des Projekts entsprechen, herausgefiltert werden können. So würden z. B. Abstracts in anderen Sprachen eliminiert. Grundsätzlich eignen sich die Möglichkeiten, die `transform` bietet, eher für Volltexte wie Zeitschriftenartikel oder Hochschulschriften, die sich in ihren Strukturen und in der jeweiligen Länge ähneln. Bei monografischen Texten müssten verschiedene Einstellungen getestet und evaluiert werden, um zu sehen, ob man mit dieser Option eine Verbesserung erreichen kann.

Es fällt außerdem auf, dass das Backend Fasttext im Test mit den Volltexten sehr niedrige Werte erreicht hat. Um das genauer zu untersuchen, habe ich mir die Ergebnisse der einzelnen Labels angesehen und festgestellt, dass jeweils ein bestimmtes Label aus dem Vokabular für einen großen Teil der Dokumente vorgeschlagen wurde. Ich vermute, dass es hier zu einem so genannten Overfitting, also einer Überanpassung an die Trainingsdaten, gekommen sein könnte und dass Annif auf ein bestimmtes Label zurückfällt, wenn keine andere Zuordnung möglich scheint. Das Phänomen ist in allen vier Durchläufen aufgetreten, bezog sich jedoch in jedem Durchlauf auf ein anderes Label, was man noch genauer untersuchen müsste. Meine Vermutung, dass das Lernen mit Volltexten für das Backend Fasttext gut funktioniert, hat sich nicht bewahrheitet; das Gegenteil ist der Fall.

5 Fazit und Ausblick

Ziel dieser Masterarbeit war es, das Toolkit Annif zu untersuchen und durch Trainings und Tests verschiedener Backends mit unterschiedlich zusammengestellten Trainings- und Testdaten festzustellen, ob die Ergebnisse einen praktischen Einsatz in Bibliotheken rechtfertigen. Konkret sollte getestet werden, ob die Vorhersagen für Notationen des Klassifikationssystems RVK zuverlässig genug sind, um das Werkzeug zur Klassifizierung von deutschsprachigen Monografien einzusetzen. Gleichzeitig wurde untersucht, wie hoch der Aufwand ist, um einen effektiven Einsatz in einer Bibliothek zu gewährleisten, und welche Vorarbeiten notwendig sind. Getestet wurden die Backends TF-IDF, Omikuji und Fasttext. Alle getesteten Backends entsprechen dem assoziativen Ansatz, da sich bereits zu Beginn der Untersuchungen herausgestellt hat, dass ein lexikalischer Ansatz nicht geeignet ist, um Notationen der Regensburger Verbundklassifikation zu vergeben, da diese keine eindeutigen Bezeichnungen tragen. So gibt es z. B. mehrere RVK-Notationen mit der Benennung „Mathematische Modelle“ (SN 400, WH 2500) oder „Einzelne Systeme“ (AN 75200, ST 201, ST 601).

Es wurden für die Untersuchung mehrere Analyzer genutzt und ausgewertet sowie beispielhaft ein Training mit einer kleinen Auswahl an Volltexten durchgeführt und ausgewertet. Um herauszufinden, welche Rolle die Zusammensetzung der Trainingsdaten in Hinblick auf die Häufigkeit des Vorkommens einzelner Labels und auf das Vorhandensein von zusätzlichen Beschreibungen des Inhalts spielt, wurden Trainings und Tests mit verschieden zusammengestellten Daten durchgeführt und evaluiert.

Die durchgeführten Trainings und die Auswertungen der Tests haben gezeigt, dass es Unterschiede in der Effektivität der verschiedenen Backends und auch der Analyzer gibt. Die vorliegenden Trainings- und Testdaten haben sich aus mehreren Gründen als nur teilweise geeignet herausgestellt. Eine Schwierigkeit liegt darin, dass in den vorliegenden Trainings- und Testdaten pro Objekt in vielen Fällen nur ein Label vergeben war; die durchschnittliche Anzahl an Labels pro Objekt lag bei 1,3. Annif ist standardmäßig so eingestellt, dass 10 Vorschläge ausgegeben werden. Unter diesen Voraussetzungen ist eine niedrige Precision nachvollziehbar, da Annif gezwungenermaßen Labels vorschlägt, die unzutreffend sind, wodurch sich die Anzahl der False Positives erhöht, was den Wert der Precision verringert. In Bibliotheken können pro Ressource mehrere RVK-Notationen

vergeben werden. Häufig stammen diese jedoch aus verschiedenen Hauptgruppen, um die Interdisziplinarität von Texten hervorzuheben und Nutzenden die Möglichkeit zu geben, im Bibliothekskatalog Ressourcen zu ihrem Thema zu finden, die im Regal nicht unmittelbar nebeneinanderstehen. Da ich mich bei meinen Untersuchungen auf eine Hauptgruppe beschränkt habe, ließ sich dies in meinen Tests nicht abbilden. Im nächsten Schritt wäre es erforderlich, weitere Hauptgruppen der RVK in das Vokabular aufzunehmen und gezielt Trainingsdaten auszuwählen, in denen für einen Hauptteil der Objekte mehrere Labels vergeben wurden, die aus den verschiedenen Hauptgruppen stammen.

Eine weitere Rolle bei der Auswertung spielt die ungleiche Verteilung der Labels. Es hat sich gezeigt, dass Ergebnisse dadurch verbessert wurden, dass ich die Trainingsdaten auf Labels reduziert habe, die in mindestens 50 Objekten vorkommen. Damit wurden Labels ausgeschlossen, die von einer Bibliothekarin oder einem Bibliothekar bei der klassifikatorischen Sacherschließung mitberücksichtigt werden oder zumindest zur Verfügung stehen. Mit meiner Herangehensweise wird ein großer Teil der selten vorkommenden Notationen, also der Long Tail, ignoriert. Aus den 802 verschiedenen Labels, die in meinem Gesamtkorpus an Trainings- und Testdaten enthalten sind, kommen 560, also fast drei Viertel, in weniger als 50 Objekten vor. In Abbildung 5.1 ist die Verteilung veranschaulicht.

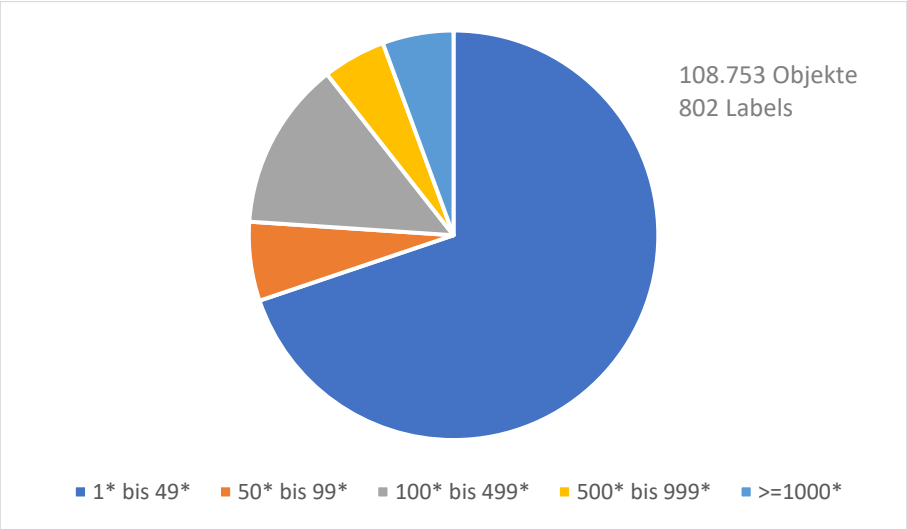

Abbildung 5.1: Vorkommen einzelner Labels in der Gesamtkollektion der Trainings- und Testdaten

Man kann argumentieren, dass Labels, die so selten vorkommen, im Bereich der Sacherschließung keine Rolle spielen. Die Frage ist, inwieweit man damit die RVK an sich

verändert und ob dann nicht genauso gut entgegengehalten werden könnte, dass die RVK nicht mehr das geeignetste Instrument für die Sacherschließung in Bibliotheken ist und man grundsätzlich auf andere, modernere Methoden wie z. B. das Tagging durch Nutzende zurückgreifen oder die RVK noch radikaler umschreiben sollte, als es momentan durch regelmäßige Anpassungen der Fall ist. Ein interessantes Experiment könnte auch darin bestehen, die Metadaten des Bestandes einer Bibliothek einem unüberwachten Lernverfahren zu übergeben. Beim so genannten Clustering werden dem System keine Zielwerte vorgegeben und es entscheidet selbst über eine Auswahl an Kategorien.

Will man Annif produktiv nutzen und dabei alle möglichen RVK-Notationen berücksichtigen, muss man eine Lösung finden, die bei der Auswahl der Daten für das Training ansetzt. Zum einen werden Titeldatensätze benötigt, die Abstracts enthalten. Zum anderen sollten die entsprechenden Ressourcen idealerweise mehrere RVK-Notationen enthalten und die jede einzelne RVK-Notation sollten an vielen Dokumenten vergeben sein. Es ist jedoch fraglich, ob sich geeignete Daten in der erforderlichen Menge beschaffen lassen. Wenn man bedenkt, dass in den von mir berücksichtigten 108.753 Dokumenten von 1.942 in der Hauptgruppe S verfügbaren RVK-Notationen nur 802 verwendet wurden, ist der Einwand berechtigt, dass Fachpersonen ebenfalls von vielen Notationen wenig oder keinen Gebrauch machen. Die Notationen sind möglicherweise zu spezifisch oder auch zu unspezifisch für den größten Teil der zu klassifizierenden Ressourcen oder zu ungeläufig bei dem Fachpersonal, das die Klassifizierungen vornimmt.

Es wurde durch ein Sampling gezeigt, dass ein Training mit Volltexten nicht zu Ergebnissen führt, die den beträchtlichen Aufwand, der notwendig wäre, um genügend Volltexte zu beschaffen, rechtfertigen. Verbesserte Ergebnisse könnten mit der Option `transform` erreicht werden, aber insgesamt lässt sich feststellen, dass Volltexte gerade im Bereich Monografien zu viele Abweichungen enthalten und Annif aus Metadaten effektivere Modelle kreiert. Darüber hinaus ist der Bedarf an Zeit und Speicherplatz für ein Training mit Volltexten hoch und führt beispielsweise mit dem Analyzer Spacy zu Konflikten. Da die Vergabe von RVK-Notationen sich in den meisten Bibliotheken auf monografische Ressourcen konzentriert, können Volltexte wie Zeitschriftenartikel oder Hochschulschriften nicht eingesetzt werden.

Was die Zusammenstellung der Trainingsdaten angeht, hat sich gezeigt, dass es nicht notwendig ist, um jeden Preis Volltexte zusammenzusuchen, um beste Ergebnisse zu erreichen. Bei der Nutzung von Metadaten sollte jedoch darauf geachtet werden, dass die beschreibenden Daten zumindest Abstracts oder Inhaltsverzeichnisse enthalten. Für eine detailliertere Beschreibung der Ressourcen wäre es denkbar, Schlagwörter, z. B. aus dem MARC-Feld 650, in die Trainingsdaten aufzunehmen. Um die Daten über Schnittstellen von Bibliotheksverbünden herunterzuladen und in das von Annif geforderte Format zu bringen, sind Programmierkenntnisse von Vorteil.

Die Ergebnisse, die von Annif erzielt werden, sind vielversprechend und ein Einsatz in Bibliotheken könnte sich durchaus lohnen. Ich würde Annif eher als Vorschlagswerkzeug empfehlen und es nicht vollautomatisiert einsetzen. Damit würde gewährleistet, dass als letzte Instanz eine Fachperson die Ergebnisse von Annif überprüft. Bei einem Einsatz muss eine Bibliothek sich der Schwierigkeiten, die ich bezüglich der RVK-Notationen und ihrer ungleichen Verteilung und Nutzung beschrieben habe, bewusst sein und einen Umgang damit finden. Insbesondere für kleine Bibliotheken empfiehlt es sich grundsätzlich, mit anderen zu kooperieren bzw. sich einem Verbund anzuschließen, um ein derartig aufwändiges Vorhaben nicht allein bewerkstelligen zu müssen.

Literaturverzeichnis

Alpaydin, E. (2022). *Maschinelles Lernen*. Berlin: De Gruyter Oldenbourg. https://doi.org/10.1515/9783110740196

American Library Association, Canadian Federation of Library Associations, & Chartered Institute of Library and Information Professionals. (2022). *Homepage | RDA Toolkit*. https://www.rdatoolkit.org/

Aust, H. (2021). *Das Zeitalter der Daten: Was Sie über Grundlagen, Algorithmen und Anwendungen wissen sollten*. Berlin: Springer. https://doi.org/10.1007/978-3-662-62336-7

Babbar, R., & Schölkopf, B. (2019). Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, *108*(8), 1329–1351. https://doi.org/10.1007/s10994-019-05791-5

Barbaresi, A. (2023). *Simplemma* [Python]. https://github.com/adbar/simplemma (Original work published 2021)

Beckmann, R., Hinrichs, I., Janßen, M., Milmeister, G., & Schäuble, P. (2019). Der Digitale Assistent DA-3: Eine Plattform für die Inhaltserschließung. *o-bib. Das offene Bibliotheksjournal*, *6*(3). https://doi.org/10.5282/o-bib/2019H3S1-20

Bibliotheksservice-Zentrum Baden-Württemberg. (2023). *K10plus—SWB - BSZ-WIKI*. https://wiki.bsz-bw.de/display/SWB/K10plus#Allgemeines

Biemann, C., Heyer, G., & Quasthoff, U. (2022). *Wissensrohstoff Text: Eine Einführung in das Text Mining*. Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-35969-0

Deutsche Nationalbibliothek. (2023). *Deutsche Nationalbibliografie*. https://www.dnb.de/nationalbibliografie

Ertel, W. (2021). *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-32075-1

Facebook Open Source. (2022). *List of options · fastText*. https://fasttext.cc/index.html

Frochte, J. (2020). *Maschinelles Lernen*. München: Hanser. https://doi.org/10.3139/9783446463554.fm

Gantert, K., & Hacker, R. (2016). *Bibliothekarisches Grundwissen*. Berlin: De Gruyter Saur.

Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol: O'Reilly.

Grandini, M., Bagli, E., & Visani, G. (2020). *Metrics for Multi-Class Classification: An Overview* (arXiv:2008.05756). arXiv. https://doi.org/10.48550/arXiv.2008.05756

Häusler, I., & Werr, N. (2018). Die Regensburger Verbundklassifikation (RVK). In H. Alex, G. Bee, & U. Junger (Hrsg.), *Klassifikationen in Bibliotheken: Theorie, Anwendung, Nutzen* (S. 127–163). Berlin: De Gruyter Saur.

Heidenreich, H. (2018). *Introduction to Word Embeddings*. https://towardsdatascience.com/introduction-to-word-embeddings-4cf857b12edc

Kähler, M. (2023). Inhaltliche Erschließung mit KI in der Deutschen Nationalbibliothek. *Deutsche Gesellschaft für Information & Wissen e.V.* https://dgi-info.de/inhaltliche-erschliessung-mit-ki-in-der-deutschen-nationalbibliothek/

Lanquillon, C. (2019). Grundzüge des maschinellen Lernens. In S. Schacht & C. Lanquillon (Hrsg.), *Blockchain und maschinelles Lernen: Wie das maschinelle Lernen und die Distributed-Ledger-Technologie voneinander profitieren* (S. 89–142). Berlin: Springer. https://doi.org/10.1007/978-3-662-60408-3_3

Lorenz, B. (2018). Zur Theorie und Terminologie der bibliothekarischen Klassifikation. In H. Alex, G. Bee, & U. Junger (Hrsg.), *Klassifikationen in Bibliotheken: Theorie, Anwendung, Nutzen* (S. 1–22). Berlin: De Gruyter Saur.

Mockenhaupt, A. (2021). *Digitalisierung und Künstliche Intelligenz in der Produktion: Grundlagen und Anwendung*. Wiesbaden: Springer Fachmedien. https://doi.org/10.1007/978-3-658-32773-6

Mödden, E. (2020). *KI im Einsatz für die inhaltliche Erschließung—Ein Erfahrungsbericht aus der Deutschen Nationalbibliothek*. Technische Informationsbibliothek (TIB),Berufsverband Information Bibliothek e. V. (BIB). https://doi.org/10.5446/36428

NLTK Project. (2023). *NLTK :: Natural Language Toolkit*. https://www.nltk.org/index.html

Regensburger Verbundklassifikation Online. (2023). *RVK Online*. https://rvk.uni-regensburg.de/regensburger-verbundklassifikation-online

Scheven, E., Nadj-Guttandin, J., & Arbeitsstelle für Standardisierung (Hrsg.). (2017). *Regeln für die Schlagwortkatalogisierung: RSWK*. Leipzig: Deutsche Nationalbibliothek.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47. https://doi.org/10.1145/505282.505283

Shmueli, B. (2020). Multi-Class Metrics Made Simple, Part I: Precision and Recall. https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2

Suominen, O. (2019). Annif: DIY automated subject indexing using multiple algorithms. *LIBER Quarterly: The Journal of the Association of European Research Libraries*, *29*(1), 1–25. https://doi.org/10.18352/lq.10285

Suominen, O., & Koskenniemi, I. (2022). Annif Analyzer Shootout: Comparing text lemmatization methods for automated subject indexing. *Code4Lib*, *54*(54). https://journal.code4lib.org/articles/16719

Suominen, O., Lehtinen, M., & Inkinen, J. (2022). Annif and Finto AI: Developing and implementing automated subject indexing. *JLIS*, *1*, 265–282. https://doi.org/10.4403/jlis.it-12740

Suominen, O., Lehtinen, M., Inkinen, J., Kasprzik, A., & Fürneisen, M. (2023). *Annif-tutorial*. https://github.com/NatLibFi/Annif-tutorial

The National Library of Finland. (2021). *Annif Wiki: Achieving good results*. https://github.com/NatLibFi/Annif/wiki/Achieving-good-results

The National Library of Finland. (2022a). *Annif Wiki: Backend Fasttext*. https://github.com/NatLibFi/Annif/wiki/Backend%3A-fastText

The National Library of Finland. (2022b). *Annif Wiki: System requirements*. https://github.com/NatLibFi/Annif/wiki/System-requirements

The National Library of Finland. (2023a). *Annif Wiki: Backend: Omikuji*. https://github.com/NatLibFi/Annif/wiki/Backend:-Omikuji

The National Library of Finland. (2023b). *Annif Wiki: Subject vocabulary formats*. https://github.com/NatLibFi/Annif/wiki/Subject-vocabulary-formats

The National Library of Finland. (2023c). *Annif Wiki: Transforms*. https://github.com/NatLibFi/Annif/wiki/Transforms

Toepfer, M., & Seifert, C. (2020). Fusion architectures for automatic subject indexing under concept drift. *International Journal on Digital Libraries*, *21*(2), 169–189. https://doi.org/10.1007/s00799-018-0240-3

University of Helsinki. (2017). *CLI commands—Annif 0.61.0 documentation*. CLI commands--annif 0.6.10. https://annif.readthedocs.io/en/stable/source/commands.html#annif-optimize

Abbildungsverzeichnis

Inhalt der CD

Ordner	Enthaltene Dateien	Beschreibung
Masterarbeit	Masterarbeit_Mecking_Juli_2023.pdf	Masterarbeit „Automatisierte Vergabe von Notationen der Regensburger Verbundklassifikation RVK mithilfe des Toolkits Annif“
Konfigurationsdatei	projects.cfg	Konfigurationen der Projekte
Code	convert_fulltext.ipynb	Programm, das pdf-Dateien in txt ändert und zu jeder Textdatei eine Schlüsseldatei erstellt
	transform_rvk.ipynb	Programm zum Umformatieren von xml nach tsv, konvertiert RVK-Notationen zu URIs
	download_transform_metadata.ipynb	Programm zum Erstellen eines Textkorpus durch Auslesen einer Schnittstelle
	split_test_training.ipynb	Programm zum Aufteilen von Metadaten in 80% Trainingsdaten und 20% Testdaten
	calculate_occurence.ipynb	Programm zur Berechnung, welche Notation in wie vielen Dokumenten vorkommt

Metadaten	korpus_gemischt_rvk_s.tsv	Dokumente mit und ohne Abstracts
	korpus_mit_abstract_rvk_s.tsv	Dokumente, die alle Abstracts enthalten
	korpus_mit_abstract_50plus.tsv	Dokumente mit Notationen, die 50-mal oder öfter vorkommen
	rvko_2023_1.xml	RVK in xml, Stand Januar 23
	Gesamtkorpus_Volltexte (eigener Ordner)	Volltexte als Textdateien mit dazugehörigen Schlüsseldateien
Auswertungsdateien	Training_mit_Volltexten (eigener Ordner)	Enthält Auswertungsdateien aus den Trainings mit Volltexten als Trainings- und Testdaten
	Fasttext (eigener Ordner)	Enthält Auswertungsdateien für das Backend Fasttext
	Omikuji (eigener Ordner)	Enthält Auswertungsdateien für das Backend Omikuji
	TF_IDF (eigener Ordner)	Enthält Auswertungsdateien für das Backend TF-IDF

Selbständigkeitserklärung

Ich versichere, dass die vorliegende Arbeit von mir selbständig und ohne unerlaubte Hilfe angefertigt worden ist. Ich habe alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, durch Zitate bzw. Literaturhinweise als solche kenntlich gemacht.

Hoppegarten, 24.07.2023

Elisabeth Mecking