THM | CAMPUS FRIEDBERG | MND
TECHNISCHE HOCHSCHULE MITTELHESSEN
Mathematik, Naturwissenschaften und Datenverarbeitung

Masterthesis

Konzeption, Entwicklung und Analyse eines Tools zur automatisierten Qualitätskontrolle wissenschaftlicher Arbeiten

Zur Erlangung des akademischen Grades

Master of Science

Eingereicht im Fachbereich Mathematik, Naturwissenschaften und Datenverarbeitung an der Technischen Hochschule Mittelhessen

Vorgelegt von

Johann Decher

18.08.2025

Referent: Prof. Dr. habil. Markus Siepermann

Korreferent: Prof. Dr. Frank Kammer

Abstract

Das Verfassen wissenschaftlicher Arbeiten stellt für viele Studierende eine Herausforderung dar. Besonders das Einhalten formaler Vorgaben und die Sicherstellung sprachlicher Qualität bereiten häufig Schwierigkeiten, obwohl entsprechende Lehrveranstaltungen formale Vorgehensweisen vermitteln. In dieser Arbeit wird das Prüftool SciCheck vorgestellt und auf Konzepte, Entwicklung und Analyse des Tools eingegangen. SciCheck soll Studierende beim Schreibprozess mit automatisierter Qualitätskontrolle unterstützen. Es werden grundlegende Anforderungen wissenschaftlicher Arbeiten erläutert und das bestehende System sowie die umgesetzten Erweiterungskonzepte vorgestellt. Ein Testlauf des Systems in einem Seminar lieferte Informationen über Schwächen SciChecks, die identifiziert und mit den neuen Erweiterungen gezielt adressiert wurden. Die technische Umsetzung der neuen Dokumentvorverarbeitung, Sprachanalyse, Einbindung von Word-Kommentaren und Plagiatsindikatoren wird detailliert beschrieben. Darüber hinaus wird in einem Vergleich verschiedener moderner Prüftools festgestellt, welche Funktionalitäten bereitgestellt werden und welche Anforderungen wissenschaftlicher Arbeiten abgedeckt werden. Es wird gezeigt, dass SciCheck durch die neuen Funktionalitäten ein nützliches Tool für Studierende darstellt und nicht nur verlässlichere, sondern auch transparente und nachvollziehbare Ergebnisse liefert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

ASL	Average Sentence Length
ASW	Average Syllables per Word
AWE	Automatic Writing Evaluation
BETSY	Bayesian Essay Test Scoring sYstem
FRE	Flesch-Reading-Ease
GenAI	Generative Artificial Intelligence
IEA	Intelligent Essay Assessor
JSON	JavaScript Object Notation
LSA	Latent Semantic Analysis
NLP	Natural Language Processing
PEG	Project Essay Grade
XML	Extensible Markup Language

1 Einleitung

1.1 Motivation

Die Gestaltung wissenschaftlicher Arbeiten stellt für Studierende eine erhebliche Herausforderung dar. Trotz gezielter Lehrveranstaltungen, in denen die nötigen Fähigkeiten zur Erstellung einer Arbeit frei von Formfehler erlernt werden sollen, kommt es häufig zu Abgaben dessen Formgestaltung optimiert werden kann. Herausforderungen reichen von korrekter Zitierweise bis hin zur Einhaltung struktureller, formativer und sprachlicher Vorgaben. Diese Anforderungen können sich für Studierende insbesondere in den frühen Phasen ihres akademischen Werdegangs als anspruchsvoll erweisen (Heinze et al., 2009, S. 12). Dazu kommt, dass unerfahrene Autoren wissenschaftlicher Arbeiten bereits eine kognitive Überlastung durch komplexe Themen und mentale Barrieren erfahren können (Pietrobon et al., 2009, S. 6). Studierende haben wenig praktische Erfahrung und haben nach Pietrobon Schwierigkeiten ihre Arbeit sinnvoll zu strukturieren und den Inhalt fassbar zu vermitteln. Sich überdies noch mit den formalen und sprachlichen Anforderungen zu beschäftigen, kann zu Fehlern und schwer lesbaren Arbeiten führen.

Gerade bei großen Gruppen von Studierenden ist eine individuelle Betreuung schwer möglich. Die Studie der Universität Augsburg (Heinze et al., 2009, S. 12-13) zeigt, dass besonders Studierende niedriger Semester anonym Rat bei Tutoren oder per E-Mail bezüglich Formalia und Schreibweise suchen. Obwohl diese Nachfrage bei Studierenden höherer Semester stark sinkt, ist nicht davon auszugehen, dass deren wissenschaftliche Arbeiten frei von Formfehlern oder minderer Sprache sind. Gerade wenn Studierende trotz formalen Fehlern Prüfungsleistungen bestehen, kann der Eindruck entstehen, dass solche Fehler keine Konsequenzen haben, weshalb sie im späteren Verlauf ihres Studiums weniger nach Unterstützung fragen.

Aus diesem Grund wurde im Rahmen eines Lehr-Innovations-Projekts an der Technischen Hochschule Mittelhessen das Tool SciCheck entwickelt. SciCheck soll als Begleitung für Studierende beim Verfassen wissenschaftlicher Arbeiten fungieren. Studierende werden durch SciCheck nicht nur auf grobe Formfehler aufmerksam gemacht, sondern auch auf Details, die schnell untergehen oder denen sie sich zuvor nicht bewusst waren. Die Integration eines Tools zur automatisierten Überprüfung formaler Anforderungen in Seminaren bringt Studierende zwangsläufig dazu, sich näher mit den Anforderungen zu beschäftigen. Im Sommersemester 2024 wurde SciCheck in einem

Seminar an der Technischen Hochschule Mittelhessen erstmals getestet. Zwar wurde das Tool von Studierenden gut angenommen, jedoch zeigte die dazugehörige Evaluierung deutlich, dass SciCheck noch Verbesserungsbedarf aufweist. Im Rahmen dieser Arbeit sollen neue Anforderungen implementiert und evaluiert werden, damit SciCheck einen größeren Mehrwert für Studierende und Dozierende darstellt.

1.2 Zielsetzung

Aufbauend auf den Erkenntnissen der ersten Testphase SciChecks und weiteren Anforderungen, sollen neue Funktionen implementiert und die bisherigen Funktionen verbessert werden. Dadurch soll eine breitere Reihe an Kriterien geprüft werden und die Qualität der wissenschaftlichen Arbeiten gesteigert werden. Darüber hinaus soll das gegebene Feedback verbessert werden, damit Studierende ihre Fehler genauer einsehen und nachvollziehen können. Fehler sollen somit problemlos behoben werden können.

Zusätzlich soll nicht nur ein Überblick über SciCheck gegeben werden, sondern auch über andere bereits vorhandene Tools zur automatisierten Überprüfung wissenschaftlicher Arbeiten. Dabei gilt es auch auf die verschiedenen Anforderungen an wissenschaftlichen Arbeiten einzugehen, da es für unterschiedliche Anforderungen unterschiedliche Tools zur Unterstützung der Erfüllung dieser gibt.

1.3 Aufbau der Arbeit

Diese Arbeit ist in mehrere Abschnitte aufgeteilt, die sich mit der Einordnung, Entwicklung und Evaluation des Tools beschäftigen. Zu Beginn wird der Projektkontext definiert. Es wird sich der Frage gewidmet, welche Anforderungen zu wissenschaftlichen Arbeiten gehören. Dabei wird ein Überblick über das bestehende System gegeben, um ein Verständnis für die Rahmenbedingen und Notwendigkeiten der Erweiterungen zu schaffen (Kapitel 2). Darauf folgt eine Übersicht über bestehende Tools, um SciCheck besser einordnen zu können. Zum einen werden Anwendungen betrachtet, die nur nach Bezahlung zugänglich sind und zum anderen Lösungen aus wissenschaftlichen Publikationen (Kapitel 3). Anschließend wird die technische Umsetzung der neuen Funktionen detailliert beschrieben. Dazu gehört die Vorverarbeitung von Dokumenten, das automatische Einfügen von Kommentaren in Word, die Analyse der Sprache und die Plagiatshinweise. Diese Schritte werden ausführlich behandelt, damit die Funktionsweise des Tools verdeutlicht werden kann (Kapitel 4). Abschließend wird das Nutzerfeedback diskutiert und die Funktionalitäten verschiedener Prüftools verglichen (Kapitel 5), gefolgt

von einer Zusammenfassung der Ergebnisse und einem Ausblick, wie zukünftige Erweiterungen aussehen könnten (Kapitel 6).

2 Projektkontext

2.1 Anforderungen an wissenschaftlichen Arbeiten

Wissenschaftliche Arbeiten gelten als Fundament des Erkenntnisgewinns und der Wissensverbreitung in der akademischen Welt. Das primäre Ziel einer wissenschaftlichen Arbeit ist es, einen Beitrag zum bestehenden Wissensstand zu leisten, neue Erkenntnisse zu generieren, bestehende Theorien zu überprüfen oder eigene Ergebnisse in den wissenschaftlichen Diskurs einzuordnen. Für Studierende ist der Beitrag neuer Ideen zum wissenschaftlichen Diskurs im Laufe des Studiums nicht verlangt, denn das korrekte wissenschaftliche Forschen und Arbeiten, sowie notwendige fachliche Kenntnisse müssen noch erlernt werden. In jedem Fall wird sich systematisch mit einem Forschungsgegenstand beschäftigt. Dieser muss gegebenenfalls zuerst gefunden und ausgewählt werden. Selbst die Suche nach dem richtigen Thema kann eine Herausforderung sein.

Wurde ein passendes Thema gefunden, gilt es zahlreiche Anforderungen verschiedener Kategorien einzuhalten. Aus den Leitfäden zum wissenschaftlichen Arbeiten von Universitäten und Hochschulen und aus Ergebnissen der Literatur wurden inhaltliche und zielorientierte Anforderungen (Bänsch & Alewell, 2020, S. 3; Bennewitz et al., 2022, S. 4; Haines, 2020, S. 1; Wilbers & Kimmelmann, 2024, S. 4), methodische Anforderungen (Bänsch & Alewell, 2020, S. 5, 14; Katholische Universität Eichstätt-Ingolstadt, 2017, S. 6-7; Ritschl et al., 2016, S. 22; Warburg, 2021, S. 18) und formale und stilistische Anforderungen (Bänsch & Alewell, 2020, S. 19-27, 131; Haines, 2020, S. 26-28; Katholische Universität Eichstätt-Ingolstadt, 2017, S. 9; Wytrzens et al., 2012, S. 119-121) gesichtet.

1 Inhaltliche und zielorientierte Anforderungen:

- Erkenntnisgewinn und Lösungsansatz: Es soll für die Gesellschaft relevantes Wissen geschaffen werden und der Stand der Forschung erweitert bzw. vertieft werden. Gegebenenfalls wird ein eigener Lösungsansatz oder in empirischen Arbeiten eine eigene Untersuchung inklusive Auswertung erwartet.
- Problemorientierung: Erkenntnisse sollen nicht nur aufgelistet werden, sondern zielgerecht und systematisch einer Fragestellung folgen, um neue Erkenntnisse hervorzubringen. Eine noch offene oder unbeantwortete Frage soll beantwortet werden.

- Fachkenntnis: Beim Schreiben müssen Autoren über die notwendigen Fachkenntnisse verfügen, um Inhalte sicher zu bearbeiten und zu vermitteln.
- Kritische Auseinandersetzung: Im gleichen Zuge muss das erlangte Wissen kritisch und aus verschiedenen Perspektiven betrachtet werden, sodass eine kritische Diskussion geführt werden kann. Eigene Erkenntnisse und Erkenntnisse auf dem nationalen und internationalen Stand der Forschung gilt es zu berücksichtigen und kritisch abzuwägen.
- Eigenleistung und Gedankenführung: Während der Einbezug bestehenden Wissens gefragt ist, muss dennoch ein gewisser Eigenanteil zur Bearbeitung einer Forschungsfrage geleistet werden.
- Überzeugende Argumentation: In wissenschaftlichen Arbeiten sollen Leser durch schlüssige Beweisführung bzw. logische Argumentationsstruktur überzeugt werden.

2 Methodische Anforderungen:

- Systematik und rationale Methoden: Um Erkenntnisse zu erlangen, soll beim wissenschaftlichen Arbeiten systematisch und rational vorgegangen werden. Techniken und Regeln des wissenschaftlichen Arbeitens sind für den Prozess wichtig, um Erkenntnisziele zu erreichen und sollen stets eingehalten werden.
- Präzision: Neben sprachlicher Genauigkeit, soll die Argumentation logisch aufgebaut und widerspruchsfrei sein.
- Intersubjektivität und Begründbarkeit: Komplexe Sachverhalte müssen nachvollziehbar dargestellt werden, indem der Weg zu einem Ergebnis oder einer Erkenntnis logisch und argumentativ aufgeführt wird. Aufgestellte Hypothesen sollen begründet, überprüfbar und intersubjektiv eindeutig sein.
- Zuverlässigkeit: Verwendete Quellen müssen offengelegt werden und es soll eine klare Unterscheidung und Klassifizierung zwischen Tatsachen, Wertungen und Vermutungen geben.
- Datenerhebung: Sollen im Rahmen einer wissenschaftlichen Arbeit Daten gesammelt und analysiert werden, kann dies durch z. B. Interviews, Tests, Befragungen oder Beobachtungen erreicht werden.

3 Formale und stilistische Anforderungen

- Sachlichkeit: Wissenschaftliches Arbeiten zeichnet sich durch Verwendung eines neutralen Sachstils aus.

- Sprachliche Korrektheit: Der geschriebene Text sollte keine Grammatik-, Rechtschreib- Zeichensetzungs- oder andere sprachliche Fehler aufweisen. Zu viele sprachliche Fehler können sich negativ auf die Bewertung auswirken.
- Sprachliche Gestaltung: Wissenschaftliche Arbeiten sollen verständlich, nachvollziehbar und zielgruppengerecht formuliert werden, ohne auf die Nutzung des entsprechenden Fachvokabular zu verzichten. Dabei ist eine präzise und ausdrucksvolle Sprache erwünscht. Das Verwenden von Umgangssprache und Füllwörtern soll vermieden werden. Auch bei Arbeiten mit mehreren Autoren ist auf einen einheitlichen Schreibstil zu achten.
- Struktur: Neben einer sinnvollen inhaltlichen Aufteilung, muss bei der Struktur auf formale Aspekte geachtet werden. Jedem ersten Punkt bei Überschriften oder Aufzählungen muss ein Zweiter folgen. Wissenschaftliche Arbeiten sollen ein Deckblatt und ein Inhaltsverzeichnis beinhalten, wenn nötig gefolgt von weiteren Verzeichnissen wie Abkürzungs-, Abbildungs- oder Tabellenverzeichnis. Nach dem Textteil bestehend aus Einleitung, Hauptteil und Schluss folgt das Literaturverzeichnis und optional der Anhang. Zum Schluss soll eine Ehrenwörtliche bzw. Eidesstattliche Versicherung unterzeichnet vorzufinden sein.
- Korrekte Zitation und Quellenangabe: Literatur soll einheitlich und korrekt nach einem Zitierstil zitiert werden. Übernommenes geistiges Eigentum muss kenntlich gemacht werden. Alle Zitationen brauchen einen entsprechenden Eintrag im Literaturverzeichnis und jede dort aufgeführte Quelle soll mindestens einmal verwendet werden.
- Layout: Der erste Eindruck prägt die Einstellung zur gesamten Arbeit. Vorgaben zu Schriftart, Schriftgröße, Seitenrand, Textausrichtung und Zeilenabstand sind stets einzuhalten. Auch Überschriften, Fußnoten oder Beschriftungen von Abbildungen sind korrekt und einheitlich zu formatieren.

Die aufgeführten Anforderungen an wissenschaftliche Arbeiten und auch an Studierende zeigen, dass beim wissenschaftlichen Schreiben zahlreiche Aspekte beachtet werden müssen. Neben sauberer Literaturarbeit, sinnvollem Aufteilen der Kapitel und korrekter Form und dem gründlichen Korrekturlesen gibt es weitere potenzielle Hindernisse wie psychische Schwierigkeiten oder Zeitmanagement. Versagensängste, Selbstzweifel, Überforderung oder Schreibblockaden können das Verfassen einer wissenschaftlichen Arbeit für Studierende erschweren. Dabei spielt das Zeitmanagement eine entscheidende Rolle, um Stress zu vermeiden.

Studierende haben die Möglichkeit sich bei einigen Anforderungen durch automatisierte Prüftools oder Schreibassistenten unterstützen zu lassen. Dabei ist zu beachten, dass kein System die eigene Leistung ersetzen kann oder soll. Die Hauptlast liegt weiterhin auf den Studierenden und automatisierte Prüftools dienen lediglich als Unterstützung.

2.2 Überblick über das bestehende System

SciCheck besteht aus zwei Hauptkomponenten: dem Frontend, das mit Flutter entwickelt wurde und dem Backend, das mittels Python implementiert wurde. Relevante Informationen und Funktionen werden im Frontend angezeigt, während im Backend Analysen und die Beschaffung persistenter Daten durchgeführt wird. Zur persistenten Speicherung der Daten wird die NoSQL-Datenbank MongoDB eingesetzt. Diese Komponenten sind modular aufgebaut und mittels Docker containerisiert, damit sie auf dem Server des Feedbacksystems der Technischen Hochschule Mittelhessen aufrufbar sind. Anwender können SciCheck im Internet unter *https://feedback.mni.thm.de/sci-check* erreichen.

Auf der Benutzeroberfläche SciChecks haben Nutzer verschiedene Optionen: Im mittleren Bildschirmabschnitt werden sie aufgefordert eine Word-Datei hochzuladen. Wenn nach dem Hochladen einer Word-Datei auf den Button *Analysiere Datei* geklickt wird, startet die automatische Überprüfung. Dafür sendet Flutter eine HTTP-POST-Anfrage an einen Endpunkt im Python Backend. Dieser Anfrage kann außerdem eine individuell definierbare Konfigurationsdatei angehängt werden.

Anwender können neben einer vordefinierten Standardkonfiguration entweder selbst definierte Konfigurationen wählen oder wenn sie Teil eines Kurses im Feedbacksystem sind, auf die von Dozierenden des entsprechenden Kurses erstellten Konfigurationen zugreifen. Konfigurationen können durch Klicken auf das Menü im oberen linken Bildschirmrand unter dem Reiter Konfiguration definiert werden.

Der Konfigurationsbildschirm ist in verschiedene Abschnitte untergliedert. Im oberen Abschnitt kann ein Name für eine neue oder bestehende Konfiguration festgelegt werden. Auch können die genauen Einstellungen zuvor erstellter Konfigurationen eingesehen werden. Hier haben Nutzer die Möglichkeit Konfigurationen zu löschen, wenn sie die nötigen Berechtigungen besitzen. Der Hauptteil des Konfigurationsbildschirms ist in die Kategorien eingeteilt, die SciCheck derzeit überprüft. Dazu zählen Formalia, Struktur und Zitation.

Im untern Bildschirmabschnitt können Dozierende auswählen, ob die erstellte Konfiguration für sie selbst oder für einen bestimmten Kurs angelegt werden soll. Durch Klicken auf den *Speichern*-Button werden alle getroffenen Einstellungen in der MongoDB hinterlegt.

Hat das Backend die Bytes der hochgeladenen Word-Datei und die ausgewählten Konfigurationsdaten erhalten, beginnt die Überprüfung der Datei. Nacheinander werden Funktionen zum Prüfen der verschiedenen Kategorien ausgeführt und die Ergebnisse gespeichert. Nach erfolgreicher Prüfung sendet das Backend eine Antwort an das Frontend mit den Ergebnissen der Analyse. Die Ergebnisse der Analyse werden auf dem Ergebnisbildschirm im Frontend aufbereitet. In tabellarischer Form werden die Ergebnisse aller Kriterien in ihren jeweiligen Kategorien angezeigt. Nutzer bekommen Feedback, durch einen Button in der rechten Spalte der Tabelle. Der Button gibt durch Text und farbliche Markierung an, ob ein Kriterium richtig behandelt worden ist. Bei gefundenen Fehlern können Nutzer auf den Button klicken, um ein Pop-up-Fenster zu öffnen, dass ihnen nähere Informationen liefert. Darunter fällt z. B. eine Erläuterung welche inkorrekte Schriftart oder Schriftarten verwendet worden sind. Nicht erkennbar ist jedoch, an welcher Stelle im Text sich der gefundene Fehler bzw. die gefundenen Fehler befinden. Als Hilfe zur Behebung der Fehler können Nutzer auf den Namen jedes Kriteriums klicken und Hinweise erhalten, wie dieses Kriterium korrekt umzusetzen ist.

2.3 Erweiterungskonzepte

SciCheck liefert eine benutzerfreundliche Oberfläche, in der Nutzer problemlos durch wenige Klicks eine Analyse starten und ihre Ergebnisse einsehen können. Ein Problem bei der Darstellung der gefundenen Fehler ist jedoch, dass diese teilweise schwer zu finden sind. Besonders wenn nur einzelne Wörter oder sogar Zeichen falsch formatiert sind, kann es sich als umständlich erweisen, eine wissenschaftliche Arbeit nach diesen kleinen Fehlern zu durchsuchen. Selbst wenn das relevante Kapitel mit Absatz oder sogar der Satz, in dem der Fehler auftritt, gegeben ist, stellt sich eine Suche in der Datei als unpraktisch dar. Außerdem wirkt diese Darstellung auf der Benutzeroberfläche bei Abgaben mit zahlreichen gefundenen Fehlern schnell überwältigend. Manuelles Durchsuchen des Dokuments sorgt für zusätzlichen Aufwand bei der Fehlerkorrektur. Um dieses Phänomen zu vermeiden, soll eine neue Form der Fehlerlokalisierung und Darstellung implementiert werden. Nutzer sollen die Möglichkeit bekommen auf dem Ergebnisbildschirm eine kommentierte Word-Datei herunterzuladen, in der gefundene

Fehler automatisch mit Hilfe von Kommentaren in Word lokalisiert und erläutert werden. Die Kommentarfunktion bietet sich für die Lösung dieses Problems an, da Fehler der Reihe nach durchlaufen und verbessert werden könne. Der Text in Kommentaren kann Fehler näher erläutern. Beim Erstellen der kommentierten Version sollen entweder Kommentare für alle geprüften Kriterien eingefügt werden oder Nutzer wählen gezielt aus, für welche Kriterien sie Hinweise im Dokument erhalten möchten. Eine gezielte Auswahl soll erlauben, sich zunächst auf wenige spezifische Fehler zu konzentrieren, ohne das Dokument mit Hinweisen zu überladen. Die kommentierte Word-Datei soll die bestehende tabellarische Ergebnisübersicht ergänzen und Studierenden ihre Fehler gezielt verbessern lassen.

Eine weitere wesentliche Verbesserung, die am bestehenden System vorgenommen werden soll, ist die systematische Vorverarbeitung des hochgeladenen Dokuments vor Beginn der eigentlichen Analyse. Durch die Aufteilung des Dokuments in bestimmte Textabschnitte für die jeweiligen Prüffunktionen, sollen potenzielle Fehler bei der Analyse vermieden werden und die Prüfgenauigkeit deutlich erhöht werden. In der bisherigen Architektur wird teilweise das gesamte Dokument durchlaufen oder innerhalb einzelner Prüffunktionen eine individuelle Kürzung durchgeführt. Dies soll redundante und fehleranfällige Verfahren im Analyseprozess durch einheitliche Vorverarbeitung vermeiden. Es sollen Abschnitte wie Deckblatt, eidesstattliche Versicherung, Inhaltsverzeichnis oder andere Verzeichnisse entfernt werden. Besonders auf dem Deckblatt wurden zuvor häufig falsch positive Ergebnisse vermerkt, wenn z. B. zu große Schriftgrößen im Stil des Fließtexts oder sogar Abbildungen gefunden wurden. Auch soll das Literaturverzeichnis separat als eigener Abschnitt extrahiert werden. So lassen sich Prüfungen bezüglich der Zitation einfacher durchführen und nicht jede Funktion das Literaturverzeichnis selbst entfernen. Allgemein dient die strukturierte Vorverarbeitung dazu bei, SciCheck effizienter und präziser bei der Analyse zu machen. Auch ist es leichter nachzuvollziehen welche Funktionen welche Textabschnitte behandeln.

Derzeit beschäftigt sich SciCheck mit der automatisierten Überprüfung der Kategorien Formalia, Struktur und Zitation. Eine zentrale Eigenschaft, die sich durch automatisierte Kontrollen überprüfen lassen kann, ist die sprachliche Qualität des Textes. Diese ist ein zentraler Punkt bei wissenschaftlichen Arbeiten und kann eine inhaltlich gute Arbeit frustrierend zu lesen machen. Denn die sprachliche Qualität ist besonders relevant für die Verständlichkeit, Professionalität und Genauigkeit einer wissenschaftlichen Arbeit. Es gilt verschiedene neue Kriterien im Bereich Sprache zu prüfen. Dazu gehört grundlegend

die Rechtschreibung und Grammatik einer Arbeit. Im gleichen Zug sollen stilistische Auffälligkeiten wie z. B. ein übermäßiger Gebrauch gleichen Vokabulars oder korrekte, aber unterschiedliche Schreibweisen eines Wortes gefunden werden.

Ein weiterer Bestandteil der Analyse der Sprache soll die Analyse der Satzkomplexität sein. Dabei sollen Merkmale wie Satzlänge und Silbenlänge genutzt werden, um Einschätzungen zur Lesbarkeit zu treffen. Mit Lesbarkeitskennzahlen soll Studierenden geholfen werden zu verstehen, wie verständlich ihr Text ist. Wichtig dabei ist ergänzend die Analyse der Nebensätze, um Schachtelsätze und deren Komplexität festzustellen.

In wissenschaftlichen Arbeiten sollen präzise und sachliche Ausdrücke verwendet werden. Nicht wissenschaftliche Wörter wie Füllwörter, subjektive Wertungen, unpräzise Ausdrücke oder umgangssprachliche Begriffe sollen systematisch erkannt und gekennzeichnet werden. Eine Analyse dieser Kategorien soll Studierenden dabei helfen, ihre Ausdrucksweise gezielt zu verbessern. Dabei ist es wichtig, durch Alternativvorschläge oder Hinweise Feedback zu geben, dass zum Kontext des jeweiligen Satzes passt. Durch das Abdecken dieser Kriterien soll nicht nur die Qualität einer einzelnen Abgabe verbessert werden, sondern auch ein Lerneffekt für zukünftige Arbeiten entstehen.

Eine weitere neue Methode, um größeren Nutzen für Dozierende zu erzielen und wissenschaftliche Korrektheit zu bewahren, ist die Einführung von Plagiatshinweisen. Dabei soll es nicht direkt um eine klassische Plagiatserkennung gehen, sondern es sollen durch formale, zitationsbezogene oder sprachliche Uneinheitlichkeit Hinweise auf potenzielle Plagiate gegeben werden. Um diese Plagiatsindikatoren zu finden, sollen die bereits vorhandenen Überprüfungen genutzt werden. Als Hinweis auf potenziell plagiierte Inhalte sollen auffällige Formatierungswechsel in bestimmten Kapiteln oder Absätzen gelten. Auch wenn der sprachliche Stil eines Textes sich in einigen Absätzen stark von anderen unterscheidet, soll dies notiert werden. Es soll kein garantierter Plagiatsnachweis erbracht werden, sondern ein Hinweis für Dozierende, um an diesen Stellen den Inhalt näher zu prüfen. Danach können andere Mechanismen zur Bestätigung oder Dementierung des Plagiatshinweises durchgeführt werden.

3 Entwicklung und Landschaft automatisierter Prüftools

3.1 Anfänge automatisierter Prüftools

In der Literatur wird Software zur automatisierten Analyse von Texten als Automatic Writing Evaluation (AWE), Automatic Essay Evaluation oder Automated Essay Scoring bezeichnet. Der Ursprung dieser Tools kann bis in die 1960er Jahre zurückgeführt werden, als der pädagogische Psychologe Ellis Page das Bewerten von Aufsätzen mit Computer diskutierte (Page, 1966, S. 239-242) und seine Arbeit am sogenannten Project Essay Grade (PEG) veröffentlichte (Page, 1968). Dabei handelte es sich um eine Bewertung basierend auf strukturellen und sprachlichen Eigenschaften. Aus einem Datensatz von manuell bewerteten Arbeiten wurden Merkmale wie Textlänge, Anzahl Präpositionen, Anzahl Relativpronomen, Grammatik usw. gesammelt. Mit diesem statistischen Ansatz konnte durch lineare Regression ein Model zur Bewertung neuer Arbeiten entwickelt werden. Page gilt somit als Vorreiter der automatisierten Bewertungstools, wobei die Entwicklung dieser erst kosteneffektiv wurde, als Rechenleistung in den 1990er Jahren kostengünstiger und allgegenwärtig wurde.

Während Page sich auf das reine Bewerten von Aufsätzen fokussierte stellten Autoren wie Foltz et al. (1999) mit Intelligent Essay Assessor (IEA) erste Tools vor, die Feedback geben konnten. Außerdem wurde nicht nach strukturellen und sprachlichen Merkmalen geprüft, sondern es wurde der Inhalt einer Arbeit überprüft, indem Ideen und Konzepte unabhängig der spezifischen Wortwahl gemessen wurden. Dies wurde mittels Latent Semantic Analysis (LSA) bzw. Latent Semantic Indexing ermöglicht (Deerwester et al., 1990). LSA versucht abzubilden, dass unterschiedliche Wörter ähnliche bis gleiche Bedeutungen (Synonymie) haben können, während gleiche Wörter unterschiedliche Sachverhalte bezeichnen können (Polysemie). Es wird eine Matrix erstellt, in der Wörter in Zeilen, der Kontext, d. h. Absätze oder Dokumente in Spalten eingeteilt werden, während Einträge die Frequenz eines Wortes darstellen. Demnach ergibt sich bei z. B. 300 Absätzen und 2000 Wörtern eine Matrix von 300x2000 (Dikli, 2006, S. 7). Durch Singular Value Decomposition (Singulärwertzerlegung) kann die Dimension der Daten reduziert werden, wobei die Beziehungen zwischen Begriffen und dem Kontext erhalten bleiben. So können übergeordnete Konzepte und Assoziationen verschiedener Begriffe erkannt werden, da Begriffe und Texte als Vektoren dargestellt werden. Limitierend ist auch hier die Tatsache, dass vorerst ein Training mit umfassenden und gelabelten Datensätzen durchgeführt werden muss. Ein nicht bewerteter Text kann mit bereits

bewerteten Texten verglichen werden. Ein kleiner Winkel zwischen den Vektoren deutet auf ähnlichen Inhalt hin. Fehlende Inhalte in einem neuen Text im Vergleich zu sehr gut bewerteten Texten können als Feedback gegeben werden (Foltz et al., 1999, S. 2).

Deutliche Fortschritte durch die Nutzung modernen Natural Language Processings (NLP) wurden durch die Systeme e-rater (Burstein et al., 1998) und IntelliMetric (Elliot, 2003) erzielt. Statt sich nur auf oberflächliche Merkmale wie Textlänge oder Rechtschreibfehler zu fokussieren, wurde bei den neueren Systemen auf tiefere linguistische Merkmale geachtet, wie unter anderem syntaktische Variation durch die Identifikation von Satzstrukturen und deren Komplexität. E-rater enthält zusätzlich Regeln „zur Erkennung rhetorischer Beziehungen basierend auf der syntaktischen und absatzbasierten Verteilung von Schlüsselwörtern, Phrasen und Strukturen“ (Burstein et al., 1998, S. 4). Texte werden nach Argumenten statt Absätzen angeordnet und deren Beziehungen gelabelt. Somit können zwei Maße, ausgerechnet werden, um Ähnlichkeiten im Text festzustellen. Ein Maß basierend auf dem Wortschatz des gesamten Textes und ein weiteres Maß für das spezifische Vokabular verwendet in den zuvor angeordneten Argumentabschnitten. Ähnlich wie bei IEA werden durch Vektoren und deren Winkel, Vergleiche zwischen Texten ermöglicht. Die gefundenen Vektoren werden mit den Vektoren von sechs Bewertungsstufen verglichen und der nächste Wert wird als Ergebnis festgehalten.

IntelliMetric gilt als eines der ersten in der Lehre verfügbaren AWE-Tools, das künstliche Intelligenz verwendet hat (Elliot, 2003, S. 2). Jedem Satz eines Textes werden Eigenschaften zu verwendeten Wortarten, Wortschatz, Satzstruktur und Ausdruck von Konzepten zugewiesen (Rudner et al., 2006, S. 6). Darauffolgend werden Technologien wie morphologische Analysen und Rechtschreibprüfungen angewandt. Insgesamt werden mehr als 500 linguistische und grammatische Merkmale identifiziert. Die gewonnenen Daten werden mit unterschiedlichen mathematischen Modellen verarbeitet und mit Trainingsdaten verglichen. Die Ergebnisse der Modelle werden durch eine geschützte Optimierungstechnik zu einer Gesamtbewertung zusammengetragen. Die Verwendung unterschiedlicher mathematischer Modelle soll analog zur Überprüfung durch unterschiedliche menschliche Prüfer sein (Rudner et al., 2006, S. 6).

Ein weiterer Ansatz wurde durch das Bayesian Essay Test Scoring sYstem (BETSY) verfolgt (Rudner & Liang, 2002). BETSY unterscheidet sich von den anderen Tools durch den Einsatz des Satzes von Bayes. Es werden drei bedingte Wahrscheinlichkeiten für Merkmale im Text bestimmt: Die Wahrscheinlichkeit, dass ein bestimmtes Merkmal im

Text auftritt, gegeben es wurde eine angemessene, teilweise angemessene oder nicht angemessene Leistung erbracht. Die Kategorisierung kann ausgeweitet werden. Die bedingten Wahrscheinlichkeiten werden aus einem Datensatz abgeleitet, der von Experten bewertet wurde. Dabei kann mit zwei Modellen klassifiziert werden, dem multivariaten Bernoulli-Modell und dem multinomialen Modell. Das multivariate Bernoulli Modell betrachtet Merkmale binär, dabei wird geprüft, ob Merkmale vorhanden oder nicht vorhanden sind, ohne die Häufigkeit zu beachten. Die bedingte Wahrscheinlichkeit für das Auftreten eines Merkmals wird geschätzt, indem geprüft wird, wie oft es in den Trainingstexten jeder Bewertungskategorie vorkommt. Folglich kann durch das Vorhandensein oder Fehlen eines Merkmals eine Klassifizierung erfolgen. Das multinomiale Modell hingegen prüft die Häufigkeit eines Merkmals im Text. Hier wird die Wahrscheinlichkeit für einen Text, gegeben der Kategorie, als Produkt aller Wahrscheinlichkeiten der im Text enthaltenen Merkmale, unter Berücksichtigung deren jeweiliger Häufigkeit, berechnet (Rudner & Liang, 2002, S. 4-7).

AES System	Developer	Technique	Main Focus	Instructional Application	Number of Essays Required for Training
PEG™	Page (1966)[8]	*Statistical*	Style	N/A	100–400
IEA™	Landauer, Foltz, & Laham (1997)[9]	*LSA*	Content	N/A	100–300
E-rater®	ETS development team (Burstein, et al.,1998)[10]	*NLP*	Style and content	Criterion℠	465
IntelliMetric™	Vantage Learning (Elliot, et al., 1998)[11]	*NLP*	Style and content	MY Access!®	300
BETSY™	Rudner[12] (2002)	*Bayesian text classification*	Style and content	N/A	1000

Abbildung 1 – Überblick erster automatischer Prüftools (Dikli, 2006, S. 23)

Abbildung 1 zeigt einen Überblick der Anfänge automatisierter Prüftools. Von Analysen mit Merkmalen wie Länge eines Textes oder Anzahl verschiedener Wortarten entwickelten sich AWE-Tools und konnten auch komplexere Sachverhalte wie Beziehungen zwischen Wörtern und Argumenten überprüfen. Besonders der Wandel von reinen Bewertungstools zu Bewertungstools, die Feedback geben konnten galt als großer Fortschritt.

3.2 Moderne Prüftools

3.2.1 Vorstellung gesichteter Literatur

Die zuvor beschriebenen Prüftools legten den Grundstein für die heutige vielfältige Landschaft der automatisierten Tools zur Qualitätssicherung wissenschaftlicher Arbeiten und Texten. Durch rasante Fortschritte in künstlicher Intelligenz, maschinellem Lernen und der allgemeinen Rechenleistung haben sich Prüftools in den letzten Jahren enorm weiterentwickelt. Während einige Aspekte weiterhin durch regelbasierte Abläufe geprüft werden, nutzen moderne Systeme neuronale Netze, um Texte zu analysieren und zu bewerten. Es wird sich von statischen Ansätzen entfernt, um stattdessen mit datengesteuerten, lernenden Systemen ein tieferes Verständnis der Texte zu erlangen. Modelle sind in der Lage komplexere Muster in Texten zu erkennen und Kontext über mehrere Textpassagen hinweg zu berücksichtigen. Dabei beruhen Modelle nicht nur auf vordefinierten Merkmalen, sondern lernen relevante Merkmale selbstständig aus Datensätzen. Moderne Systeme sind in der Lage detaillierteres und spezifischeres Feedback zu generieren. Sie werden nicht nur zur reinen Bewertung verwendet, sondern können als Begleiter beim Verfassen einer wissenschaftlichen Arbeit eine unterstützende Rolle einnehmen. Deshalb prägen neue Begriffe wie Intelligent Tutoring System oder Automated Corrective Feedback Tools die Landschaft der modernen automatisierten Prüfung von Texten. Diese Erkenntnisse aus der Literatur zu modernen Prüftools und Schreibassistenten werden im Folgenden näher beleuchtet.

Strobl et al. (2019) erstellten eine umfassende Übersicht zahlreicher Prüftools, die Schülern und Studierenden bei Schreiben von Texten unterstützen sollen. Dabei wurden 44 Tools anhand 26 qualitativer und quantitativer Merkmale untersucht. Diese Merkmale umfassen den Schreibprozess, pädagogische Ansätze, Feedback und technologische Spezifikationen. Die Studie von Strobl et al. soll feststellen, welche Unterstützungstools auf dem Markt existieren und welche Prüfbereiche abgedeckt sind oder Lücken aufweisen. Die Ergebnisse zeigen, dass ein Ungleichgewicht der verfügbaren Tools hinsichtlich unterstützter Sprachen, Genres und pädagogischer Schwerpunkte existiert. Die meisten Tools unterstützen argumentatives Aufsatzschreiben in Englisch, während z. B. akademische Arbeiten und andere Sprachen unterrepräsentiert sind. Der pädagogische Fokus wird in die zwei Ebenen Mikro- und Makroebene unterteilt. Die Prüfung der Mikroebene beinhaltet Fakten wie Rechtschreibung, Grammatik und Wortfrequenzen und ist sehr gut vertreten. Weniger gefördert wird die Makroebene, die

sich übergestellten Themen wie Kohärenz oder der Argumentationsstruktur eines Textes beschäftigt. Die neuen Technologien unterstützen nicht nur bei der Bewertung, sondern verändern den gesamten Prozess des Schreibens. Während neun Tools angegeben haben NLP zu verwenden, hat die Mehrheit nicht bekannt gemacht, welche Technologien genau verwendet wurden. Lediglich drei der untersuchten Tools haben sich mit Zitationsregeln beschäftigt. Zur Prüfung der Form einer Arbeit wurden keine Befunde erwähnt. Auffällig ist außerdem, dass 39 der 44 Tools eine Webanwendung bereitstellen und es wenige eigenständige Programme auf dem Markt gibt.

Imran und Almusharraf (2023) untersuchten mit 30 relevanten Artikeln die Rolle ChatGPTs als Schreibassistent in der Lehre. ChatGPT wird als komplettes Paket bezeichnet, von der Ideenfindung, der Erstellung und Bearbeitung bis zum finalen Korrekturlesen des zu erstellenden Textes. Zu den positiven Aspekten beim Nutzen der generativen künstlichen Intelligenz (GenAI) nennen die Autoren eine erhöhte Effizienz, da kohärente und gut strukturierte Texte schneller erstellt werden können und der Fokus auf andere Anforderungen der wissenschaftlichen Arbeit gelegt werden kann. Auch sollen präzise und konsistente Inhalte mit höherer Wahrscheinlichkeit auftreten und Fehler vermieden werden. Neben diesen Chancen stehen Herausforderungen bei der Verwendung ChatGPTs. Zwar können Texte in einigen Anwendungsfällen präziser und leichter zu verstehen werden, jedoch kann es gleichermaßen zu Ungenauigkeiten oder Generalisierungen kommen. Das Fehlen von Originalität ist ein weiteres potenzielles Problem beim Arbeiten mit GenAI. Besonders im Hinblick auf ethische Aspekte ist laut Imar und Almusharraf noch viel Forschungsbedarf notwendig. ChatGPT wird hauptsächlich als Schreibassistent gesehen und soll als kollaboratives Schreibwerkzeug für Bereiche wie Feedback oder Brainstorming dienen. Dabei kann ChatGPT nicht als menschlicher Autor gesehen werden und das Wissen und die Fähigkeiten menschlicher Autoren nicht ersetzen. Menschliche Kontrolle und Überprüfung ist weiterhin notwendig. Dennoch kann die wichtige Rolle GenAIs beim Schreibprozess nicht angefochten werden und muss sowohl von Lernenden als auch von Lehrenden verstanden werden, um die Integrität wissenschaftlicher Arbeiten zu bewahren.

Als Kontrast dazu haben Mizumoto und Eguchi (2023) die Rolle GenAIs nicht als Schreibassistent, sondern als reines Bewertungstool von Texten untersucht. Es wurden 12,100 Texte aus einem Korpus schriftlicher Englischtexte von Nichtmuttersprachlern mit dem Modell GPT-3-text-davinci-003 untersucht und mit Ergebnissen anderer Tools verglichen. Die Studie zeigt durch geringe Abweichung bei der Bewertung verschiedener

Kriterien, dass GenAI präzise und zuverlässige Bewertungen abgeben und den Prozess der Bewertung beschleunigen kann. Es wurde festgestellt, dass das Hinzufügen linguistischer Merkmale die Vorhersage signifikant verbessert. Dennoch waren die Ergebnisse nicht vollkommen identisch mit einer menschlichen Bewertung, wodurch aufgewiesen wurde, dass menschliche Bewertungen nicht ersetzt werden können, sondern eine Kombination aus einer Bewertung von Mensch und Maschine sinnvoll ist. Besonders wichtig bei der Bewertung mit GenAI-Tools ist die Formulierung passender Prompts, da diese die Genauigkeit stark beeinflussen können.

Während sich die vorherigen Tools hauptsächlich mit sprachlichen und inhaltlichen Anforderungen beschäftigt haben, wird im Folgenden näher auf die automatisierte Qualitätssicherung Layoutbezogener Anforderungen eingegangen. Xia und Zhang (2024) haben ein Tool mit der auf Java basierenden Sprache MVEL entwickelt, um das Layout von Word-Dokumenten zu prüfen. Das System besteht aus den drei Hauptmodulen Regelkonfiguration, Regelausführung und dem Erkennungsbericht. Wobei die Konfiguration der Regeln sich hauptsächlich auf die Struktur bzw. den Aufbau der wissenschaftlichen Arbeiten bezieht. Es wurden mehr als zehn Abschlussarbeiten zum Testen des regelbasierten Tools ausgewählt. Während einige formale Fehler wie Schriftgröße des Titels gefunden wurden, können einige Aspekte wie Tabellen noch nicht geprüft werden und benötigen manuelle Unterstützung. Die Formate der einzelnen Teilbereiche der Abschlussarbeiten konnten jedoch zuverlässig analysiert werden. Für einige Aspekte wie Format des Abstracts oder der Kapitelüberschriften können außerdem automatische Korrekturen vorgenommen werden.

Cioffi und Peroni (2022) bewerten sieben Tools zur Extraktion und Segmentierung von Literaturverzeichnissen. Dabei werden 2,538 Literaturverweise aus 27 thematischen Bereichen evaluiert. Es wurden drei Metriken untersucht: korrekt identifizierte Referenzen, korrekt identifizierte Themenbereiche pro Referenz und korrekt identifizierte Inhalte pro Referenz. Während das Tool Anystyle insgesamt die beste Bewertung vorwies, konnten unterschiedliche Tools bei verschiedenen Metriken und Referenzen abweichende Ergebnisse erzielen. Bis auf einen Ausreißer weisen die Tools zuverlässige Ergebnisse bei der Erkennung der Literaturverweise auf. Von den Autoren wird jedoch auf den kleinen Datensatz aufmerksam gemacht. An dieser Stelle ist zusätzlich anzumerken, dass es sich nicht um Studierendenabgaben mit potenziellen Fehlern gehandelt hat, die ein Erkennen erschweren können.

Foltýnek et al. (2020) untersuchen und vergleichen die Effektivität von 15 Plagiatserkennungstools, indem sie Testdokumente in acht verschiedenen Sprachen evaluieren. Bei den untersuchten Tools handelt es sich um sogenannte Text-matching systems, die eine Datenbank potenzieller Quellen aufweisen, verschiedene Erkennungsmethoden anwenden und eine Schnittstelle für Nutzer bereitstellen. Die Ergebnisse zeigen, dass einige Systeme zwar durchaus beim Erkennen von Plagiaten helfen können, jedoch konnte keine Software alle Plagiate finden und es kann zu falsch-positiven und falsch-negativen befunden kommen. Die Effektivität der Tools variierte stark in Abhängigkeit von Dokumenttyp, Sprache sowie der Unterscheidung zwischen Plagiaten aus einer einzelnen Quelle und aus mehreren Quellen. Besonders auffällig war, dass bei übersetzten Plagiaten die Erkennungsrate sehr gering ausgefallen ist. Schließlich konnte die Effektivität keiner der 15 Plagiatserkennungstools als nützlich bewertet werden. Fünf Tools wurden als teilweise nützlich bewertet. Die Autoren weisen darauf hin, dass stets eine menschliche Prüfung notwendig ist, und warnen außerdem vor ethischen und datenschutzrechtlichen Risiken beim Nutzen der Prüftools.

3.2.2 Vorstellung ausgewählter Prüftools

Für Studierende steht eine umfangreiche Auswahl an Tools zur Prüfung ihrer Arbeiten im Internet zur Verfügung. Dabei kann es sich um frei zugängliche oder kostenpflichtige Tools handeln. Viele dieser Tools überschneiden sich in ihren Funktionalitäten, weshalb in diesem Abschnitt einige Tools vorgestellt werden, die verschiedene Anforderungen an wissenschaftlichen Arbeiten abdecken.

Eines der bekanntesten modernen Tools zu Überprüfung von Texten und zur Schreibassistenz ist Grammarly (2025). Im Laufe der Jahre hat Grammarly sich zu einem KI-gestützten, kontextsensitiven Schreibassistenten entwickelt. Derzeit sind die Features von Grammarly jedoch nur in englischen Texten einsetzbar. Grammarly bietet die Möglichkeit Texte in einem Webbasierten Editor zu prüfen. Texte können direkt im Editor verfasst oder in den Editor kopiert werden. Nutzer haben zusätzlich die Möglichkeit eigene Dateien hochzuladen, deren Text danach im Editor von Grammarly bearbeitet und geprüft werden kann. Darüber hinaus können eine Desktop-App oder Browsererweiterungen installiert werden, die zahlreiche Anwendungen wie Microsoft Word oder Google Docs unterstützen, um Grammarly in der jeweiligen Umgebung zu nutzen. Zu prüfende sprachliche Aspekte umfassen Rechtschreibung, Grammatik und

Stil, jedoch sind einige Prüfungen und Features nur für zahlende Nutzer verfügbar. Je nach gewähltem Zeitplan liegen die Kosten derzeit zwischen 12 € und 30 € pro Monat.

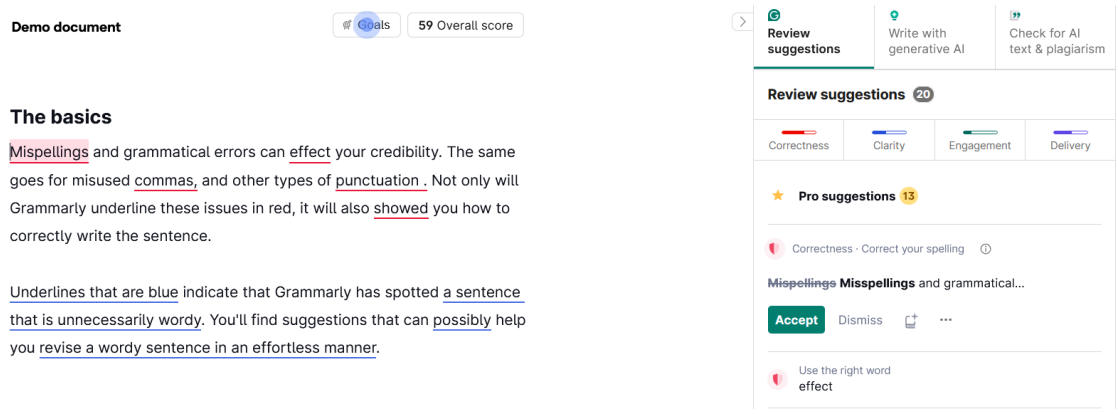

Abbildung 2 – Oberfläche des Webeditors von Grammarly

Feedback wird direkt im jeweiligen Editor angezeigt und Fehler werden Text farblich markiert. Am rechten Bildschirmrand werden Verbesserungsvorschläge und zusätzliche Informationen angezeigt. Gegebenes Feedback kann individualisiert werden, indem Kontext für den Zweck und die Zielgruppe des zu verfassenden Textes gegeben wird. Jederzeit kann ein Überblick über die Gesamtbewertung des Textes eingesehen werden.

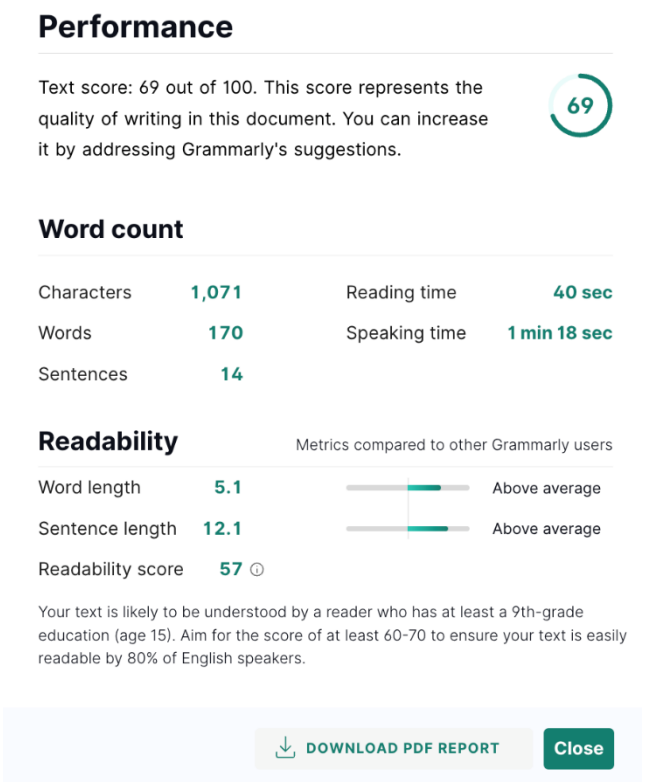

Abbildung 3 – Bewertungsbildschirm Grammarly

Grammarly gibt Nutzern eine Gesamtbewertung in der sie eine Übersicht über den allgemeinen Bewertungsscore, Details zur Textlänge und zur Lesbarkeit einsehen können. Zusätzlich kann eine detaillierte Übersicht als PDF heruntergeladen werden, die weitere wertvolle Einblicke bietet. Darin enthalten ist ein Prozentsatz seltener Wörter im Vergleich zu einem Korpus aus den 5.000 gängigsten englischen Wörtern und eine Prozentzahl, die den Anteil einzigartiger Wörter im Text aufzeigt. Auch werden die Kategorien der noch vorhandenen Fehler bereitgestellt. Am Ende des Dokuments ist der gesamte Text zu finden, in dem Fehler farblich und nummerisch hervorgehoben sind, gefolgt von einer Tabelle, die alle Fehler näher erläutert. Um den Überprüfungsprozess an verschiedene Anforderungen von Texten anzupassen, kann aus Bereichen wie z. B. wissenschaftlich, kreativ oder E-Mail gewählt werden. Mittels GenAI können Nutzer Prompts eingeben, um Texte umschreiben oder neu generieren zu lassen. Darüber hinaus bietet Grammarly eine Plagiatsprüfung, die geschriebene Texte mit Quellen aus dem Internet vergleicht und Hilfe bei der Formatierung von Zitationen im Text in den Zitierweisen APA, MLA und Chicago. Als eines der größten Schreibassistenz-Tools liefert Grammarly einen umfangreichen Katalog an Features.

LanguageTool (2025) ist ein weiterer Schreibassistent mit Fokus auf sprachlichen Überprüfungen, der eine kostenlose Open Source Basisversion und eine kostenpflichtige Premiumversion zwischen 4,99 € pro Monat pro Jahr und 19,90 € pro Monat anbietet. Neben Anbietern wie Scribbr, Quillbot, oder CliffsNotes ist LangaugeTool Teil der Unternehmensgruppe Lerneo. LanguageTools Alleinstellungsmerkmal ist, dass es über 30 Sprachen unterstützt und nicht nur in einer offiziellen Weboberfläche oder als Erweiterung für andere Texteditoren verwendet werden kann, sondern es kann ein eigener LanguageTool-Server eingerichtet werden. Bekannte Prüftools wie rechtschreibpruefung24 basieren auf der Textprüfung von LanguageTool.

Deutlich schwerer für Studierende ist es ein Prüftool für layoutbezogene Anforderungen an wissenschaftlichen Arbeiten zu finden. SciFlow (2025) bietet automatische Formatierungen und Layouts mit Hilfe vordefinierter Einreichungsvorlagen. Es kann aus verschiedenen Aufgabentypen, Sprachen und Zitationsstilen gewählt werden. Danach können Nutzer in SciFlows Editor weiterarbeiten oder die Arbeit als Tex-, docx- oder PDF-Datei herunterladen. Wobei die Formatierung zwar bei der PDF-Datei vollständig und korrekt übernommen wird, aber bei der Word-Datei einige Elemente wie Überschriften nicht korrekt dargestellt werden. Auch ist anzumerken, dass die Formatvorlage nicht näher angepasst werden kann, sodass keine Spezifikationen für z. B.

Schriftart, Schriftgröße oder Zeilenabstand getroffen werden können. Darüber hinaus ist der Fließtext standardgemäß als linksbündig und nicht im Blocksatz formatiert. Da keine weiteren Überprüfungen zum Layout vorgenommen werden, müssen Studierende diese Anforderungen wie im Word-Editor selbstständig erfüllen.

Die Softwarelösung DocShifter stellt das Modul DocValidator (DocShifter, 2025) als spezialisiertes Modul zur automatisierten Prüfung von Word-Dateien zur Verfügung. Zwar liegt der Fokus von DocShifter und DocValidator nicht auf Studierendenabgaben, aber es werden dennoch relevante formale und stilistische Kriterien behandelt. DocValidator hilft dabei, Inkonsistenzen bei der Verwendung von Formatvorlagen oder schwer lesbare Abbildungen zu erkennen. Mit DocValidator können einige der gefundenen Formatierungsfehler automatisch korrigiert werden. Zwar bietet DocValidator interessante Funktionalitäten, kann jedoch von Studierenden nicht unmittelbar genutzt werden, da eine kostenpflichtige Lizenz von DocShifter erworben werden muss. Eine einfache Online-Bezahlung für Privatpersonen ist nicht vorgesehen.

Die Plattform Scribbr (2025) liefert eine breite Spanne aus Features für Studierende und Lehrende an. Neben Korrektur, Plagiatsprüfung und Unterstützung beim Schreiben durch künstliche Intelligenz sind bei Scribbr über 700 internationale Prüfer eingestellt, die neben der automatischen Prüfung ein manuelles Korrekturlesen wissenschaftlicher Arbeiten anbieten. Die Preismodelle hängen dabei von den Prüfkriterien ab. Beispielsweise können Zitationen durch Experten für 2,20 € pro Quelle geprüft werden. Außerdem wird eine umfassende KI-Zitationsprüfung für den Zitierstil APA 6. und 7. Edition für 9,95 € angeboten. Bei der KI-Prüfung werden Details wie die korrekte Verwendung von Trennzeichen innerhalb einzelner oder mehrerer Quellenangaben oder das korrekte Verwenden von et al. abhängig von der Autorenanzahl geprüft. Neben der Zitationsprüfung wird ein manueller Layout-Check für 1,95 € pro korrigierte Seite oder in Kombination mit anderen Diensten für 0,014 € pro Wort angeboten. Es werden Formatierungen der Absätze, Seitenzahlen, die Nummerierungen der Überschriften, Verzeichnisse und Beschriftungen geprüft. Spezifische Anforderungen können von Nutzern in der Bestellung definiert werden. Neben den verschiedenen Überprüfungen sind auf der Webseite von Scribbr Informationen und Hinweise zum Verfassen wissenschaftlicher Arbeiten kostenlos zugänglich. Dabei werden die Kategorien Schreiben, Forschen und Zitieren detailliert ausgeführt.

Zwar enthalten einige der bereits vorgestellten Tools eine Plagiatserkennung, jedoch gibt es spezialisierte Tools wie PlagAware (2025), die bessere Ergebnisse erzielen sollen. PlagAware wird von Studierenden, Hochschulen und weiteren Institutionen verwendet, um eine ausführliche Plagiatsprüfung durchzuführen. Es werden mehrere Zahlungspakete angeboten, wobei die Preise für Privatpersonen und kleine Organisationen zwischen 12,90 € bei bis zu 100 Seiten und 89,90 € bei bis zu 1500 Seiten liegen. Zusätzlich wird eine Lehr-Lizenz für 189,00 € pro Jahr angeboten. Dafür erkennt PlagAware verschiedene Arten von Plagiaten wie direkte Textübernahmen oder paraphrasierte Inhalte ohne Zitation. Vor der Prüfung werden Texte normalisiert, übersetzt und es werden Textvarianten mit Synonymen erstellt. Danach werden einzelne Satzfragmente entnommen und die Relevanz dieser geprüft. Bei jeder Prüfung wird der Text mit mehreren tausenden passenden Quellen und Dokumenten verglichen. Gefundene Übereinstimmungen werden bewertet und schließlich in einem Ergebnisbericht dargestellt.

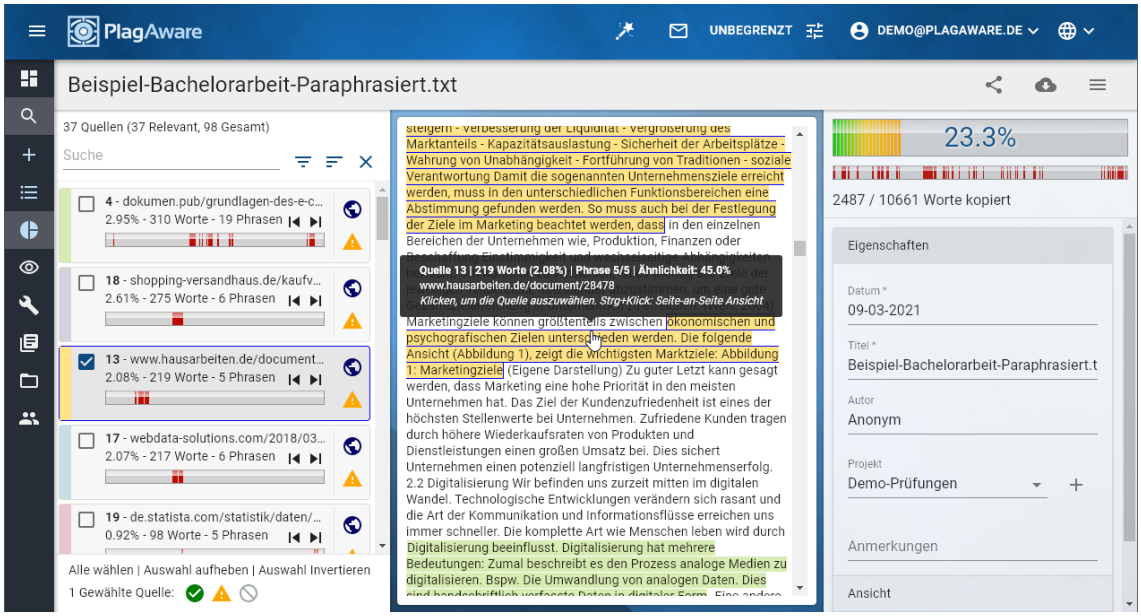

Abbildung 4 – Interaktiver Prüfbericht von PlagAware

In dem interaktiven Bericht werden potenzielle Plagiate farblich markiert, entsprechende Quellen aufgelistet und eine Gesamtbewertung, abhängig von der Wortanzahl und der Anzahl plagiatsverdächtiger Wörter, gegeben. Darüber hinaus ist eine direkte Gegenüberstellung einer Quelle mit dem zu prüfenden Text möglich und Nutzer können die Bewertung einzelner Quellen anpassen. Neben den inhaltlichen Prüfungen untersucht PlagAware Arbeiten mit einem KI-Detektor, um potenziell KI-generierte Texte zu finden. Gleichzeitig wird eine Stilanalyse durchgeführt, damit Stilbrüche im Text erkannt

werden. PlagAware wirbt außerdem mit Datenschutzkonformität, sicherer Verschlüsselung und dem Serverstandort Deutschland.

4 Technische Umsetzung

4.1 Vorverarbeitung des Dokuments

Mit der Vorverarbeitung des Word-Dokuments wird die Grundlage für die darauffolgenden Analyse- und Kommentierungsfunktionen gelegt. Da wissenschaftliche Arbeiten unterschiedlich gestaltet und formatiert werden können, ist es wichtig eine große Bandbreite an Möglichkeiten abzudecken, wie ein Dokument zu verarbeiten ist. Dokumente sind so vorzubereiten, dass spätere Analysen auf relevante Inhalte durchgeführt werden können. Zusätzlich muss die Rückverfolgbarkeit eines Abschnitts zum Originaldokument gewährleistet werden. Dies wird durch die Nachverfolgbarkeit der einzelnen Run-Elemente im Word-Dokument erzielt.

Die von einem Nutzer hochgeladene Word-Datei wird an das Backend gesendet. Im Python-Backend SciChecks wird die Datei zuerst in ihre einzelnen Bestandteile zerlegt. Da .docx-Dateien ZIP-Archive sind, die aus mehreren Extensible Markup Language-Dateien (XML) bestehen, werden die relevanten Wurzelelemente in der Liste *roots* hinterlegt. Die wichtigste Ressource, in der die tatsächlichen Inhalte eines Word-Dokuments abgebildet sind, ist die Datei *document.xml*. In dieser befinden sich unter anderem Absätze (*w:p*) und Läufe bzw. Runs (*w:r*). Ein Dokument besteht aus mehreren Absätzen, die wiederum aus mehreren Runs bestehen können. Runs entstehen, wenn innerhalb eines Absatzes Formatveränderungen auftauchen (Abbildung 5). Dazu gehört z. B. auch die farbliche Markierung durch die automatische Rechtschreibprüfung von Word. Runs spielen eine maßgebliche Rolle bei der Vorverarbeitung des Dokuments, da sich durch das Kürzen oder Aufteilen des Dokuments die Anzahl der Runs und der Startpunkt verändern.

Abbildung 5 –Darstellung in Word mit Beschreibungen aus document.xml *(Krieger & Labra, 2023)*

4.1.1 Entfernen verschachtelter Elemente

Die Funktion *preprocess_document()* beinhaltet alle Funktionen, die zur Vorverarbeitung des Wurzelelements von *document.xml* notwendig sind. Als Erstes wird die XML-Struktur der sogenannten AlternateContent-Elemente bereinigt. Da in Word-Dokumenten Inhalte nicht nur als einfacher Text, sondern auch in alternativen Darstellungen wie Textfeldern angezeigt werden können, werden AlternateContent-Elemente genutzt. Zwar sind diese Elemente für eine flexible Anzeige im Word-Dokument sinnvoll, jedoch erschweren sie aufgrund ihrer verschachtelten Struktur den Analyseprozess. Bei der Analyse wird durch alle Absätze im Dokument iteriert. AlternateContent-Elemente enthalten mehrere verschachtelte Absatz-Elemente, die dadurch doppelt durchlaufen werden. Damit beim Zählen der einzelnen Run-Indizes keine Fehler auftreten, werden AlternateContent-Elemente entfernt.

```
def remove_alternate_content_elements(root):
    # Define XML namespaces
    namespaces = {
        'w': 'http://schemas.openxmlformats.org/wordprocessingml/2006/main',
        'mc': 'http://schemas.openxmlformats.org/markup-compatibility/2006'}
    # Find all alternate content elements
    alternate_content_elements = root.xpath('//mc:AlternateContent',
                                 namespaces=namespaces)
    for alt_content in alternate_content_elements:
        parent = alt_content.getparent()
        # Remove alternate content elements
        if parent is not None:
            empty_run = etree.Element(
           '{http://schemas.openxmlformats.org/wordprocessingml/2006/main}r')
            index = parent.index(alt_content)
            parent.insert(index, empty_run)
            parent.remove(alt_content)
```

Listing 1 – Funktion zum Entfernen der AlternateContent-Elemente

Die Bereinigung wird in der Funktion *remove_alternate_content_elements()* durchgeführt. Vor dem Entfernen müssen zunächst die XML-Namensräume definiert werden. Neben dem typischen *w* für Word-Inhalte muss der Namensraum *mc* für Markupkompatibilität definiert werden. Danach kann die gesamte XML-Struktur durchsucht werden, um alle relevanten Elemente zu finden und in einer Liste zu hinterlegen und anschließend zu entfernen. Hierbei werden die Runs nicht mitgezählt, da spätere Operationen gleichermaßen ohne AlternateContent-Elemente durchgeführt werden.

4.1.2 Extraktion des Textteils

Ein wesentlicher Aspekt der Vorverarbeitung wissenschaftlicher Arbeiten besteht darin, den eigentlichen Textteil der Arbeit zu isolieren. Das heißt der Textteil muss von Abschnitten wie Deckblatt, Abstract, Inhaltsverzeichnis, anderen Verzeichnissen oder sonstigen Inhalten getrennt werden. Denn besonders Analysen der Kategorie Formalia sollen auf diese Bereiche nicht angewandt werden, da sonst falsch-positive Ergebnisse entstehen können.

Umgesetzt wird dies in der Funktion *trim_root()*, indem mittels heuristischer Regeln der Beginn des Textteils erkannt und der XML-Baum ab diesem Punkt extrahiert wird. Dafür werden sowohl formatbedingte Merkmale als auch textuelle Hinweise genutzt, um den Startpunkt des Textteils zu finden. Zuerst wird eine Objektkopie des XML-Baums erstellt, damit das Original unverändert für andere Verarbeitungsschritte genutzt werden kann. Durch *root.findall('.//*', namespaces=namespaces)* wird jedes Element des XML-Baums erfasst und in einer Liste gesammelt.

Beim Iterieren durch diese Liste wird zuerst geprüft, ob ein Element ein Run-Element ist, da die Anzahl der durchlaufenen Run-Elemente gezählt werden muss. Auch wird nach Absatz-, Stil- und Text-Elementen gesucht. Der Wert der Stil- und Text-Elemente wird gespeichert und da sich diese innerhalb von Absätzen befinden, werden die Werte zurückgesetzt, sobald ein Absatz-Element gefunden wird. Die Text-Elemente werden akkumuliert gespeichert, da sogar einzelne Wörter, die für das Finden des Startpunkts relevant sind, in mehreren Text-Elementen abgebildet werden können.

```
# Check for specific headers
if pStyle_val is not None and ("Einleitung" in accumulated_text  or
    accumulated_text.startswith("1")) and (pStyle_val.startswith("berschrift")
    or pStyle_val.startswith("Heading")):
     if not pStyle_val.startswith("TOC"):
         # Record index of the header element
         header_index = idx
         found_section_start = True
         run_counts.append(run_count)
         run_count = 0
         # Get parent paragraph of the header and add to list
         header_paragraph = elem.getparent().getparent()
         elements_to_keep.append(header_paragraph)
         continue
```

Listing 2 – Startpunkterkennung des Textteils

Wenn der Stil eines Elements durch *berschrift* (Umlaute werden nicht angezeigt) oder *Heading* auf eine Überschrift deutet, kann zusätzlich der akkumulierte Text untersucht werden. Die Kombination aus einer Überschrift und einem Text, der mit einem Schlüsselwort wie *Einleitung* oder lediglich mit der Ziffer *1* beginnt, wird als Beginn des Textteils interpretiert. Die Zählung der Run-Elemente wird zurückgesetzt und die Absätze des Textteils werden in einer Liste gespeichert. Da jedes einzelne Element durchlaufen wird und der Startpunkt stets in einem Text-Element gefunden wird, müssen auch die vorherigen Elemente dieses Text-Elements gespeichert werden. Text-Elemente sind Kinder von Run-Elementen, welche Kinder von Absatz-Elementen sind. Demnach wird der Startabsatz gefunden, indem nach demjenigen Element gesucht wird, dass sich zwei Hierarchieebenen höher befindet.

Falls die Suche nach dem Start des Textteils nicht erfolgreich war, wird der originale XML-Baum zurückgegeben. Wurde der Start jedoch gefunden, wird ein neuer XML-Baum erstellt. Auch am Ende des Dokuments wird eine weitere Abgrenzung getroffen. Die eidesstattliche Versicherung kann sich in einer wissenschaftlichen Arbeit am Anfang oder am Ende befinden und muss für die Analyse entfernt und somit nicht in den neuen XML-Baum aufgenommen werden. Umgesetzt wird dies, indem nach Schlüsselwörtern für *Literaturverzeichnis* oder *Eidesstattliche Versicherung* gesucht wird. Es ist darauf zu achten, dass diese Schlüsselwörter allein in einem Absatz stehen sollen, um falsche-positive Befunde zu vermeiden. Die eidesstattliche Versicherung befindet sich üblicherweise hinter dem Literaturverzeichnis und dazu passende Absätze werden nicht in den neuen XML-Baum übernommen. Neben dem originalen und neuen XML-Bäumen wird auch die Anzahl der Run-Elemente bis zum neuen Startpunkt zurückgegeben, um später präzise Zuordnungen treffen zu können.

4.1.3 Extraktion des Literaturverzeichnisses

Neben dem Kürzen der wissenschaftlichen Arbeit ist die Extraktion des Literaturverzeichnisses ein zentraler Bestandteil der Vorverarbeitung. Beim Überprüfen der Zitationsregeln und Quellengaben ist es essenziell das Literaturverzeichnis isoliert zu betrachten. In der Funktion *extract_bibliographie()* wird das Literaturverzeichnis als eigener XML-Baum gespeichert und in einem weiteren XML-Baum für Analysen, die das Literaturverzeichnis nicht prüfen sollen, entfernt.

```
def is_bibliography_paragraph(p):
    …
    # Return False if paragraph is empty
    paragraph_text = ''.join(t.text for t in texts if
t.text).strip().lower()
    if not paragraph_text:
        return False
    # Check if paragraph contains any bibliography keywords
    if any(bib in paragraph_text for bib in bib_keywords):
        if pStyle_val and any(pStyle_val.lower().startswith(
        keyword) for keyword in heading_keywords):
            # Ensure paragraph is styled as heading and is short
            if len(paragraph_text.split()) <= 2
            and any(bib in paragraph_text for bib in bib_keywords):
                return True
    return False
```

Listing 3 – Erkennung der Startposition des Literaturverzeichnisses

Der Prozess verläuft ähnlich wie bei der Suche nach dem Start des Textteils der wissenschaftlichen Arbeit. Erneut wird mittels eines Stils, der auf eine Überschrift deutet und bestimmten Schlüsselwörtern nach dem Literaturverzeichnis gesucht. Dabei wird darauf geachtet, dass der Absatz nur zwei oder weniger Wörter enthält. Zusätzlich wird mit einem Filter sichergestellt, dass der Befund sich nicht zu Beginn des Dokuments befindet, da ein Literaturverzeichnis am Ende einer Arbeit zu platzieren ist. Es wird ein XML-Baum erstellt, der auf dem bereits gekürzten Dokument basiert und alle Absätze bis zum Beginn des Literaturverzeichnisses enthält. Bei der Erstellung des XML-Baums, der nur das Literaturverzeichnis beinhalten soll, müssen die Inhalte geprüft werden, um nicht fälschlicherweise Inhalte aufzunehmen, die keine Quellenverweise sind. Dafür werden Absatz-Elemente und SDT-Content-Elemente (Strukturierte Dokumententags auf Blockebene) analysiert, wobei letztere vorkommen können, wenn Literaturverwaltungstools beim Anfertigen des Literaturverzeichnisses verwendet wurden. Mit einem regulären Ausdruck wird geprüft, ob es sich tatsächlich um einen Eintrag des Literaturverzeichnisses oder um einen anderen ungültigen Absatz handelt. Auch leere Absätze werden als ungültig definiert und nicht in den neuen XML-Baum übernommen. Treten drei aufeinanderfolgende ungültige Zeilen auf, wir die Extraktion beendet, da dies als Ende des Literaturverzeichnisses zu verstehen ist. Die Funktion *extract_bib()* gibt schließlich die beiden neuen XML-Bäume zurück, die später für eine präzise Analyse genutzt werden.

4.1.4 Indizierung der Run-Elemente

Das einfache Zählen der Run-Elemente ist nicht ausreichend, um später Kommentare an der korrekten Stelle in Word einzufügen. Deshalb müssen mit der Funktion *get_run_indices()* der genaue Startindex der gekürzten Runs im Originaldokument und eine Zuordnung aller Run-Elemente zu ihrem Index gefunden werden. So können die Runs später gezielt adressiert und Kommentare eingefügt werden, obwohl Analysen auf unterschiedlichen XML-Strukturen durchgeführt wurden. Dafür wird der gekürzte XML-Baum analysiert. Falls eine Kürzung des Dokuments stattgefunden hat, wird die Anzahl der anfangs entfernten Run-Elemente minus eins als Startindex gewählt. Wurde das Dokument nicht gekürzt, ist der Startindex null. Dieser Startindex wird bei der Analyse als Parameter übergeben, sodass die Indizierung der Run-Elemente mit Fehlern der Position im Originaldokument und nicht im gekürzten Dokument übereinstimmt.

```
def create_run_dict(root):
    runs_with_indices = {}
    # Iterate over all runs in document and save elements and texts in dict
    for idx, run in enumerate(root.iter(
    '{http://schemas.openxmlformats.org/wordprocessingml/2006/main}r')):
        runs_with_indices[idx] = run
        text_elem = run.find(
        '{http://schemas.openxmlformats.org/wordprocessingml/2006/main}t')
        run_text = text_elem.text if text_elem is not None else ''
        runs_with_indices[idx] = {
            'element': run,
            'text': run_text
        }
    return runs_with_indices
```

Listing 4 – Erstellung des Schlüssel-Werte-Paares für Run-Elemente

Die Funktion *create_run_dict()* erstellt ein Python-Dictionary, dass alle Run-Indizes als Schlüssel und Run-Elemente und deren Text als Werte speichert. So können später Kommentare eingefügt werden, die sich auf den Text im Run-Element beziehen. Das erstellte Python-Dictionary, die angepassten XML-Bäume und relevanten Indizes werden zusammengetragen und in einem Python-Dictionary zurückgegeben, damit bei der Weiterverarbeitung schnell auf die jeweiligen Daten zugegriffen werden kann.

4.2 Analyse Sprache

4.2.1 Extraktion und Strukturierung des Textes

Neben den bereits vorhandenen Kategorien Formalia, Struktur und Zitation wird im Folgenden beschrieben, wie SciCheck die Kriterien der neuen Kategorie Sprache

überprüft. Die einzelnen Funktionen und Schritte, die für die Analyse der Sprache notwendig sind, werden in der übergeordneten Datei *language_manager.py* in der Hauptfunktion *get_language_results()* ausgeführt.

Während im vorherigen Abschnitt die Manipulation der grundlegenden XML-Struktur beschrieben wurde, gilt es in diesem Schritt die einzelnen Text-Elemente in einen vollständigen analysierbaren Text zu transformieren. Dabei muss darauf geachtet werden, dass später Rückschlüsse auf die Position der gefundenen Fehler gezogen werden können. Die Funktion *get_text_and_runs()* sorgt dafür, dass ein zusammenhängender Text bereitgestellt wird, auf dem Analysen durchgeführt werden können. Als Grundlage dafür dient der bereits gekürzte XML-Baum, der den gesamten Textteil enthält.

```
def get_text_and_runs(root, run_start_idx, nlp):
    runs_data = extract_runs_with_metadata(root, run_start_idx)
    full_text, run_offsets = merge_runs_for_analysis(runs_data)
    sentences, words = tokenize_text(full_text, nlp)
    return runs_data, run_offsets, full_text, sentences, words
```

Listing 5 – Schritte zur Verarbeitung des Textes

Es werden drei Funktionen aufgerufen, um folgende Teilschritte durchzuführen:

1. Extrahieren der Run-Elemente inklusive Zusatzinformationen
2. Erstellung des Fließtextes und speichern der exakten Zeichenpositionen der Run-Elemente
3. Segmentierung des Textes in einzelne Sätze und Wörter

Zuerst werden in *extract_runs_with_metadata()* alle Absätze und deren Run-Elemente durchlaufen und der Wert der Text-Elemente gespeichert. Der Text wird außerdem von Zeichen wie *\xa0* gesäubert, das für geschützte Leerzeichen steht. Es wird nicht nur der Text und der Run-Index gespeichert, sondern, wenn vorhanden, auch die Kapitelnummer und der Kapitelname. Dafür wird für jeden Absatz das Stil-Element geprüft und im Falle einer Überschrift, wird sich das genaue Kapitel bis zum nächsten Kapitel gemerkt.

Nachdem die einzelnen Bestandteile mit Zusatzinformationen extrahiert wurden, müssen diese in *merge_runs_for_analysis()* zu einem Fließtext verbunden werden. Dafür wird durch alle Bestandteile iteriert, um die Textinhalte zu kombinieren. Auch die exakte Anzahl der Zeichen jedes einzelnen Textabschnitts wird gezählt und gespeichert, damit später bei der Verarbeitung des gesamten Textes durch die Position der Zeichen auf den oder die passenden Run-Elemente verwiesen werden kann. Beim Kombinieren der

Textabschnitte wird geprüft, wann ein neuer Absatz beginnt. Hier muss ein Leerzeichen eingefügt werden, da die Texte nach einem Absatzwechsel sonst ohne Trennung aneinanderhängen würden.

Anschließend kann in der Funktion *tokenize_text()* mit Hilfe der NLP-Bibliothek spaCy der Fließtext in Sätze und Wörter unterteilt werden. Abhängig von der konfigurierten Sprache wird für diesen Prozess ein passendes Modell gewählt. Die Segmente werden in Listen gespeichert. Die Listen der Sätze und Wörter, der Fließtext und alle Informationen zu den Run-Elementen werden zurückgegeben und dienen in den nächsten Schritten als Grundlage zur Analyse der sprachlichen Kriterien.

4.2.2 Rechtschreibprüfung mit LangaugeTool

Als erste Überprüfung wird eine umfassende Rechtschreibprüfung mit LanguageTool durchgeführt. Die Funktion *check_spelling()* fungiert als Einstiegspunkt für die Rechtschreibprüfung. Die Funktion bekommt den Fließtext, die Zeichenpositionen und die gewählte Sprache als Parameter übergeben. Die Rechtschreibprüfung läuft in einer eigenständigen Instanz von LanguageTool, die in einem Docker-Container betrieben wird. Es wird das Docker Image *meyay/languagetool* verwendet, welches unter Port 8081 zur Verfügung steht. Nach der korrekten Einrichtung des Docker-Containers kann der Instanz eine Anfrage mit dem zu prüfenden Text gesendet werden. Die LanguageTool-Instanz gibt eine Liste gefundener Fehler mit zahlreichen Details zurück:

- *error_text*: Der fehlerhafte Text
- *suggestions*: Eine Liste mit Verbesserungsvorschlägen
- *message*: Eine Beschreibung des Fehlers
- *offset*: Die Startposition des Fehlers im Text
- *length*: Die Länge des fehlerhaften Abschnitts

```
def map_errors_to_runs(errors, run_offsets):
    errors_with_runs = []
    for error in errors:
        # Calculate start and end position of error
        error_offset = error["offset"]
        error_end = error_offset + error["length"]
        affected_runs = []
        # Check if error overlaps with runs
        for run in run_offsets:
            if run["start"] <= error_offset < run["end"] or
            run["start"] < error_end <= run["end"]:
                affected_runs.append(run["run_index"])
        # Add affected runs to error
        error["run_indices"] = affected_runs
        errors_with_runs.append(error)
    return errors_with_runs
```

Listing 6 – Zuordnung Fehler zu Run-Elementen

Mit diesen Daten kann jeder gefundene Fehler den entsprechenden Run-Elementen zugeordnet werden. Für jeden Fehler wird geprüft, welche Positionen der Run-Elemente sich mit dem Fehler überschneiden. Ein Run-Element gilt als betroffen, wenn der Fehler entweder innerhalb seiner Grenzen beginnt oder innerhalb seiner Grenzen endet. Die zutreffenden Run-Indizes werden für jeden Fehler gespeichert.

```
  if len(errors_with_runs) == 0:
      spelling_results = {'correct': True,
                          'message': "Es wurden keine Rechtschreib- oder
                                      Grammatikfehler gefunden.",
                          'runs_idx': runs_idx,
                          'comments_text': comments_text,
                          }
  else:
      message = f"Es wurde{'n' if len(errors_with_runs) > 1 else ''}
      {len(errors_with_runs)} Rechtschreib- oder Grammatikfehler gefunden.
      Nähere Details können der kommentierten Word-Datei entnommen werden."
      spelling_results = {'correct': False,
                          'message': message,
                          'runs_idx': runs_idx,
                          'comments_text': comments_text,
                          }
  return spelling_results
```

Listing 7 – Rückgabe der Ergebnisse der Rechtschreibprüfung

Schließlich wird geprüft, ob Fehler gefunden wurden oder nicht und abhängig davon werden die Ergebnisse und eine Bewertung des Kriteriums Rechtschreibung in strukturierter Form an das Frontend zurückgegeben. Dies beinhaltet zusätzlich die Run-

Elemente, in denen die Fehler gefunden wurden und ein passender Text der als Kommentar eingefügt werden kann.

4.2.3 Unangemessene Wörter

Das nächste Kriterium bei der Sprachüberprüfung beschäftigt sich mit der Identifizierung unpassender oder nicht-wissenschaftlicher Wörter und Phrasen. Dabei greift SciCheck auf eine vordefinierte Liste solcher Wörter und deren Kategorien zurück und markiert deren Vorkommen im Text. Dies wird in der Funktion *check_invalid_words()* umgesetzt, in der die Liste der unpassenden Wörter aus einer JavaScript Object Notation-Datei (JSON) geladen und in das entsprechende Format umgewandelt werden. Der Datensatz besteht aus folgenden Kategorien:

- Füllwort
- Umgangssprache
- Unpräzise Ausdrucksweise
- Verallgemeinerung
- Übertreibung
- Subjektiver Ausdruck

Die einzelnen ungültigen Wörter und Phrasen werden sortiert, sodass längere Begrifflichkeiten zuerst erkannt werden, wenn sich mehrere Bestandteile davon in der Liste befinden. Der Fließtext wird in Kleinbuchstaben umgewandelt, damit Groß- und Kleinschreibung bei der Suche keine Rolle spielt. Bei positivem Befund wird bei einzelnen Wörtern mit einem regulären Ausdruck sichergestellt, dass es sich tatsächlich um ein vollständiges Wort und nicht den Teil eines Wortes handelt. Gefundene Phrasen die Leerzeichen beinhalten werden ohne weitere Prüfung markiert. Wie bei der Rechtschreibprüfung wird eine Zusammenfassung der Ergebnisse angefertigt und zurückgegeben. Das Feedback an Nutzer passt sich dynamisch an den Inhalt an und weist bei gefundenen Fehlern darauf hin, in welchen Kategorien diese gefunden wurden.

4.2.4 Satzkomplexität

Ein weiterer wichtiger Aspekt der Analyse ist die Satzkomplexität. Durch diese Analyse sollen Lesbarkeit und Verständlichkeit der wissenschaftlichen Arbeit in ein für Nutzer verständliches Maß gebracht werden. Die Analyse und eine Einordnung der Satzkomplexität wird mit der Funktion *check_sentence_complexity()* bereitgestellt. Die notwendigen Operationen zur Ermittlung der Satzkomplexität werden auf die zuvor

tokenisierten Sätze und Wörter angewandt. Da die Satzkomplexität bzw. Lesbarkeit eines Textes nicht durch definierte Regeln wie z. B. bei einer Rechtschreibprüfung ermittelt werden kann, verfolgt SciCheck den Ansatz verschiedene Aspekte der Komplexität zu analysieren. Dazu zählen die Identifikation auffälliger Satzlängen, die Berechnung des Lesbarkeitsindexes Flesch-Reading-Ease (FRE) und die Analyse der Satztiefe.

Der Lesbarkeitsindex FRE erklärt die Lesbarkeit eines Textes mit einem Wert zwischen 0 und 100, wobei niedrigere Werte auf einen schwer zu lesenden Text deuten und hohe Werte auf einen leicht verständlichen Text (Readable, 2025). Dabei werden zwei zentrale Merkmale eines Textes bewertet:

- Average Sentence Length (ASL) ist die durchschnittliche Satzlänge, die sich aus der Gesamtzahl aller Wörter geteilt durch die Anzahl der Sätze ergibt.
- Average Syllables per Word (ASW) ist die durchschnittliche Anzahl an Silben pro Wort, die sich aus der Gesamtzahl der Silben geteilt durch die Anzahl der Wörter ergibt.

Zur Berechnung des FRE für die englische Sprache wird folgende Formel angewendet:

$$FRE = 206{,}835 - (1{,}015 \cdot ASL) - (84{,}6 \cdot ASW)$$

Da sich die sprachlichen Strukturen zwischen der deutschen und englischen Sprache deutlich unterscheiden, ist eine direkte Übertragung der englischen Formel auf deutsche Texte nicht sinnvoll. Deutsche Wörter sind im Durchschnitt länger und enthalten mehr Silben als englische. Demnach hat Toni Amstad die Formel für die deutsche Sprache angepasst (Rottensteiner, 2010, S. 3896):

$$FRE_{deutsch} = 180 - ASL - (58{,}5 \cdot ASW)$$

Die für die Formeln notwendigen Daten können mit den tokenisierten Sätzen und Wörtern ausgerechnet werden. Zum Zählen der Silben wird ein einfacher heuristischer Ansatz gewählt, der mittels eines regulären Ausdrucks nach Vokalgruppen innerhalb eines Wortes sucht. Somit kann der Lesbarkeitsindex FRE für Deutsch und Englisch ausgerechnet werden. Nutzer bekommen nicht nur den Lesbarkeitsindex ihrer Arbeit angezeigt, sondern auch eine Tabelle, die dabei hilft, den ausgerechneten Wert zuzuordnen.

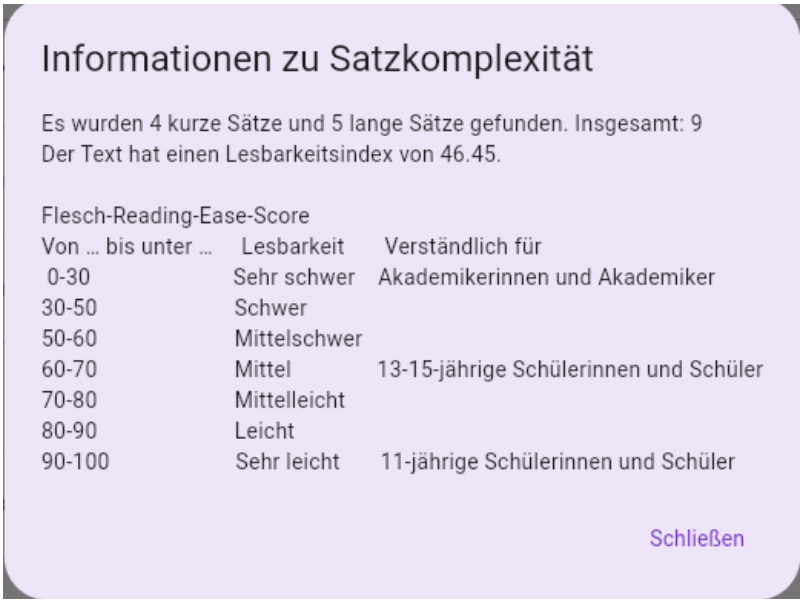

Abbildung 6 – Erklärung der Lesbarkeit im Pop-up

Zur Bewertung der Satzkomplexität und Lesbarkeit wird außerdem die Länge aller Sätze ermittelt und kategorisiert. Es ist zu beachten, dass sowohl kurze als auch lange Sätze in wissenschaftlichen Arbeiten zulässig sind. Deshalb wird lediglich ein Hinweis gegeben, wenn zwei kurze oder zwei lange Sätze unmittelbar aufeinander folgen. Während die vereinzelte Verwendung dieser Sätze legitim ist, kann die Häufung solcher Satzfolgen den Lesefluss beeinträchtigen. Nutzer werden darauf hingewiesen, dass in diesen Fällen eine manuelle Untersuchung notwendig ist.

Da neben Merkmalen wie Satz- und Silbenlänge auch die syntaktische Struktur ein wesentliches Kriterium für die Verständlichkeit von Texten ist, wird diese durch die Analyse der Syntaxbäume aller Sätze bewertet werden. Die Analyse basiert auf der syntaktischen Zerlegung von Sätzen durch die Bibliothek spaCy, welche Dependency Parsing anwendet. Dabei wird jeder Satz in seine syntaktischen Komponenten zerlegt und kann in einem Abhängigkeitsgraphen dargestellt werden.

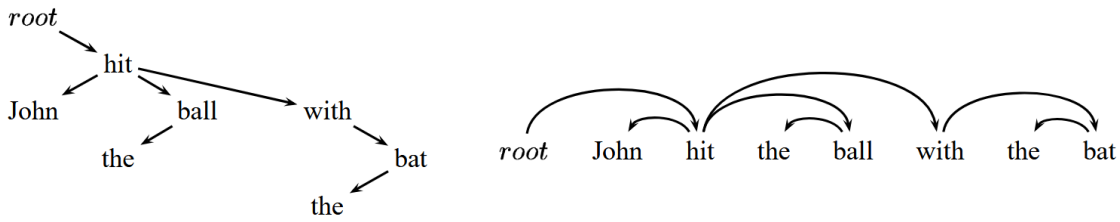

Abbildung 7 – Beispiel eines Abhängigkeitsgraphen (McDonald & Pereira, 2006)

Anhand der Baumtiefe können Informationen erhalten werden, wie verschachtelt und damit potenziell komplex ein Satz ist. Das zur Veranschaulichung einfache Beispiel aus

Abbildung 7 hat eine Tiefe von drei mit dem Verb *hit* als Wurzel des Baums. Abhängigkeiten zwischen Wörtern werden mit gerichteten Kanten angezeigt. Tiefere Verschachtelungen im Satz erhöhen die Tiefe des Baumes. Nachdem mit spaCy die Abhängigkeiten eines Satzes gefunden wurden, kann die Tiefe rekursiv bestimmt werden (Listing 8). Somit können einzelne lange Sätze, die zusätzlich eine hohe Tiefe aufweisen, für Nutzer als potenziell zu komplex angezeigt werden.

```
def get_parse_tree_depth(token):
    # If Token has no children -> depth = 0
    if not list(token.children):
        return 0
    # Depths equals 1 + max depth of all children
    max_depth = 1 + max(get_parse_tree_depth(child) for child in
                                                    token.children)
    return max_depth
```

Listing 8 – Funktion zum Bestimmen der Satztiefe

Am Ende der Sprachanalyse werden die Ergebnisse der verschiedenen Prüfungen zu einem einzigen strukturierten Objekt zusammengeführt. Dieses wird mit den Prüfungen der anderen Kategorien zu einer umfassenden Sammlung der Analyseergebnisse kumuliert. Neben den Daten, die tabellarisch und in Pop-ups angezeigt werden sollen, ist die Übertragung der Metadaten für die Kommentare besonders wichtig.

4.3 Einfügen der Kommentare

4.3.1 Frontend- und Backend-Integration

Das Einfügen der Kommentare in die von Nutzern hochgeladenen Word-Dokumente ist ein wichtiger Schritt, um besseres Feedback zu geben. Mit diesem Prozess können Fehler leicht gefunden und notwendige Änderungen schnell vorgenommen werden. Die Umsetzung umfasst mehrere Schritte, die sowohl das Frontend als auch das Backend betreffen. Im Frontend werden die Analyseergebnisse angezeigt und Nutzer haben die Möglichkeit auszuwählen, welche Kriterien welcher Kategorien kommentiert werden sollen. Diese Auswahl wird an das Backend gesendet, wo die Kommentare in die XML-Struktur von Word eingefügt werden. Danach wird das Dokument an das Frontend zurückgesendet und der Download beginnt.

Zunächst müssen eine Oberfläche und Logik zur Auswahl der zu kommentierenden Kriterien und Kategorien implementiert werden. Dafür werden zunächst Checkboxen für

alle Kriterien und Kategorien, die aus einer JSON-Datei geladen werden, initialisiert. Die tabellarische Darstellung der Ergebnisse ermöglicht, dass ohne weitere Veränderungen eine neue Spalte mit Checkboxen hinzugefügt werden kann.

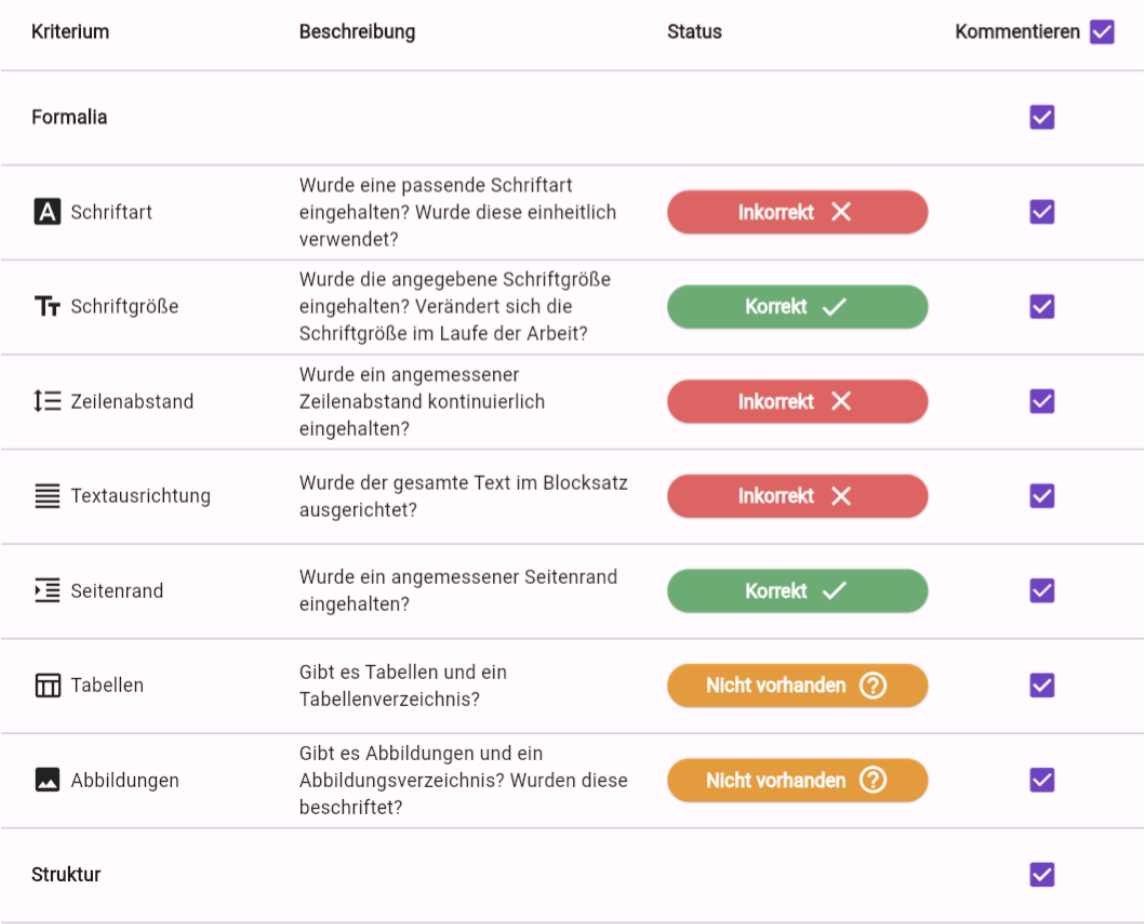

Abbildung 8 – Ausschnitt des Ergebnisbildschirms mit Checkboxen

Nutzer können im Flutter-Frontend interaktiv das Kommentieren einzelner Kriterien, ganzer Kategorien oder aller Inhalte steuern. Bei Veränderungen wird der neue Status der Checkboxen gespeichert. So kann die kommentierte Word-Datei bei einer hohen Fehleranzahl übersichtlich gehalten werden und Fehler können in kleinen Schritten verbessert werden. Am unteren Bildschirmrand kann durch einen Button der Prozess zum Herunterladen der kommentierten Word-Datei gestartet werden. Die Werte der Checkboxen, die Word-Datei und die relevanten Daten der Analyse werden per HTTP-POST-Anfrage an das Python-Backend gesendet.

```
Future<Uint8List> commentFile(
    Uint8List fileData, Map<String, dynamic>? commentData,
    Map<String, bool> criterionCheckboxStates) async {
  // Create multipart request to send all data to backend
  final request = http.MultipartRequest('POST',
                  Uri.parse('$serverUrl/comment-file'),);
  // Add file data
  request.files.add(http.MultipartFile.fromBytes('file', fileData,
      filename: 'upload_comment.docx'));
  // Iterate over commentData, encode and add to fields
  if (commentData != null) {
    commentData.forEach((key, value) {
      if (value is Map) {
        request.fields[key] = json.encode(value);
      } else {
        request.fields[key] = value.toString();
      }});}
  // Add checkbox states to request fields
  request.fields['criterionCheckboxStates'] =
      json.encode(criterionCheckboxStates);
  // Send request and await response from backend
  final response = await request.send();
```

Listing 9 – Senden der Analyse- und Checkboxdaten an das Backend

In *commentFile()* wird eine Anfrage als *MultipartRequest* gestellt, da sowohl Binärdaten des Word-Dokuments als auch die Ergebnisse der Analyse und die Auswahldaten der Checkboxen als JSON-Daten gesendet werden sollen. Die Anfrage wird an das Backend gesendet, welches mit *https://feedback.mni.thm.de/scicheck* angesprochen wird. Für diese Funktion wird der Endpunkt */comment-file* spezifiziert. Im Python-Backend befindet sich die dazugehörige Funktion *comment_file()*, welche die Anfrage beantworten soll, während im Flutter-Frontend auf eine Antwort gewartet wird.

```python
@app.route('/comment-file', methods=['POST'])
def comment_file():
    try:
        # Retrieve uplaoded word file
        file = request.files['file']
        comment_data = {}
        # Iterate over key-value pairs in request form
        for key, value in request.form.items():
           comment_data[key] = json.loads(value)
        …
        # Add relevant comments to new word file
        modified_file_data = comment_utils.comment_handler(roots,
                             roots_processed, file_data, comment_data)
        # Convert file data to base64
        commented_word_bytes = utils.read_word_file_to_bytes(
                               modified_file_data)
        return jsonify({
            'result': 'Commented successfully',
            'word_bytes': commented_word_bytes
        }), 200
    except Exception as e:
        print(f'Error: {str(e)}')
        return jsonify({'error': 'Internal Server Error.
                                  Error commenting Word file.'}), 500
```

Listing 10 – Endpunkt im Backend zum Einpflegen der Kommentare

Mit dem Dekorator *@app.route('/comment-file', methods=['POST'])* wird die Funktion als Endpunkt zum Beantworten der HTTP-POST-Anfrage gekennzeichnet. Die hochgeladene Word-Datei wird abgerufen und zur Weiterverarbeitung vorbereitet. Die vom Frontend als Formfelder gesendeten JSON-Daten werden extrahiert und dekodiert, damit in Python mit den korrekten Datentypen weitergearbeitet werden kann.

4.3.2 Verarbeitung und Einbindung der Kommentare

Die Funktion *comment_handler()* wird als zentraler Punkt aufgerufen, um die Analyseergebnisse als Kommentare in der XML-Struktur von Word einzubetten. Die vom Frontend erhaltenen Daten müssen gefiltert und in ein passendes Format zur Weiterverarbeitung gebracht werden. Die Run-Indizes und dazugehörige Kommentare aller Kategorien und Kriterien werden durchlaufen und in einer neuen Liste gespeichert, wenn die Checkbox des jeweiligen Kriteriums ausgewählt war. Da bei der Analyse alle Run-Indizes notiert werden, selbst bei Folgefehlern, werden im nächsten Schritt Bereiche für diese definiert. Das heißt, es wird ein Startpunkt und ein Endpunkt für jeden Fehler und Kommentar definiert. Danach wird die Spanne der Kommentare im Text festgelegt.

Beim Konvertieren der Daten wird validiert, dass jedem Bereich jeweils Start- und Endpunkt und Kommentarinhalt zugewiesen ist.

Die Start- und Endpunkte enthalten bis zu diesem Zeitpunkt nur die Indizes der Run-Elemente an deren Positionen Kommentare eingefügt werden sollen. Dies ist jedoch nicht ausreichend, da Run-Elemente in *document.xml* nicht indiziert sind. Um vom Index auf ein Run-Element zurückzuschließen wird auf das bei der Vorverarbeitung erstellte Python-Dictionary der Indizes und Run-Elemente zurückgegriffen. So können den Start- und Endpunkten der Kommentare die passenden Run-Elemente zugewiesen werden, die als direkte Ankerpunkte im Dokument fungieren.

Somit sind alle notwendigen Daten vorbereitet, um Kommentare in Word einzufügen. Zunächst wird das Dokument in Python als ZIP-Archiv geöffnet, sodass *document.xml* manipuliert und das gesamte Dokument später gespeichert werden kann. Kommentare in Word benötigen eine eindeutige Zuordnung, deswegen wird eine Kommentar-ID als Integer null initialisiert, mit der Kommentare versehen werden. Es wird über alle Run-Daten und Kommentarinhalte iteriert und eine Kommentarreferenz bei genau den Run-Elementen eingefügt, bei denen Fehler gefunden wurden.

```
def append_comment_reference(start_run_element, end_run_element, comment_id,
                             namespaces):
    # Add comment range start
    if start_run_element is not None:
        comment_start = etree.Element(f'{{{namespaces["w"]}}}
                        commentRangeStart', {f'{{{namespaces["w"]}}}id':
                        str(comment_id)})
        start_run_element.insert(0, comment_start)

    # Add comment range end
    if end_run_element is not None:
        comment_end = etree.Element(f'{{{namespaces["w"]}}}
                      commentRangeEnd', {f'{{{namespaces["w"]}}}id':
                      str(comment_id)})
        end_run_element.addnext(comment_end)
```

Listing 11 – Einfügen der Kommentarreferenz bei den jeweiligen Run-Elementen

Für jede hinzugefügte Referenz wird die Kommentar-ID inkrementiert. Wurden alle Referenzen hinzugefügt, wird eine Liste mit den Kommentar-IDs und den Kommentarinhalten erstellt. Damit können die Kommentarinhalte an das Ende des XML-Baums gehängt werden und die jeweils dazugehörigen Bereiche referenzieren.

```
def append_comments(root, comments_list, namespaces):
    body = root.find('.//w:body', namespaces)
    comments_tag = etree.Element(f'{{{namespaces["w"]}}}comments')
    # Iterate over each comment in comments_list and create comment element
    for comment_id, comment_text in comments_list:
        comment = etree.Element(f'{{{namespaces["w"]}}}comment', {
            f'{{{namespaces["w"]}}}id': str(comment_id),
            f'{{{namespaces["w"]}}}author': "SciCheck",
            f'{{{namespaces["w"]}}}date': datetime.now().strftime(
                                        "%Y-%m-%dT%H:%M:%SZ")})
        # Create comment content and split paragraphs properly
        paragraph = etree.SubElement(comment, f'{{{namespaces["w"]}}}p')
        run = etree.SubElement(paragraph, f'{{{namespaces["w"]}}}r')
        lines = comment_text.split('\n')
        for i, line in enumerate(lines):
            if i > 0:
                etree.SubElement(run, f'{{{namespaces["w"]}}}br')
            text = etree.SubElement(run, f'{{{namespaces["w"]}}}t')
            text.text = line
        comments_tag.append(comment)
    # Append comments to end of XML-tree
    body.getparent().append(comments_tag)
```

Listing 12 – Einfügen der Kommentarinhalte mit Kommentar-ID

Die Funktion *append_comments()* übernimmt diesen Schritt. Sie erzeugt ein neues Kommentar-Element für jeden anzuhängenden Kommentar. Dabei werden Metadaten wie die Kommentar-ID, der Autor und ein Zeitstempel erstellt. Der Kommentarinhalt selbst, wird wie der Standardfließtext mit Absätzen und Runs strukturiert. Wenn der Inhalt eines Kommentars aus mehreren Zeilen besteht, muss ein Zeilenumbruch mit dem Element für Zeilenumbrüche *w:br* eingefügt werden. So bleiben die Kommentare in Word stets übersichtlich. Schließlich werden alle Kommentar-Elemente an das Ende des XML-Baums angehängt.

4.3.3 Speicherung und Rücksendung der modifizierten Datei

Nachdem die Kommentare in das Word-Dokument eingefügt wurden, ist der nächste Schritt das Speichern der modifizierten Datei und das Senden an das Frontend. Für das Speichern der neuen Datei ist die Funktion *save_new_word_file()* zuständig.

```
def save_new_word_file(root, docx_zip):
    document_xml_modified = etree.tostring(root, pretty_print=True,
                            xml_declaration=True, encoding='UTF-8')
    modified_docx = io.BytesIO()
    # Create new ZIP, iterate over all items and copy them
    with zipfile.ZipFile(modified_docx, 'w', zipfile.ZIP_DEFLATED) as
                         new_docx_zip:
        for item in docx_zip.infolist():
            # Use modified document.xml instead of original file
            if item.filename == 'word/document.xml':
                new_docx_zip.writestr(item.filename, document_xml_modified)
            else:
                new_docx_zip.writestr(item.filename,
                                      docx_zip.read(item.filename))
    # Get binary data
    modified_file_data = modified_docx.getvalue()
    return modified_file_data
```

Listing 13 – Speichern der modifizierten Word-Datei

Diese Funktion nimmt die modifizierte XML-Struktur von *document.xml* und das ursprüngliche ZIP-Archiv als Parameter entgegen und erstellt damit ein neues ZIP-Archiv mit den modifizierten Daten. Um dies umzusetzen, wird zunächst das neue *document.xml* in einen String umgewandelt und es wird ein neues ZIP-Archiv erstellt. Es werden alle Dateien des ursprünglichen ZIP-Archivs durchlaufen und in das neue kopiert. Dabei wird jedoch spezifisch nach der Datei *document.xml* gefiltert, sodass statt der ursprünglichen Datei die modifizierte Datei übertragen wird. Schließlich wird das modifizierte Dokument zurückgegeben und in eine Base64-codierte Zeichenkette konvertiert, damit es mittels JSON als Antwort der Anfrage zurück an das Frontend gesendet werden kann.

Im Frontend werden die empfangenen Byte-Daten verwendet, um das kommentierte Dokument herunterzuladen. Die Funktion *downloadWordDocument()* erstellt einen Blob mit den Dokumentdaten und startet den Download.

```
Future<void> downloadWordDocument(
    Uint8List wordBytes, String fileName) async {
  // Create Blob and ensure correct MIME type for .docx files
  final blob = html.Blob([wordBytes], 'application/vnd.openxmlformats-
              officedocument.wordprocessingml.document');
  // Create URL for the blob
  final url = html.Url.createObjectUrlFromBlob(blob);
  // Create anchor element to trigger download
  final anchor = html.AnchorElement(href: url)
    ..setAttribute('download', '$fileName.docx')
    ..click(); // Simulate click to download file
  // Revoke URL to free up resources
  html.Url.revokeObjectUrl(url);}
```

Listing 14 – Download des kommentierten Word-Dokuments

Für den Blob wird eine URL erstellt, der beim Download referenziert wird. Mit einem unsichtbaren HTML-Anker-Element wird ein Klick simuliert, der den Download des Dokuments im Browser auslöst. Nutzer können die kommentierte Word-Datei öffnen und gefundene Fehler einsehen.

4.4 Plagiatsindikatoren

SciCheck führt keine klassische inhaltliche Plagiatsprüfung durch, wie sie in anderen Tools zu finden ist, sondern legt den Fokus auf Inkonsistenzen in Form und Sprache. Ziel ist es Plagiatsindikatoren zu sammeln, die Lehrenden Hinweise geben, an welchen Stellen eine genauere Überprüfung notwendig sein könnte. Demnach ist es wichtig zu erwähnen, dass SciCheck keine garantierte Plagiatsprüfung durchführt, auf die sich bedingungslos verlassen werden kann. Die Plagiatsindikatoren sollen es Lehrenden leichter machen Unstimmigkeiten zu finden und danach eine detaillierte manuelle Überprüfung durchzuführen. Bei den Plagiatsindikatoren wird sich auf verschiedene Merkmale berufen, die zuvor bereits analysiert wurden. Nach Heines (2020, S. 50) gibt es eindeutige Merkmale, die einen Verdacht auf Plagiate entstehen lassen. Einige formale Aspekte davon können mit SciCheck bereits ansatzweise geprüft werden:

- Wechsel des Schreibstils
- Auffällige Erweiterung des Wortschatzes
- Plötzliche fehlerfreie Abschnitte
- Unmotivierte Änderungen der Formatierungen

Besonders Änderungen der Formatierung lassen sich sehr gut mit SciCheck überprüfen. Auch sprachliche Inkonsistenzen können analysiert werden. Besonders wichtig dabei ist

es nicht nur die negativen bzw. inkorrekten Aspekte zu betrachten, sondern auch plötzlich sehr gute bzw. korrekte Vorkommnisse näher zu inspizieren.

4.4.1 Vorverarbeitung

Die Suche nach Plagiatsindikatoren wird für alle Analysen durchgeführt, die eine Konfiguration eines Kurses ausgewählt haben. Dies startet in der Funktion *get_plagarism_indicators()* direkt nach der Analyse der vier Kategorien. Abhängig davon, wie die Ergebnisse der Analyse zur Plagiatserkennung weiterverarbeitet werden können, werden die relevanten Daten übergeben. Teilweise müssen die Daten aus vorherigen Funktionen angepasst werden, damit sinnvolle Schlüsse auf Inkonsistenzen gezogen werden können. Inkonsistenzen werden kapitelweise überprüft, weshalb eine Zuordnung der Run-Elemente zu den jeweiligen Kapiteln notwendig ist.

```
def get_chapter_run_ranges(run_data):
    # Defaultdict to store chapter ranges more easily
    chapter_run_ranges = defaultdict(lambda: {'start': None, 'end': None})
    current_chapter = None
    # Iterate over runs and assign run-ranges to chapters
    for run in run_data:
        chapter_key = (run['chapter_number'], run['chapter_title'])
        if chapter_key != (None, None):
            current_chapter = chapter_key
        if current_chapter:
            if chapter_run_ranges[current_chapter]['start'] is None:
                chapter_run_ranges[current_chapter]['start'] =
                run['run_index']
            chapter_run_ranges[current_chapter]['end'] = run['run_index']
    # Convert defaultdict to a regular dict for easier handling
    chapter_run_ranges = dict(chapter_run_ranges)
    return chapter_run_ranges
```

Listing 15 – Zuordnung Run-Elemente zu Kapiteln

Die Funktion für diesen Prozess ist *get_chapter_run_ranges()*, in der Run-Elemente durchlaufen werden und für jedes Run-Element ein Schlüssel bestehend aus Kapitelnummer und Kapitelname gebildet wird. Das erste Run-Element eines Kapitels wird einmalig als Startpunkt definiert und die darauffolgenden werden als Endpunkt eines Kapitels überschrieben. Dies stellt sicher, dass der Endpunkt immer den Index des letzten dem Kapitel zugeordneten Run-Elements enthält. Wurden alle Run-Elemente zugeordnet, wird der Datensatz zurückgegeben und kann bei zur Kapitelzuordnung verwendet werden.

4.4.2 Plagiatsindikatoren durch Unstimmigkeiten

Die Funktion *check_plagiarism_indicators()* analysiert die Listen mit Run-Indizes der einzelnen Analyseergebnisse relevanter Kriterien. Falls Kapitel signifikante Abweichungen aufweisen, sowohl positive als auch negative, werden diese markiert. Die Kriterien umfassen derzeit Schriftart, Schriftgröße, Zeilenabstand, Textausrichtung, Rechtschreibung und Lesbarkeit.

Mittels der als fehlerhaft markierten Run-Indizes und der zuvor erstellten Zuordnung aus Run-Indizes und Kapiteln wird der Anteil an Fehlern für jedes Kriterium und Kapitel berechnet. Die einzelnen Fehleranteile können mit allen Kapiteln verglichen werden. Für jedes Kriterium wird festgestellt, ob das Dokument insgesamt einen hohen oder niedrigen Fehleranteil aufweist. Folglich können inkonsistente Kapitel identifiziert werden, indem für jedes Kriterium und Kapitel geprüft wird, ob sich das Kapitel signifikant von der ermittelten Mehrheitsnorm unterscheidet. Hat die Mehrheit der Kapitel bei einem Kriterium einen hohen Fehleranteil, aber ein einzelnes Kapitel einen niedrigen Fehleranteil, könnte es sich bei diesem Abschnitt um kopierte Inhalte handeln. Gleichermaßen kann die Mehrheit der Kapitel bei einem Kriterium einen niedrigen Fehleranteil aufweisen, während ein einzelnes Kapitel einen auffällig hohen Fehleranteil haben kann. Auch dies könnte auf kopierte Inhalte deuten, da sich beim Kopieren von Inhalten Formatierungen ändern können. Diese Auffälligkeiten werden in eine Liste von Plagiatsindikatoren aufgenommen. Eine Besonderheit weist hierbei das Kriterium Lesbarkeit auf, da für diese keine fehlerhaften Run-Elemente existieren. Um die Lesbarkeit dennoch verwerten zu können werden Standardabweichung und Durchschnitt der Lesbarkeitsbewertungen aller Kapitel ausgerechnet. Wenn die Standardabweichung einen Schwellenwert von 10 überschreitet, deutet dies auf variierende Lesbarkeit der Kapitel hin. Kapitel deren Lesbarkeitsbewertung weit vom Durchschnitt entfernt sind, werden als Plagiatsindikator aufgenommen.

Mit den Plagiatsindikatoren für jedes Kapitel kann eine Bewertung vorgenommen werden, die angibt, wie stark sich einzelne Kapitel formal und sprachlich von den anderen unterscheiden. Für jedes auffällige Kriterium in einem Kapitel wird diesem ein Wert von 0,2 zugeschrieben, wobei die maximale Punktzahl auf 1,0 beschränkt ist. Sollte ein Kapitel sowohl formal als auch sprachlich auffällige Kriterien enthalten, werden weitere 0,4 Punkte hinzugefügt, da diese Kombination als deutlicher Plagiatsindikator interpretiert werden kann. Mit der daraus resultierenden Bewertung wird Lehrenden im

Frontend verdeutlicht, welches Dokument und welche Kapitel sie näher untersuchen sollten. Die Ergebnisse der Suche nach Plagiatsindikatoren und die geprüfte Word-Datei werden in der MongoDB gespeichert.

4.4.3 Anzeigen der Plagiatsindikatoren im Frontend

Damit die Plagiatsindikatoren für Lehrende übersichtlich dargestellt werden können, wurde im Flutter Frontend ein neuer Bildschirm implementiert. Dieser zeigt sowohl Ergebnisse der Überprüfung als auch die Plagiatsindikatoren und erlaubt das Herunterladen der analysierten Datei. Dozierende können in einem Dropdown-Menü spezifische Kurse auswählen. Sobald ein Kurs gewählt ist, werden alle Teilnehmenden dieses Kurses angezeigt, die einen Eintrag in der MongoDB haben. Für alle Nutzenden können durch Klicken auf das jeweilige Element alle Überprüfungen eingesehen werden.

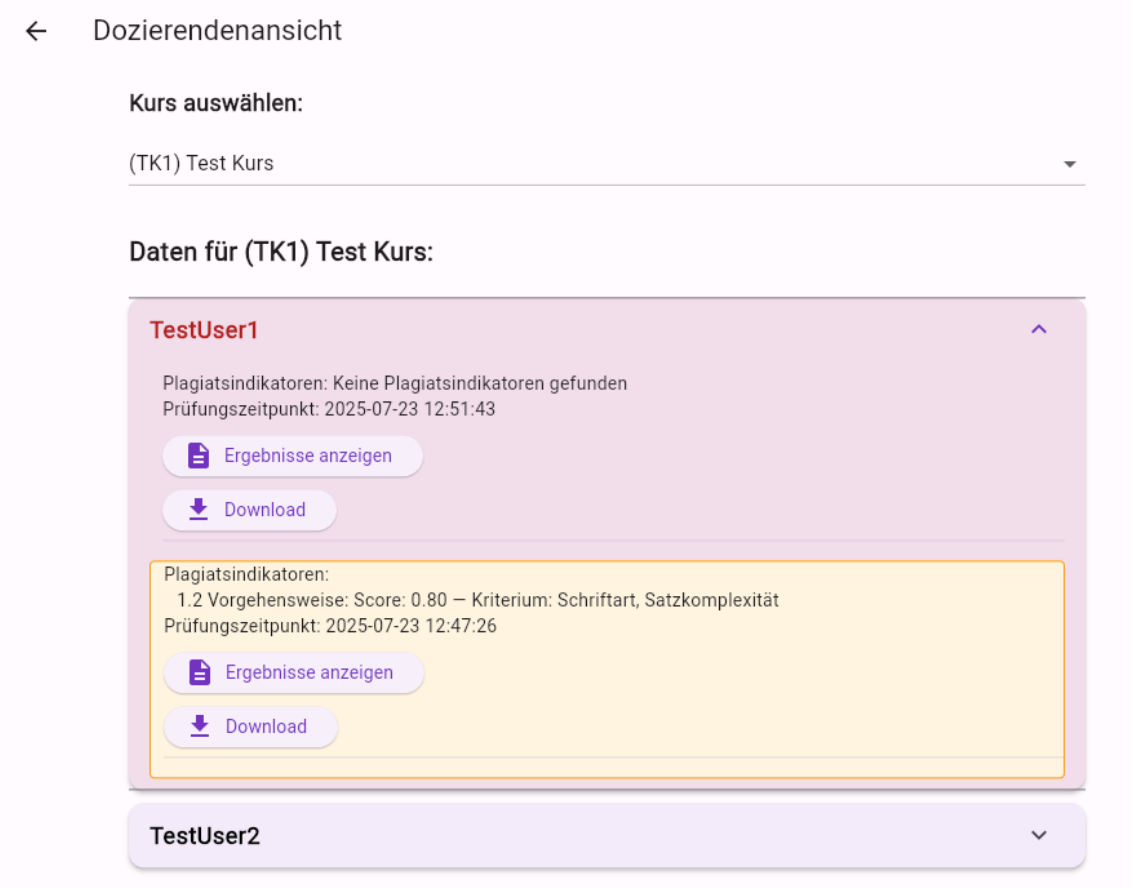

Abbildung 9 – Übersicht für Lehrende

Haben Teilnehmende in einem beliebigen Eintrag Plagiatsindikatoren, wird der Nutzer rot markiert, damit Lehrende auf die gefundenen Plagiatsindikatoren aufmerksam werden. Der einzelne Eintrag ist hervorgehoben und beinhaltet Details zu den Plagiatsindikatoren. Lehrende können einsehen welche Kapitel und Kriterien SciCheck

als auffällig bewertet hat. Darüber hinaus kann eine zusammenfassende Ergebnistabelle für jedes Dokument angezeigt werden. Durch den *Download*-Button kann das überprüfte Word-Dokument für eine manuelle Untersuchung heruntergeladen werden.

5 Evaluation und Einordnung

5.1 Auswertung des Nutzerfeedbacks

Im Rahmen eines Seminars an der Technischen Hochschule Mittelhessen im Sommersemester 2024 wurde SciCheck eingesetzt und mit einem Feedbackbogen evaluiert. Von den insgesamt vierzehn Teilnehmern wurde der Feedbackbogen von sechs Teilnehmenden ausgefüllt. Bei den Teilnehmenden handelte es sich um Studierende aus höherem Fachsemester mit hoher Affinität zu digitalen Tools und Vorerfahrung im wissenschaftlichen Arbeiten. Es wurden Fragen zur Benutzerfreundlichkeit, Verständlichkeit und Nützlichkeit SciChecks gestellt.

Die Ergebnisse zu Fragen bezüglich der Benutzerfreundlichkeit und Verständlichkeit vielen überwiegend positiv aus. Die einzelnen Bedienelemente wurden als logisch angeordnet und leicht zugänglich empfunden. Außerdem ging aus den Daten hervor, dass SciCheck problemlos ohne fremde Hilfe oder Handbuch genutzt werden konnte. Zudem werden keine überflüssigen Eingaben benötigt, lediglich das Hochladen einer Word-Datei. Auch die Darstellung der Informationen wurde positiv bewertet. Die Teilnehmenden empfanden die Menge der Informationen auf dem Bildschirm angemessen und logisch gruppiert. Jedoch konnten Teilnehmende teilweise Informationen nicht dort finden, wo sie erwartet wurden. Bedeutungen angezeigter Texte wurde als verständlich bewertet. SciCheck ließ sich ohne Einarbeitungsaufwand der Teilnehmenden nutzen.

Die Ergebnisse der Nützlichkeit SciChecks zeigen ein gemischtes Bild, das Hinweise auf Verbesserungen gegeben hat, welche im Rahmen dieser Arbeit implementiert wurden. Positiv wurde bewertet, dass SciCheck zu den Anforderungen der Veranstaltung zugeschnitten war einige Teilnehmende fanden, dass ausschließlich hilfreiche und ausreichende Informationen über die einzuhaltenden Vorgaben gegeben wurden. Gemischt aber tendenziell positiv war das Feedback bei der Frage, ob SciCheck dabei hilft formale Vorgaben besser einzuhalten und bei der Quellen- und Literaturarbeit unterstützt. Eine verbesserte Struktur und bessere Arbeit als Ganzes durch SciCheck wurden von den Teilnehmenden ausgeglichen bewertet. Ähnlich war die Bewertung bei der Frage, ob SciCheck bei der Fehlererkennung hilft. Als negativ wurde die Verständlichkeit der Hinweise auf Fehler bewertet. Im Freitext-Feedback wurde die Erkennung verschiedener Elemente als noch nicht fehlerfrei beschrieben. Teilnehmende stellen dar, dass einzelne Kriterien als inkorrekt bewertet wurden, obwohl keine Fehler in

der Arbeit waren. Während der Aufbau und die Art der Informationsvermittlung positiv bewertet wurden, kritisierten Teilnehmer, dass es keine Rückmeldung zu den Stellen der Fehler gab. Trotz der Kritik an fehlerhaften Anzeigen zeigten die Teilnehmenden hohes Interesse an der Weiternutzung des Tools. Sie würden SciCheck gerne weiterhin auch in anderen Veranstaltungen nutzen und an Kommilitonen weiterempfehlen.

Um die im Nutzerfeedback aufgeführten Schwächen SciChecks zu beheben, wurden die Abgaben manuell untersucht und eine Grundlage der Weiterentwicklung geschaffen. Ein zentraler Kritikpunkt waren falsch-positive Rückmeldungen. Diese traten unter anderem auf, da formale Fehler in Bereichen wie dem Deckblatt erkannt wurden, die nicht zum Textteil der Arbeit gehören. Studierende nutzen typischerweise die Standardformatvorlage für den Fließtext, wenn sie ihr Deckblatt formatieren. Zwar wird die Schriftgröße, Textausrichtung und Zeilenabstand angepasst, jedoch sieht SciCheck den Text weiterhin als Fließtext, da keine spezielle Formatvorlage gewählt wurde und findet somit vermeintliche Fehler. Die Prüfung der Abbildungen hatte ähnliche Probleme, da häufig Informationen über die Hochschule als Abbildung eingefügt wurden. Damit dieses Problem behoben werden kann, wurde die gezielte Vorverarbeitung der Dokumente vereinheitlicht und verbessert. Das System erkennt jetzt automatisch, wann der eigentliche Fließtext beginnt, und führt relevante Analysen nur auf diesem Abschnitt aus.

Ein weiterer wesentlicher Aspekt betraf die Nachvollziehbarkeit der gegebenen Bewertungen. Teilnehmende konnten nicht nachverfolgen, an welcher Stelle ihrer Arbeit SciCheck Fehler gefunden hat. Sie bekamen lediglich die Information, ob ein Kriterium als korrekt oder inkorrekt bewertet wurde und einen Hinweis was falsch gemacht wurde, wie z. B. die inkorrekt gewählte Schriftart. Bei der manuellen Untersuchung aller Abgaben hat sich herausgestellt, dass oft echte Fehler vorlagen, jedoch waren diese schwer erkennbar und ohne weitere Hilfe nicht auffindbar. Beispielsweise wurden einzelne Wörter oder Zeichen falsch formatiert. Auch wurden Beschriftungen von Abbildungen nicht mit der passenden Formatvorlage versehen, folglich hat SciCheck eine inkorrekte Schriftgröße erkannt. Durch die manuelle Untersuchung wurden auch Fehler im Systemverhalten SciChecks bei Überprüfungen wie der Schriftart gefunden. Die Logik des Ablaufs bei dieser Überprüfung konnte angepasst und verbessert werden, sodass in den getesteten Dokumenten stets die korrekte Schriftart identifiziert wird. Die genannten Mängel in Bezug auf die Nachvollziehbarkeit und Nützlichkeit SciChecks wurden durch die Einführung des kommentierten Word-Dokuments behoben.

Unabhängig von der Fehlerquelle, können Nutzer unmittelbar einsehen, auf welche Textabschnitte sich eine Rückmeldung bezieht und erhalten in den Kommentartexten zusätzliche Informationen zu Fehlern. Dies erleichtert zum einen die Korrekturarbeit und erhöht zum anderen das Vertrauen in die Korrektheit der Systemrückmeldungen. Gleichermaßen können im Falle von Fehlern des Systems diese leichter nachvollzogen und verbessert werden.

5.2 Vergleich mit anderen Tools

Um SciCheck in die vielfältige Landschaft automatisierter Prüftools und Schreibassistenten einzuordnen, wurde ein Vergleich mit den in Kapitel 3.2.2 vorgestellten Tools vorgenommen. Zu diesen Systemen gehören Grammarly, LanugaeTool, Scribbr, PlagAware, Recite, SciFlow und DocShifter. Während sich Funktionalitäten einiger Tools zwar überschneiden, decken sie insgesamt unterschiedliche Teilbereiche ab, um verschiedene Anforderungen an wissenschaftliche Arbeiten automatisiert zu prüfen. Dabei reicht die Spanne von reiner Sprachprüfung über Form-, Struktur-, Zitations- und Plagiatsanalyse. Die folgende Tabelle fasst die unterschiedlichen geprüften Kategorien der Tools zusammen.

Tool / Kategorie	Formalia	Struktur	Zitation	Sprache	Plagiatsprüfung
SciCheck	Ja	Teilweise	Ja	Ja	Teilweise
Grammarly	Nein	Nein	Ja*	Ja	Ja*
LanguageTool	Nein	Nein	Nein	Ja	Nein
Scribbr	Manuell*	Teilweise*	Ja*	Ja	Ja*
SciFlow	Teilweise	Nein	Teilweise	Ja	Nein
Recite	Nein	Nein	Ja	Nein	Nein
PlagAware	Nein	Nein	Nein	Teilweise	Ja*
DocShifter	Ja*	Teilweise*	Nein	Nein	Nein

Tabelle 1 – Zusammenfassung der Tools und Prüfkategorien

Der Überblick aus Tabelle 1 zeigt, dass SciCheck versucht eine Kombination verschiedener Kategorien abzudecken, um Studierenden bei der Erstellung wissenschaftlicher Arbeiten umfangreich zu unterstützen. Hauptsächliche werden formale Anforderungen an wissenschaftliche Arbeiten gedeckt. Die Übersicht behandelt die einzelnen Prüfkategorien nicht im Detail, sondern bietet lediglich einen allgemeinen Überblick. Wie detailliert die einzelnen Kategorien überprüft werden, wird im Folgenden durch Vergleiche innerhalb der einzelnen Kategorien zwischen den jeweiligen Tools

untersucht. Die kostenpflichtigen Features der Tools wurden nicht getestet, weshalb sich auf die Informationen auf den jeweiligen Webseiten des Tools bezogen wird.

Kriterium / Tool	SciCheck	SciFlow	DocShifter	Scribbr
Schriftart	Ja	Teilweise	Ja*	Manuell*
Schriftgröße	Ja	Teilweise	Nein	Manuell*
Zeilenabstand	Ja	Nein	Nein	Manuell*
Seitenrand	Ja	Nein	Nein	Nein
Textausrichtung	Ja	Nein	Nein	Manuell*
Abbildungs- und Tabellenverzeichnis	Ja	Nein	Nein	Manuell*
Beschriftungen	Ja	Nein	Nein	Manuell*
Verzerrte Abbildungen	Ja	Nein	Nein	Nein
Abbildungen mit niedriger Auflösung	Nein	Nein	Ja*	Nein
Formatierung Abbildungen und Tabellen	Nein	Nein	Ja*	Nein
Farblicher und hervorgehobener Text	Nein	Nein	Ja*	Nein
Korrekte Formatvorlagen	Nein	Nein	Ja*	Nein
Verlinkungen	Nein	Nein	Ja*	Nein
Leere Absätze	Nein	Nein	Ja*	Nein
Einzüge Absätze	Nein	Nein	Nein	Manuell*

Tabelle 2 – Verlgeich Formalia

SciCheck prüft zahlreiche Anforderungen, die für eine korrekte Form wissenschaftlicher Arbeiten relevant sind. SciFlow wirbt mit automatischer Formatierung und gibt Formatvorlagen für verschiedene Arten von Abschlussarbeiten, jedoch werden dabei viele wichtige Kriterien nicht berücksichtigt. Außerdem findet schließlich keine Prüfung statt, ob Formate tatsächlich einheitlich verwendet wurden. Studierende können im Editor SciFlows die gleichen formalen Fehler wie in jedem anderen Texteditor machen. DocShifters DocValidator legt den Fokus nicht auf Abgaben von Studierenden, sondern auf technische Fehler in Word-Dokumenten. Zwar werden typische Layout-Anforderungen nicht berücksichtigt, jedoch prüft DocShifter andere potenzielle Fehler, die SciCheck nicht findet. Scribbr bietet zwar eine Prüfung der Formalia an, jedoch handelt es sich dabei um eine manuelle Prüfung durch Experten und kein automatisiertes Tool.

Kriterium / Tool	SciCheck	SciFlow	DocShifter	Scribbr
Anzahl Kapitel	Ja	Nein	Nein	Manuell*
Unterkapitelstruktur	Ja	Nein	Nein	Unbekannt
Aufzählungsstruktur	Ja	Nein	Nein	Unbekannt
Unzulässige Kapitelnamen	Ja	Nein	Nein	Unbekannt
Reihenfolge Kapitel	Nein	Teilweise	Ja*	Unbekannt
Pflichtabschnitte	Nein	Teilweise	Nein	Manuell*
Wiederholte Kapitelüberschriften	Nein	Nein	Nein	Manuell*
Inhaltliche Strukturprüfung	Nein	Nein	Nein	Manuell*

Tabelle 3 – Vergleich Strukturprüfung

Die Strukturprüfung SciChecks beschränkt sich auf formale Kriterien der Struktur einer wissenschaftlichen Arbeit. SciCheck erkennt keine semantischen Zusammenhänge und kann somit die den inhaltlichen Aufbau eines Textes nicht bewerten. Auch die anderen vorgestellten Tools bieten keine Alternative bei der Strukturprüfung. SciFlow gibt zwar strukturierte Vorlagen für den Aufbau verschiedener Arten wissenschaftlicher Arbeiten, jedoch findet erneut keine Prüfung statt. DocShifters DocValidator prüft die korrekte Reihenfolge der Nummerierungen von Kapiteln. Eine Alternative zur automatisierten Prüfung der Struktur bietet Scribbr durch eine manuelle Korrektur der Arbeit. Dabei werden viele Kriterien überprüft, jedoch können nicht alle Details der Untersuchung eingesehen werden.

Kriterium / Tool	SciCheck	Scribbr		Grammarly	Recite
		KI	Experten		
Validierung Zitationen	Ja	Ja*	Manuell*	Ja*	Ja*
Unterstützung gängiger Zitierweisen	Teilweise	Teilweise	Manuell*	Teilweise*	Teilweise
Zitationen im Text	Ja	Ja*	Manuell*	Ja*	Ja*
Zitationen in Fußnoten	Ja	Nein	Manuell*	Nein	Nein
Konsistenzprüfung Zitationen	Ja	Ja*	Manuell*	Ja*	Ja*
Kapitelzugehörigkeit	Ja	Nein	Nein	Nein	Nein
Literaturverzeichnis alphabetisch	Ja	Nein	Nein	Nein	Ja*
Validierung Literaturverzeichnis	Nein	Nein	Manuell*	Nein	Ja*
Konsistenz Literaturverzeichnis	Nein	Nein	Manuell*	Nein	Ja*
Zuordnung Zitationen und Quellen	Ja	Ja*	Manuell*	Nein	Ja*
Generierung Zitationen	Nein	Ja		Ja	Nein
Generierung Eintrag Literatur-verzeichnis	Nein	Ja		Ja	Nein

Tabelle 4 – Vergleich Zitationsprüfung

SciChecks Zitationsprüfung umfasst Zitationsstile wie APA, Harvard und die deutsche Zitierweise. Zusätzlich können benutzerdefinierte Einstellungen getroffen werden, um den Validierungsprozess anzupassen. Es wird geprüft, ob das Literaturverzeichnis alphabetisch angeordnet ist und ob Zitationen und Verweise im Literaturverzeichnis zusammengehören. Grammarly unterstützt die Prüfung der Zitationsstile APA, MLA und Chicago und erlaubt die Generierung von Verweisen durch Eingabe der notwendigen Informationen in Textfelder. Recite ist auf die Zitationsstile APA und Harvard beschränkt und prüft neben alphabetischer Reihenfolge und Zuordnung die korrekte Formatierung der Einträge im Literaturverzeichnis. Scribbr bietet zwei Formen der Zitationskontrolle: eine automatisierte KI-Lösung und eine manuelle Überprüfung durch Experten. Bei der

automatisierten Prüfung ist jedoch nur APA unterstützt. Für Zitierstile wie MLA, Chicago, Vancouver, Harvard oder andere institutionsspezifische Vorgaben kann eine manuelle Prüfung beantragt werden. Die Generierung von Zitationen im Text und Einträgen für das Literaturverzeichnis erfolgt mit Scribbrs kostenlosen Zitiergenerator. Es stehen sowohl gängige als auch zahlreiche spezifische Zitationsstile zur Verfügung.

Kriterium / Tool	SciCheck	Grammarly	Scribbr	LanguageTool
Rechtschreibung	Ja	Ja	Ja	Ja
Grammatik	Ja	Ja	Ja*	Ja
Zeichensetzung	Ja	Ja	Ja*	Ja
Umgangssprache	Ja	Ja	Ja*	Ja
Füllwörter	Ja	Ja	Unbekannt	Ja*
Subjektive Sprache	Teilweise	Nein	Unbekannt	Nein
Wortwiederholungen	Teilweise	Nein	Unbekannt*	Ja*
Lesbarkeit	Ja	Ja	Ja*	Nein
Deutsche Texte	Ja	Nein	Ja	Ja
Englische Texte	Ja	Ja	Ja	Ja
Weitere Sprachen	Nein	Nein	Nein	Ja
Textgenerierung	Nein	Ja*	Ja*	Ja*
Semantik	Nein	Nein	Nein	Teilweise*

Tabelle 5 – Vergleich Sprachprüfung

SciCheck nutzt zur Rechtschreibprüfung die Funktionalitäten von LanguageTool. Zusätzlich sucht SciCheck spezifisch nach nicht-wissenschaftlichen Ausdrücken im Text. Jedoch erfolgt diese Suche mit einer Liste von Worten und Phrasen, was die Suche nach Subjektiver Sprache einschränkt. LanguageTools Premium Version findet nicht nur komplexere Grammatik-, Stil- und Kommafehler, sondern auch einige semantische Fehler wie Redundanzen oder doppelte Formulierungen. In der kostenlosen Version werden Wordwiederholungen lediglich am Anfang eines Satzes erkannt, während diese in der Premium Version unabhängig ihrer Position im Satz erkannt werden. Auch die Textgenerierung ist nur in der bezahlten Version nutzbar. Grammarly bietet eine umfangreiche kostenlose Sprachprüfung an, jedoch nur auf englischer Sprache. Scribbr bietet eine einfache kostenlosen Rechtschreibprüfung und eine umfangreiche kostenpflichtige KI-Korrektur für wissenschaftliche Arbeiten.

Kriterium / Tool	SciCheck	PlagAware	Grammarly	Scribbr
Textübereinstimmung	Nein	Ja*	Ja*	Ja*
Paraphrasierte Inhalte	Nein	Ja*	Ja*	Ja*
Erkennung KI-generierter Texte	Nein	Ja*	Ja*	Ja*
Datenbankzugriff	Nein	Ja*	Ja*	Ja*
Selbstplagiatsprüfung	Nein	Ja*	Nein	Ja*
Kapitelweise fehlende Quellen	Teilweise	Nein	Nein	Nein
Auffälligkeiten Formalia	Ja	Nein	Nein	Nein
Auffälligkeiten Sprache	Ja	Ja*	Nein	Nein

Tabelle 6 – Vergleich Plagiatsprüfung

Da SciCheck keine klassische Plagiatsprüfung durchführt, werden keine kopierten oder paraphrasierten Texte erkannt, sondern es werden Hinweise auf verschiedene Auffälligkeiten gegeben. Stilbrüche oder unregelmäßige Formatierungen weisen auf potenzielle Plagiate hin. PlagAware ist auf die Plagiatsprüfung spezialisiert und erkennt wörtliche und paraphrasierte Übereinstimmungen, indem Texte mit Internetquellen, wissenschaftlichen Publikationen oder internen Datenbanken verglichen werden. Auch die Lesbarkeit der Texte wird von PlagAware genutzt, um Stilbrüche und somit mögliche Plagiate zu erkennen. Darüber hinaus werden KI-stereotypische Formulierungen und die Relevanz dieser analysiert. Grammarly bietet eine Prüfung basierend auf größtenteils Webseiten, aber auch akademischen Datenbanken von ProQuest, um kopierte Inhalte zu finden. Scribbrs Plagiatsprüfung basiert auf der Basis eines weiteren Plagiatsprüfungstools namens Turnitin. Neben der klassischen Prüfung können Ähnlichkeiten zwischen eigenen Arbeiten und denen anderer Studierender gefunden werden.

Kriterium / Tool	SciCheck	Grammarly	Scribbr	Language Tool	SciFlow	ReCite	PlagAware	DocShifter
Eingebauter Texteditor	Nein	Ja	Ja	Ja	Ja	Nein	Nein	Nein
Import/Upload .docx	Ja	Ja	Ja	Ja	Nein	Ja	Ja*	Ja*
Import/Upload.pdf	Nein	Nein	Ja	Nein	Nein	Nein	Ja*	Nein
Fehlerlokalisierung	Ja	Ja	Ja	Ja	Ja	Ja	Ja*	Teilweise*
Automatische Fehlerkorrektur	Nein	Ja	Ja	Ja	Ja	Nein	Nein	Teilweise*
Ergebnisse sofort einsehbar	Ja	Ja	Teilweise	Ja	Ja	Ja	Ja*	Ja*
Konfigurierbar	Ja	Ja*	Teilweise	Teilweise	Ja	Teilweise	Teilweise*	Nein
Kurssystem	Ja	Ja*	Nein	Nein	Nein	Nein	Ja*	Nein
Kostenlos nutzbar	Ja	Teilweise	Teilweise	Ja	Teilweise	Teilweise	Nein	Nein
Offline verfügbar	Nein	Nein	Nein	Ja	Nein	Nein	Nein	Nein

Tabelle 7 – Vergleich allgemeiner Merkmale

Neben der eigentlichen Prüfung sind auch allgemeine und technische Merkmale beim Einsetzen der verschiedenen Prüftools von großer Bedeutung. Ein wesentliches Unterscheidungsmerkmal ist die Art und Weise wie Prüftools die Texte einlesen. Tools wie Grammarly, Scribbr, LanguageTool und SciFlow stellen einen integrierten Texteditor zur Verfügung, sodass Texte direkt im Browser eingegeben und geprüft werden können. Außer SciFlow erlauben diese Tools zusätzlich den Upload eines Word-Dokuments, um Textinhalte danach im Texteditor abzubilden. Scribbr ermöglicht zusätzlich das Einlesen und Verarbeiten von PDF-Dokumenten. Die Fehlerlokalisierung ist bei den meisten Tools gegeben. Erkannte Probleme werden entweder in einem Texteditor angezeigt und können direkt verbessert werden oder werden wie im Falle von PlagAware und Recite als nicht-editierbarer Text dargestellt. SciCheck hingegen erlaubt das Herunterladen des originalen Dokuments mit Annotationen. Anders als bei der Mehrheit der untersuchten Tools ist bei SciCheck, PlagAware und Recite keine automatisierte Korrektur möglich. Studierende müssen gefundene Fehler selbst korrigieren, anstatt durch einen Klick auf Haken, Fehler automatisch zu beheben. DocShifter gibt die Möglichkeit einige Fehlertypen automatisch korrigieren zu lassen, während andere lediglich markiert werden. Abgesehen von Scribbrs manueller Korrektur, die einige Tage in Anspruch nehmen kann, sind alle Ergebnisse direkt nach Upload oder Eingabe einsehbar. Ein weiterer wichtiger Aspekt SciChecks ist die individuelle Konfigurierbarkeit der zu prüfenden Kriterien. Grammarly erlaubt das Erstellen von Style Guides, um die sprachliche Prüfung zu individualisieren. LanguageTool, Scribbr und SciFlow ermöglichen die Auswahl verschiedener Sprachen für die Korrektur. Neben der Auswahl der Sprache können bei Scribbrs manueller Prüfung individuelle Informationen zu den Prüfkriterien mitgeteilt werden. SciFlow gibt außerdem die Möglichkeit, Zitierweisen und bestimmte Arten wissenschaftlicher Arbeiten für die Vorlage zu definieren. Bei PlagAware können Nutzer einstellen, mit welchen Quellen Texte verglichen werden sollen. Recite erlaubt nur die Auswahl zwischen den Zitierweisen APA und Harvard. Für die didaktische Einbindung in der Lehre verfügen SciCheck, Grammarly und PlagAware über ein Kurssystem, in dem Lehrende Ergebnisse der eingereichten Überprüfungen erhalten. Besonders auffällig beim Vergleich aller Tools ist, dass die meisten Prüfungen kostenpflichtig sind und teilweise nur kostenfreie Basis- oder Testfunktionen zur Verfügung gestellt werden. Alle Prüfungen SciChecks sind kostenfrei nutzbar und können beliebig oft durchgeführt werden. Notwendig dafür ist jedoch wie bei vielen anderen Tools eine Verbindung zum Internet. Nur LanguageTool kann heruntergeladen und offline genutzt werden.

5.3 Limitationen

Zwar bietet SciCheck umfangreiche Funktionalitäten und wurde im Rahmen dieser Arbeit gezielt weiterentwickelt, dennoch bestehen einige Limitationen, die es näher zu erläutern gilt. Ein Teil dieser Einschränkungen insbesondere im Hinblick auf nicht vorhandene Funktionen wurden im vorherigen Kapitel bereits ausgeführt. Da es sich bei SciCheck um eine regelbasierte Prüfarchitektur handelt, können nur Fehler gefunden werden, die zuvor definiert wurden. Diese Regeln und Prüfkriterien wurden im Laufe der Entwicklung an verschiedene geprüfte Seminararbeiten angepasst. Diese Herangehensweise sorgt dafür, dass für neue Arbeiten ähnliche Formatierungen und Strukturen erwartet werden. Bei einigen Abweichungen kann dazu kommen, dass SciCheck Fehler nicht finden kann oder falsch-positive Ergebnisse meldet. Besonders wenn Studierende mit eigenen oder inkorrekten Formatvorlagen arbeiten, kann dies zu Fehlern von SciCheck führen. Die Genauigkeit der Ergebnisse hängt somit davon ab, ob vorgesehene Formatvorlagen korrekt und konsistent genutzt werden.

Eine weitere Limitation ist, dass spezielle Layoutelemente wie Textfelder nicht interpretiert werden können. Einige Studierende verwenden Textfelder, um z. B. Abbildungen oder Tabellen zu beschriften. Diese Abschnitte werden derzeit nicht geprüft, da es zu Problemen bei der Zuordnung und dem Zählen der Run-Elemente kommt. Ähnliche Probleme hat SciCheck bei der Behandlung von Tabellen. Auch die Inhalte dieser werden im aktuellen Entwicklungsstand weder im Hinblick auf Formalia der Tabelleninhalte noch in sprachlicher Hinsicht geprüft.

Während die vier zu prüfenden Kategorien Formalia, Struktur, Zitation und Sprache bereits verschiedene Prüfkriterien aufweisen, fehlen wie in Kapitel 5.2 dargestellt dennoch einige wichtige Kontrollen. Besonders bei der Kategorie Zitation sind einige für wissenschaftliche Arbeiten wichtige Anforderungen nicht abgedeckt. Die Validierungsmethode umfasst zwar einige Regeln, findet aber noch nicht alle Zitationsfehler. Außerdem werden noch nicht alle gängigen Zitationsstile unterstützt wie z. B. IEEE oder Vancouver. Die größte Limitation hierbei liegt bei der Prüfung des Literaturverzeichnisses. Zwar werden Verbindungen zwischen Zitationen im Text und Einträgen im Literaturverzeichnis erstellt, jedoch werden die Einträge selbst nicht überprüft.

6 Fazit und Ausblick

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde das Prüftool SciCheck um neue Funktionalitäten erweitert, die teilweise auf den Ergebnissen einer Testphase in einem Seminar an der Technischen Hochschule Mittelhessen basieren. Durch diese Erweiterungen deckt SciCheck eine breitere Spanne an Anforderungen an wissenschaftliche Arbeiten ab und bietet eine bessere Unterstützung für Studierende.

Durch eine einheitliche Vorverarbeitung der XML-Baumstrukturen wird sichergestellt, dass ausschließlich für die jeweiligen Prüfkriterien relevante Textabschnitte untersucht werden. Somit können falsch-positive Befunde minimiert und insgesamt präzisere Prüfergebnisse erzielt werden. Die systematische Erfassung der Startpositionen der Abschnitte anhand der Run-Elemente, erlaubt ein Einfügen von Kommentaren an der korrekten Stelle im ursprünglich hochgeladenen Word-Dokument. Das Einfügen der Kommentare ist eine wichtige Erweiterung aus dem Studierendenfeedback, um Studierenden transparent zu zeigen, wo ihre Fehler gelegen haben. Somit kommt es nicht mehr dazu, dass Studierende Fehler nicht finden oder nachvollziehen können. Zur Integration dieser Funktionalität mussten alle bisherigen Prüfkriterien angepasst werden, um die Rückgabe positionsbezogener Fehlerdaten mittels indizierten Run-Elementen zu ermöglichen. Dies erlaubt im Frontend die gezielte Auswahl der zu kommentierenden Kategorien und Kriterien mit Checkboxen.

Die sprachliche Überprüfung erfolgt zum Großteil mit einem LanguageTool-Server, der im Feedbacksystem mit einem Docker-Container betrieben wird. Dieser nimmt Textdaten entgegen und gibt gefundene Fehler und deren Position im Text zurück. Darüber hinaus werden weitere stilistische Mängel wie der Einsatz von Füllwörtern, unpräzise Ausdrücke oder Verallgemeinerungen von SciCheck erkannt. Zusätzlich gibt SciCheck Studierenden einen Überblick über die Lesbarkeit ihres Textes.

Basierend auf dem Kurssystem des Feedbacksystems bietet SciCheck einen Bereich für Dozierende, in dem sie innerhalb ihrer Kurse die Ergebnisse der Studierenden einsehen können. Dabei wird nach Plagiatsindikatoren gesucht, die Lehrenden Hinweise auf potenzielle Plagiate geben. Die Plagiatsindikatoren basieren auf den Ergebnissen der durchgeführten Prüfungen von SciCheck. Dazu gehören formale- und sprachliche Auffälligkeiten. Es erfolgt keine garantierte Plagiatserkennung, sondern es werden lediglich Hinweise durch formale nicht-semantische Aspekte gegeben.

Im Vergleich zu anderen online-verfügbaren Prüftools stellt SciCheck eine kostenfreie Alternative mit vielen Funktionalitäten dar. Während viele vergleichbare Tools lediglich Basisfunktionen kostenfrei bereitstellen und erweiterte Überprüfungen kostenpflichtig sind, ermöglicht SciCheck eine umfassende automatisierte Prüfung wissenschaftlicher Texte ohne zusätzliche Gebühren.

6.2 Weiterentwicklung

Aus den bisherigen Erkenntnissen und Limitationen SciChecks können Ansätze für zukünftige Entwicklungen gesammelt werden. Dabei können sowohl bestehende Prüfbereiche verbessert als auch neue Funktionalitäten eingeführt werden.

Eine wesentliche Erweiterung betrifft die Unterstützung weiterer Dateiformate. Neben der Verarbeitung von docx-Dateien ist es wünschenswert andere Dokumentenformate wie .odt (OpenOffice, LibreOffice) oder .tex (LaTeX) prüfen zu können. OpenDocument Text-Dateien ähneln der XML-Struktur von Word-Dateien und alle Prüfungen müssen auf die unterschiedlichen XML-Schemata und Elementnamen angepasst werden. Die Integration von TeX-Dateien ist aufwändiger, da es sich nicht mehr um eine XML-Struktur handelt. Es kann ein Parser für LaTeX entwickelt werden, der den logischen Aufbau versteht und Informationen wie Text oder Formalia extrahieren kann.

Besonderes Potenzial liegt in der Weiterentwicklung der bestehenden Kategorien. Mit der Überprüfung des Literaturverzeichnisses würde die Zitationsprüfung SciChecks eine weitere wichtige Anforderung abdecken. Dafür könnte entweder eine eigenständige Lösung entwickelt oder ein Tool wie AnyStyle genutzt werden, um Einträge des Literaturverzeichnisses in ihre einzelnen Bestandteile zu zerlegen und anschließend zu prüfen. DocShifters DocValidator zeigt, dass weitere formale Kriterien wie Einzüge der Absätze, Seitennummerierungen oder farbliche Texte für Nutzer relevant sind und in Zukunft auch mit SciCheck überprüfbar sein könnten. Mit einer semantischen Untersuchung der Sprache könnte die inhaltliche Qualität der Arbeiten differenzierter beurteilt werden. Durch Prüfung der Kohärenz und Kohäsion könnten die logischen Zusammenhänge der Kapitel und sprachlichen Verknüpfungen der Arbeit bewertet werden. So könnte SciCheck neben formalen Aspekten auch inhaltlich-strukturelle Merkmale berücksichtigen. Um die Arbeit inhaltlich besser nachzuvollziehen, könnte zusätzlich Texterkennung und Computer Vision angewendet werden, um Abbildungen zu analysieren. So könnte unter anderem untersucht werden, ob Abbildungen inhaltlich zum Text passen.

Literaturverzeichnis

Bänsch, A., & Alewell, D. (2020). *Wissenschaftliches Arbeiten* (12. Ausg.). Berlin; Boston: De Gruyter Oldenbourg.

Bennewitz, H., Wendrich, R., & Kuras, J. (2022). Leitfaden zum wissenschaftlichen Arbeiten im Kernstudium. *(8)*. (U. Kassel, Hrsg.) Abgerufen am 14. Juli 2025 von https://www.uni-kassel.de/uni/files/Studium/Alle-Studiengaenge/Lehramt/LAF_Bildungs_und_gesellschaftswissenschaftliches_Kernstudium/Im_Studium/Leitfaden_8._Auflage_2022.pdf

Burstein, J., Kukich, K., Wolff, S., Lu, C., & Chodorow, M. (1998). Computer Analysis of Essays. *NCME symposium on automated scoring*. https://doi.org/10.1002/j.2333-8504.1998.tb01764.x

Cioffi, A., & Peroni, S. (2022). Structured References from PDF Articles: Assessing the Tools for Bibliographic Reference Extraction and Parsing. In *Linking Theory and Practice of Digital Libraries* (S. 425-432). Cham: Springer International Publishing; Imprint Springer. https://doi.org/10.48550/arXiv.2205.14677

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., & Harshman, R. (1990). Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci. (Journal of the American Society for Information Science)*(6), 319-407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Dikli, S. (2006). An Overview of Automated Scoring of Essays. *The Journal of Technology, Learning and Assessment*(1).

DocShifter. (2025). DocValidator: Automate compliance checks for MS Word. Abgerufen am 14. Juli 2025 von https://www.docshifter.com/software-solutions/docvalidator/

Elliot, S. (2003). IntelliMetric: From here to validity. *Automated essay scoring: A cross-disciplinary perspective*, 67-81. https://doi.org/10.4324/9781410606860

Foltýnek, T., Dlabolová, D., Anohina-Naumeca, A., Razı, S., Kravjar, J., Kamzola, L., . . . Weber-Wulff, D. (2020). Testing of support tools for plagiarism detection. *International Journal of Educational Technology in Higher Education*. https://doi.org/10.1186/s41239-020-00192-4

Foltz, P., Laham, D., & Landauer, T. (1999). The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*.

Grammarly Inc. (2025). *Grammarly: Free AI Writing Assistance.* Abgerufen am 19. Juli 2025 von https://www.grammarly.com/

Haines, M. (2020). Leitfaden zum Verfassen einer wissenschaftlichen Arbeit.

Heinze, N., Fink, J., & Wolf, S. (2009). *Informationskompetenz und wissenschaftliches Arbeiten: Studienergebnisse und Empfehlungen zur wissenschaftlichen Recherche im Hochschulstudium.* Augsburg: Universität Augsburg.

Imran, M., & Almusharraf, N. M. (2023). Analyzing the role of ChatGPT as a writing assistant at higher education level: A systematic review of the literature. *Contemporary Educational Technology*, S. ep464. https://doi.org/10.30935/cedtech/13605

Katholische Universität Eichstätt-Ingolstadt (2017). Leitfaden Wissenschaftliches Arbeiten. Abgerufen am 17. Juli 2025 von https://www.ku.de/fileadmin/150305/Studiengangsseiten/Leitfaden_Wissenschaftliches_Arbeiten_im_Fach_Geographie.pdf

Krieger, S., & Labra, J. B. (2023). *Microsoft Learn*. Abgerufen am 09. 07 2025 von https://learn.microsoft.com/de-de/office/dev/add-ins/word/create-better-add-ins-for-word-with-office-open-xml

LanguageTool. (2025). *LanguageTool*. Abgerufen am 19. Juli 2025 von https://languagetool.org

McDonald, R., & Pereira, F. (2006). Online Learning of Approximate Dependency Parsing Algorithms. (S. 81-88). Italien: 11th Conference of the European Chapter of the Association for Computational Linguistics.

Mizumoto, A., & Eguchi, M. (2023). Exploring the potential of using an AI language model for automated essay scoring. *Research Methods in Applied Linguistics*. https://doi.org/10.1016/j.rmal.2023.100050

Page, E. (1968). The use of the computer in analyzing student essays. *International Review of Education*(2), 210-225. https://doi.org/10.1007/BF01419938

Page, E. B. (1966). The Imminence of... Grading Essays by Computer. *The Phi Delta Kappan*, 238-243.

Pietrobon, R., Anand, S., & Jatin, S. (2009). Scientific Writing of Novice Researchers: What Difficulties and Encouragements Do They Encounter? *Academic Medicine*. https://doi.org/10.1097/ACM.0b013e31819a8c3c

PlagAware. (2025). *PlagAware*. Abgerufen am 19. Juni 2025 von https://www.plagaware.com

Readable. (2025). *Flesch Reading Ease and the Flesch Kincaid Grade Level – Readable*. Abgerufen am 17. Juli 2025 von https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/

Ritschl, V., Weigl, R., & Stamm, T. A. (2016). *Wissenschaftliches Arbeiten und Schreiben: Verstehen, Anwenden, Nutzen für die Praxis* (1. Ausg.). Berlin, Heidelberg: Springer Berlin Heidelberg.

Rottensteiner, S. (2010). Structure, function and readability of new textbooks in relation to comprehension. *Procedia - Social and Behavioral Sciences*, S. 3892-3898. https://doi.org/10.1016/j.sbspro.2010.03.611

Rudner, L. M., Garcia, V., & Welch, C. (2006). An Evaluation of IntelliMetric™ Essay Scoring System. *Journal of Technology, Learning, and Assessment*.

Rudner, L., & Liang, T. (2002). Automated Essay Scoring Using Bayes' Theorem. *The Journal of Technology, Learning and Assessment*(2).

SciFlow. (2025). *SciFlow - Schreiben und Veröffentlichen vereinfacht*. Abgerufen am 19. Juli 2025 von https://www.sciflow.net

Scribbr. (2025). *Scribbr - Lektorat & Korrekturlesen deiner Abschlussarbeit*. Abgerufen am 19. Juli 2025 von https://www.scribbr.de/

Strobl, C., Ailhaud, E., Benetos, K., Devitt, A., Kruse, O., Proske, A., & Rapp, C. (2019). Digital support for academic writing: A review of technologies and pedagogies. *Computers & Education*, S. 33-48. https://doi.org/10.1016/j.compedu.2018.12.005

Warburg, S. (2021). Leitfaden zum wissenschaftlichen Arbeiten. Justus-Liebig-Universität Gießen. Abgerufen am 14. Juli 2025 von https://www.uni-giessen.de/de/fbz/fb03/institutefb03/elementarbildung/studium/leitfaden

Wilbers, K., & Kimmelmann, N. (2024). Hinweise zur Erstellung wissenschaftlicher Arbeiten am Institut für Wirtschaftspädagogik. Abgerufen am 14. Juli 2025 von https://www.wipaed.rw.fau.de/files/2024/03/Hinweise-Erstell-wissenschaftl-Arbeiten_Mrz24.pdf

Wytrzens, H. K., Schauppenlehner-Kloyber, E., Sieghardt, M., & Gratzer, G. (2012). *Wissenschaftliches Arbeiten Eine Einführung* (3. Ausg.). Wien: Facultas-Verlag.

Xia, Y., & Zhang, W. (2024). Research on Automatic Detection of Thesis Format Based on Rule Engine. *Advances in Computer and Engineering Technology Research*, S. 87-92. https://doi.org/10.61935/acetr.4.1.2024.P87