Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

KI-gestützte Clusterung von studentischen Programmierlösungen zur Verbesserung automatisierter Feedbackprozesse

AI-supported clustering of student programming solutions to improve automated feedback processes

Gregor Germerodt

Bachelor-Abschlussarbeit

Betreuer: Prof. Dr. Michael Striewe

Trier, 07.07.2025

Vorwort

Ein Vorwort ist nicht unbedingt nötig. Falls Sie ein Vorwort schreiben, so ist dies der Platz, um z.B. das Unternehmen vorzustellen, in der diese Arbeit entstanden ist, oder um den Personen zu danken, die in irgendeiner Form positiv zur Entstehung dieser Arbeit beigetragen haben.

Auf keinen Fall sollten Sie im Vorwort die Aufgabenstellung näher erläutern oder vertieft auf technische Sachverhalte eingehen.

Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

Abstract

The same in English.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1

Einleitung und Problemstellung

Für Lehrkräfte an Hochschulen oder Universitäten kann das Kontrollieren und Bewerten von studentischen Einreichungen je nach Menge zu einer großen Herausforderung werden. Gerade bei einer hohen Anzahl an Abgaben steigt der Korrekturaufwand erheblich, was den zeitlichen Rahmen für individuelles Feedback einschränken kann. Eine Untersuchung zeigte, dass das Kontrollieren und Bewerten von Arbeiten der Hauptfaktor für Arbeitsbelastung und Beeinträchtigung des Wohlbefindens ist (vgl. [JS21]). In der Informatik könnte es sich hier auf Programmieraufgaben beziehen. Dabei müssen Lehrkräfte konsistente Bewertungen abliefern, während jede Abgabe unterschiedliche Syntax und Semantik beinhalten kann.

Eine Lösung dieses Problems bieten etablierte Systeme zur automatischen Auswertung von Programmieraufgaben. Systematische Übersichtsarbeiten zeigen, dass viele Werkzeuge vorwiegend auf Unit-Tests oder statische Analysen setzen, was meist zu eher generischem Feedback führt (vgl. [MBKS]). Die Hoschule Trier benutzt beispielsweise ASB - Automatische Software-Bewertung[1]. Nachdem hierbei Studierende die von der Lehrkraft gestellte Aufgabe bearbeitet haben, können sie online ihre Lösungen hochladen. Das Programm prüft danach nach statischen Kriterien, ob z. B. alle benötigen Dateien hochgeladen wurden, ob sie der Namenskonvention entsprechen, etc. Daraufhin wird das hochgeladene Programm mit Testdaten ausgeführt und geprüft, ob die zu erwarteten Ergebnisse ausgegeben werden. Sollte das nicht der Fall sein, wird eine Fehlermeldung ausgegeben, dass das Programm oder bestimmte Module nicht erwartungsgemäß funktionieren.

Das erzeugte Feedback solcher statischen Systeme dient zur Orierentierung, jedoch weniger zur Fehlersuche, da es recht allgemein gehalten ist. Um die Feedbackgenerierung zu verbessern könnten KI-gestütze Verfahren eingesetzt werden. Damit befassten sich beispielsweise die Autoren der wissenschaftlichen Arbeiten ... (Hier Quellen einfügen und erläutern).

Dazu wurde in dieser Arbeit versucht, einen Schritt vor der Feedbackgenerierung zu entwickeln. Er befasst sich mit der KI-gestützten Clusterung studentischer Programmierlösungen. Dieser Ansatz ermöglicht es für mehrere Einreichungen ein gemeinsames Feedback zu generieren, indem sie nach Ähnlichkeit in Bezug auf Syn-

[1] https://www.hochschule-trier.de/informatik/forschung/projekte/asb

tax und Semantik geclustert bzw. gruppiert werden. Weiterführende Prorgamme oder auch Lehrkräfte könnten dann einen Kandidat pro Cluster wählen, Feedback erzeugen und dieses an alle anderen Teilnehmer des Clusters weiterleiten. Dies könnte eine erhebliche Zeitersparnis zur Folge haben und weiterhin mehr Spielraum für präziseres individuelles Feedback ermöglichen.

2

Theoretische Grundlagen

2.1 Entwicklungswerkzeuge

Das Projekt wurde über die frei verfügbare Entwicklungsumbegung Visual Studio Code (VSC) implementiert und über Git verwaltet. Aufgrund der weltweiten Etablierung und die dadurch gegebene vielfältige Auswahl an Bibilotheken und online verfügbare Hilfestellungen, fiel die Wahl der Prorgammiersprache auf Python. Weiterhin wurde das Textsatzsystem LaTeX unter einer von der Hochschule Trier bereitgestellten Vorlagen zum Verfassen wissenschaftlicher Arbeiten[1] genutzt. Folgend eine Auflistung von Erweiterungen die VSC notwendigerweise oder unterstützend hinzugefügt wurden.

- Notwendig:
  - LaTeX Workshop (Kompilierung, Vorschau, Autovervollständigung)
  - Python
  - Pylance (effizienter language Server für Python)
- Unterstützend:
  - LaTeX language support (Syntax-Highlighting und Sprache für .tex-Dateien)
  - Python Debugger
  - isort (automatische Sortierung von Python-Imports)
  - Git Graph (Visualisierung von Arbeitsverlauf)

2.2 Künstliche Intelligenz

Künstliche Intelligenz (KI) bezeichnet die Wissenschaft und Technik der Entwicklung von Maschinen, die Aufgaben ausführen können, für die normalerweise menschliche Intelligenz erforderlich ist (Russell & Norvig, 2021, S. 1). Der Mensch hat sich damit Systeme geschaffen, um die Kapazitäten menschlicher Intelligenz für bestimmte Aufgaben zu schonen oder zu erweitern. Dabei übernimmt die KI den Teil der Automatisierung monotoner Arbeit, also jenen Part, der durch menschliches Handeln erwiesenermaßen fehleranfällig ist, um konsistente Ergebnisse bereitzustellen. Automatisierung bedeutet in diesem Zusammenhang, dass Maschinen

[1] `https://www.hochschule-trier.de/informatik`

bestimmte Entscheidungs- und Handlungsprozesse eigenständig durchführen, ohne dass eine Person direkt in jeden einzelnen Schritt eingreifen muss. Während sich solche Systeme ursprünglich vor allem in der industriellen Fertigung etablierten, stellt sich zunehmend die Frage, wie sich vergleichbare Konzepte auf andere Bereiche übertragen lassen, wie etwa auf das Bildungswesen und hier speziell auf die Bewertung und Rückmeldung (Feedback) zu studentischen Programmierlösungen. Wie kann eine intelligente Automatisierung Lehrkräfte dabei unterstützen, qualitativ hochwertiges und individualisiertes Feedback zu generieren, ohne jede Lösung einzeln manuell prüfen zu müssen?

In dieser Arbeit wird KI anhand verschiedener Open-Source-Bibliotheken genutzt. Das System kombiniert mehrere KI-Techniken wie

- Deep Learning - ein Teilbereich des Machine Learnings (ML), der künstliche neuronale Netze mit vielen Schichten verwendet, um komplexe Muster in Daten zu erkennen,
- Unsupervised Machine Learning - ein Verfahren, bei denen Modelle ohne beschriftete Trainingsdaten Muster oder Strukturen in den Daten erkennen, z.B. durch Clustering, und
- andere verschiedene Machine Learning Methoden, bei denen Computer aus Beispieldaten eigenständig Muster und Zusammenhänge erkennen, um daraus Vorhersagen oder Entscheidungen abzuleiten.

Das Zusammenspiel dieser Methoden führt letztlich zur Unterstützung der Lehrkräfte zur Feedbackgenerierung, was in den Ergebnissen zu sehen sein wird.

2.3 Verwendete Algorithmen

2.3.1 Einbettung (Embedding)

Einer der ersten Schritte des Programms ist das Einbetten von Quellcode-Text der Programmierlösungen, dessen Erstellung im späterem Verlauf der Arbeit erklärt wird. Der Quellcode-Text wird in numerische hochdimensionale Vektor-Repräsentationen (Embeddings) umgewandelt. Diese numerischen Vektoren fassen semantische und strukturelle Merkmale des Codes zusammen und machen die Daten für anschließende Verfahren wie Dimensionsreduktion, Clustering und Visualisierung nutzbar. Hier wurde eine erste Erwähnung solch eines Verfahrens in der Arbeit von Orvalho et al. (2022, [OJM]) erfasst. Das dort beschriebene Verfahren CodeBERT stellte sich durch weitere Recherche für diese Arbeit als geeignet heraus. CodeBERT ist ein auf die Transformer-Architektur[2] basiertes Modell, das gleichzeitig mit natürlicher Sprache und Programmiercode (u.a. Java und Python) trainiert wurde. Es verwendet dabei machine learning (ML) wie Masked Language

[2] Neuronales Netzwerkmodell, das mithilfe von Self-Attention-Mechanismen die Beziehungen zwischen Elementen in einer Sequenz erfasst und dadurch besonders leistungsfähig für Aufgaben mit Text- oder Code-Daten ist

Modeling[3] und Replaced Token Detection[4], um inhaltlich sinnvolle und strukturierte Vektorrepräsentationen (Embeddings) für Code zu erzeugen (vgl. [FGT+]).

2.3.2 Dimensionsreduktion

Weiterhin wurden Verfahren eingebunden die die durch das Embedding erstellten hochdimensionale Vektoren in ihren Dimensionen reduzieren, um sie ebenso für weiterführende Prozesse wie z. B. zur Clusterung und besonders zur Visualisierung im zwei- oder dreidimensionalem Raum nutzbar zu machen. In dieser Arbeit wurden folende Verfahren benutzt:

- Principal component analysis (PCA) - projiziert hochdimensionale Daten in einen lineareren Unterraum mit geringerer Dimension, indem neue Achsen, entlang derer die Daten am stärksten streuen, berechnet werden und stellt die Daten entlang dieser Achsen dar (vgl. [Kar01]),
- t-distributed stochastic neighbor embedding (t-SNE) - visualisiert hochdimensionaler Daten, indem es berechnet wie ähnlich Punkte zu ihren Originaldaten sind, um sie entsprechend weit auseinander oder nahe zusammen zu platzieren (vgl. [Lau08]), und
- Uniform Manifold Approximation and Projection (UMAP) - nutzt Topologie und Geometrie, um skalierbare, strukturtreue EIinbettungen in niedrigere Dimensionen zu erreichen (vgl. [MHM]).

Diese Algorithmen wurden bevorzugt, da sie aufgrund ihrer Verfügbarkeit in öffentlichen Bibliotheken in das vorgestellte Python Projekt einfach eingebunden werden konnten.

2.3.3 Gruppierung (Clustering)

Das zentral angesprochene Verfahren ist das Clustering. Inspiration für diese Arbeit wurde aus aktuellen Werken entnommen, wie Orvalho et al. (2022), die mit InvAASTCluster ein Verfahren zur Clusterung von Programmierlösungen mittels dynamischer Invarianten-Analyse vorstellen (vgl. [OJM]); aus Paiva et al. (2024) die AsanasCluster, ein inkrementelles k-means-basiertes Verfahren, zur Clusterung von Programmierlösungen für automatisiertes Feedback entwickelt haben (vgl. [PLF24]); und Tang et al. (2024) die Large Language Models[5] (LLMs) und Clustering kombinieren, um personalisiertes Feedback in Programmierkursen zu skalieren (vgl. [TWH+]).

Die Algorithmen dieser Quellen und weitere Recherche dienten zum Kennenlernen und zum späteren Einbinden von Algorithmen, die aufgrund ihrer besonderen Eignung zur Clusterung von Programmieraufgaben herausstachen wie

[3] Trainingsmethode, bei der zufällig Wörter im Text verdeckt werden und das Modell lernen soll, diese fehlenden Wörter richtig vorherzusagen

[4] Trainingsmethode, bei der das Modell lernt zu erkennen, welche Wörter im Text durch andere ersetzt wurden, um so bessere Sprachrepräsentationen zu entwickeln.

[5] auf Textdaten trainierte KI-Modelle, die natürliche Sprache verarbeiten und generieren

- k-means - teilt N Beobachtungen in k Cluster auf, wobei jede Beobachtung zu dem Cluster mit dem nächstgelegenen Mittelwert gehört, der als Prototyp des Clusters dient (vgl. [Mac67]), und
- hdbscan - erweitert des Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Algorithmus, indem es eine Hierarchie von Clustern aufbaut und die stabilen Cluster über unterschiedliche Dichteebenen hinweg extrahiert (vgl. [CMS]),

2.3.4 Visualisierung

Zur Visualisierung der zuvor geclusterten studentischen Programmierlösungen wurden die Python-Bibliotheken pandas und plotly.express verwendet:

- pandas: Dient zur effizienten Verarbeitung und Analyse von tabellarischen Daten. In diesem Fall wurden damit die Cluster-Zuordnungen und die zugehörigen Punktkoordinaten in einer DataFrame-Struktur verwaltet.
- plotly.express: Eine High-Level-Bibliothek für interaktive Diagramme. Sie wurde genutzt, um die studentischen Lösungen als farbige Punkte in einem Streudiagramm darzustellen, wobei die Farbe jeweils das zugehörige Cluster repräsentiert.

So konnte die Qualität und Trennschärfe der Clusterung visuell überprüft werden.

2.3.5 Evaluierung

Das Projekt wurde so erstellt, dass für ein beliebigen Clustering-Algorithmus ein Diagramm erstellt wird, in denen farbige Punkte die entsprechenden studentischen Lösungen repäsentieren. Um diesen Prozess zu bewerten wurden Evaluierungsverfahren etabliert. Nach Halkidi et al. 2001 bewerten interne Clustering-Evaluierungsverfahren die Qualität einer Clusterlösung anhand der Dichte innerhalb der Cluster und der Trennung zwischen den Clustern, ohne dabei externe Referenzdaten heranzuziehen (vgl. [HBV01]). In dieser Arbeit wurden erstmalig Erwähnungen solcher Verfahren in [You24] entdeckt, wobei daraus nur zwei der benutzen Verfahren und erst durch weitere Recherche ein drittes hier eingebunden wurde. Folgende Auflistung beschreibt die benutzten Verfahren:

- Silhouette Score - berechnet für jeden Datenpunkt einen Silhouette-Wert, der die Qualität der Clusterzuordnung anhand der Abstände innerhalb und zwischen Clustern bewertet (vgl. [Rou87])
- Caliński-Harabasz Index - bewertet die Clusterqualität anhand des Verhältnisses von Streuung zwischen und innerhalb der Cluster. Das Verfahren wurde ursprünglich von Calinski und Harabasz (1974, [Cal74]) eingefügt, eine Beschreibung findet sich in [HBV01].
- Davies-Bouldin Index - bewertet die Clusterqualität anhand des Verhältnisses von Intra-Cluster-Distanzen zu den Distanzen zwischen den Clustermittelpunkten. Das Verfahren wurde ursprünglich von Davies und Bouldin (1979,

[DB79]) eingeführt, eine Beschreibung findet sich beispielsweise in der scikit-learn-Dokumentation[6].

[6] https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

3

Vorgehensweise und Implementierung

3.1 Themenfindung

Die zunehmende Digitalisierung und der technologische Fortschritt eröffnen vielfältige Einsatzmöglichkeiten für Methoden der künstlichen Intelligenz (KI). Besonders im Bildungsbereich entsteht dadurch die Chance, Prozesse zu optimieren und Lehrkräfte bei Routineaufgaben zu entlasten.

Vor diesem Hintergrund wurde das Thema dieser Arbeit gewählt. Ziel ist es, das Potenzial von KI-gestützten Verfahren im Kontext der automatisierten Auswertung studentischer Programmierlösungen zu untersuchen. Durch die Clusterung ähnlicher Lösungen können neue Ansätze für Feedbackprozesse entwickelt werden, die eine effizientere und gezieltere Betreuung von Studierenden ermöglichen.

3.2 Recherche und Vorbereitung

Zu Beginn wurden diverse Quellen herausgesucht, die sich im Rahmen dieses Themas bewegen. Besonders oft stach dabei der k-Means Algorithmus heraus oder wie dieser als Grundlage für erweiternde Algorithmen wie den InvAASTCluster (vgl. [OJM]) oder AsanasCluster (vgl. [PLF24]) benutzt wurde, um bessere Ergebnisse zu liefern. Anhand eines Rankings wurde die Relevanz der Quellen festgelegt, um das weitere Vorgehen einzugrenzen.

Andere Methoden wie der Caliński-Harabasz Index oder der Silhouette Score wurden erwähnt, die zur Evaluation des Clusterings dienen. Es wurde klar, dass der Verlauf von studentischen Programmierlösungen bis zu nutzbaren Daten zur Feedbackgenerierung ein mehrschrittiger Prozess ist.

Der folgende Abschnitt, beschreibt die Entstehung des Programms bzw. einer Pipeline, die die benötigten Prozessschritte beeinhaltet, um dies zu erreichen.

3.3 Implementierungsverlauf

3.3.1 Erster Implementierungsabschnitt

Um einen mehrschrittigen Prozess zu vereinen, eignete sich das Prinzip einer Pipeline, in der nacheinander Algorithmen ablaufen, dessen Eingabewerte vom vor-

herigen Algorithmus abhängen und dessen Ausgabewerte dem Nächsten folglich wieder als Eingabe dienen.

Dateien laden

Als erstes musste es eine Möglichkeit geben die Java-Dateien der studentischen Programmierlösungen einlesen zu können. Diese wurden als Datensätze durch den betreuenden Prof. Dr. Striewe bereitgestellt.

Der Datensatz an denen das Programm fortlaufend getestet wurde, bestand aus mehreren Überordnern und final aus drei Java-Dateien und einer Text-Datei, welche den Studierenden vorgegeben waren und vervollständigt werden mussten. Jedoch soll die Menge der Java-Dateien keine überwiegende Rolle spielen.

Das Programm musste also in der Lage sein Ordner zu durchsuchen und Java-Dateien zu erkennen. Dies ermöglichte eine Methode des ersten implementierten Moduls `data_loader.py`. Es extrahierte Quellcode-Text und speicherte pro Datei Code-Schnipsel (Code Snippets) in eine Liste und gab diese an die Pipeline zurück.

Anfangs entstanden durch problematische Zeichen innerhald der Java-Dateien Fehlermeldungen, welche jedoch einfachheitshalber ignoriert werden.

Vektorrepräsentation

Als nächstes folgte ein Modul zum einbetten dieser Code Snippets bzw. zum dessen repräsentieren durch hochdimensionale Vektoren. Dies war notwendig um sie für Clustering-Algorithmen vorzubereiten.

Dafür wurde das Modul `embedding_model.py` erstellt, welches vortrainierte Sprachmodelle aus der Transformers-Bibliothek[1] von Python (hier CodeBERT) und passende Tokenizer, die den Code vorbereitend für den Transformer in Tokens zerlegt, importiert.

Anhand einer Methode werden die Code Snippets in numerische Vektoren umwandelt (Embeddings). Alle enstandenen Embeddings wurde anschließend jeweils an die Pipeline zurückgegeben und in eine Liste abgespeichert.

Clustering

Nun mussten die Embeddings geclustert werden. Das entsprechend erstellte Modul `clustering_engine.py` importierte dafür die beiden Cluster Algorithmen k-Means aus der scikit-learn Bibliothek[2] für maschinelles Lernen und HDBSCAN aus der separaten HDBSCAN Python-Bibliothek[3]. An die Pipeline werden schließlich mit Markierungen (Labels) versehene Cluster zurückgegeben.

[1] `https://huggingface.co/docs/transformers`
[2] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`
[3] `https://hdbscan.readthedocs.io/en/latest/api.html#hdbscan.hdbscan.HDBSCAN`

Evaluationsmetriken

Anschließend wurde ein Evaluationsverfahren eingebaut, um die Qualität der Cluster zu bewerten. Dafür wurde das Modul `evaluation_metrics.py` implementiert. Darin wird jeder Cluster nach den Evaluationsmetriken Silhouette Score[4], Caliński-Harabasz Index[5] und den Davies-Bouldin Index[6] getestet, welche ebenfalls aus der scikit-learn Bibliothek importiert wurden. Zurückgegeben wird ein Dictionary mit den drei Zahlenwerten der Evaluationsmetriken.

Tests, Optimierungen und Dokumentation

Jedes implementierte Modul wurde einzeln getestet. Dazu wurden das Ergebnis und die Zeit, welche für den jeweiligen Prozessschritt notwendig waren durch print-Anweisungen ausgegeben. Dabei stellte sich heraus, dass Embedding und besonders Imports relativ viel Zeit in Anspruch nahmen.

Da die zu testenden Datensätze teilweise aus mehreren hunderten Dateien bestanden, wurde dazu caching der Embeddings eingeführt, um das Testen des Zusammenspiels der einzelnen Module zu beschleunigen. Versuche die Imports durch lazy loading zu beschleunigen waren in diesem Fall nur geringfügig merkbar.

Des Weiteren wurde eine Requirements.txt-Datei erstellt. Diese beeinhaltet sämltiche Information über die aktuell im Projekt benutzen Versionen der importierten Bibliotheken. Sie sorgt, dass für andere Nutzende gleiche Bedingungen wie auch in der Entwicklung herrschen, um ein lauffähiges Programm zu gewährleisten.

Die noch später hinzugefügte Datei HowToInstall.txt dient zusätzlich als Schritt-für-Schritt-Anleitung. Weiterhin wurde eine config.yaml-Datei hinzugefügt. Diese diente als zentrale Ansprechquelle der Pipeline für Parameter.

Dimensionsreduktion

Zu diesem Zeitpunkt war ein Visualisierungs-Modul geplant, um die Clusterungen sichtbar zu machen, doch bevor es sinnvoll eingesetzt werden konnte, wurde noch ein weiterer Prozessschritt eingebunden, das Dimensionsreduktionsverfahren bzw. dessen Algorithmen. Das entsprechende Modul `dimensionality_reducer.py` importiert auch hier Algorithmen aus der scikit-learn Bibliothek, die Principal Component Analysis (PCA)[7] und t-Distributed Stochastic Neighbor Embedding (t-SNE)[8]. Der dritte Algorithmus Uniform Manifold Approximation and Projection (UMAP)[9] stammt aus der eigenständigen UMAP-learn Bibliothek.

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`
[5] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html`
[6] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html`
[7] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`
[8] `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`
[9] `https://umap-learn.readthedocs.io/en/latest/`

Dieser Prozessschritt findet zwischen Embedding und Clustering statt. Er reduziert die Embeddings bzw. die hochdimensionalen Vektoren in ihren Dimensionen (hier zu 2D oder 3D), welche danach zur Clusterung weitergereicht und dadurch auch besser als menschlicher Betrachter in einem Diagramm beobachtet werden können.

Visualisierung

Schließlich wurde noch das Modul `cluster_plotter.py` implementiert, welches anhand der dimensionsreduzierten Embeddings und der aus dem Clustering hervorgegangenen Labels ein statisches Diagramm (engl. plot) in einem separatem Fenster erstellt (Beispiel-Clusterung in Abbildung 3.1). Hierfür wurden aus der Matplotlibs Bibliothek die Plotting-Module pyplot[10] und `mpl_toolkits.mplot3d`[11] importiert, die für 2D- und 3D-Visualisierung (falls gewünscht) zuständig sind.

Die zuvor erstellten Module gleichten sich durch ihre einheitlichen Klassenstruktur, da jedoch für dieses Modul keine Zwischenspeicherung von Zuständen anhand einer Instanz notwendig war, wurden dessen Methoden statisch definiert.

Pipeline

Die Pipeline besteht zu diesem Zeitpunkt aus folgendem Ablauf:

Daten laden → Einbetten → Dimensionen reduzieren → Clustern → Evaluieren → Visualisieren

3.3.2 Zweiter Implementierungsabschnitt

Interaktive Diagramme

In Abbildung 3.2 sind zwar farbige Punkte in unterschiedlicher Anzahl pro Cluster zu erkennen, jedoch wurden keine Labels angezeigt, die Informationen über die Punkte anzeigen sollten. Als vorbereitender Schritt für weiterführende Prozesse zur automatischen Feedbackgenerierung, müssen sie zumindest zum Testen visuell zuordenbar sein.

Aus diesem Grund wurde ein weiteres Modul zur Visualisierung implementiert. Das entstandene Modul `interactive_plot.py` nahm dafür wieder Embeddings, Labels und zustätzlich noch den Namen der aktuellen Java-Datei entgegen, die im `data_loader.py`-Modul gespeichert wurde.

Dazu wurden aus der Python-Bibliothek die Module Pandas[12] und Plotly Express[13] importiert, die einerseits zur Erstellung von Tabellenstrukturen (DataFrames) und andererseits für einfache und interaktiave Diagramme zuständig sind. Ausgeführt, öffnete sich ein Fenster im Webbrowser mit einem von der Gestalung her ähnlichem Diagramm wie in der statischen Variante (3.1).

[10] `https://matplotlib.org/stable/api/pyplot_summary.html`
[11] `https://matplotlib.org/stable/gallery/mplot3d/2dcollections3d.html#sphx-glr-gallery-mplot3d-2dcollections3d-py`
[12] `https://pandas.pydata.org/docs/`
[13] `https://plotly.com/python/plotly-express/`

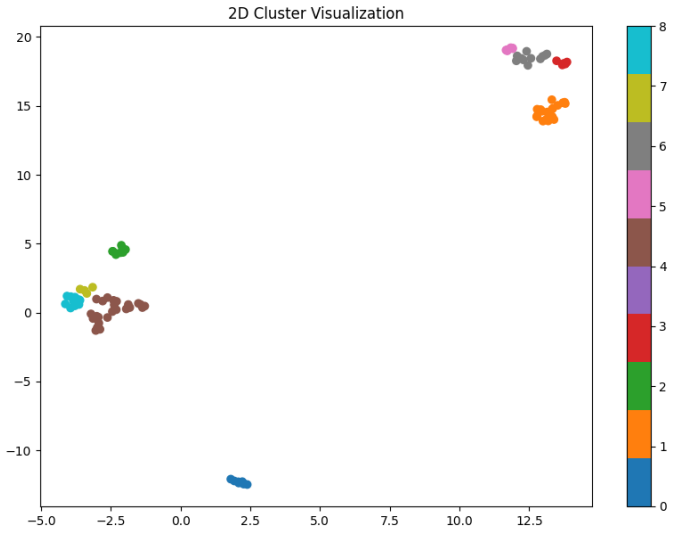

Abbildung 3.1: Zweidimensionales Clustering-Diagramm einer Clusterung von 147 Java-Dateien. Die unterschiedlichen Farben kennzeichnen die Zugehörigkeit der Punkte zu einem Cluster. Am rechten Rand sind die benutzten Farben in einer Farbskala angeordnet.

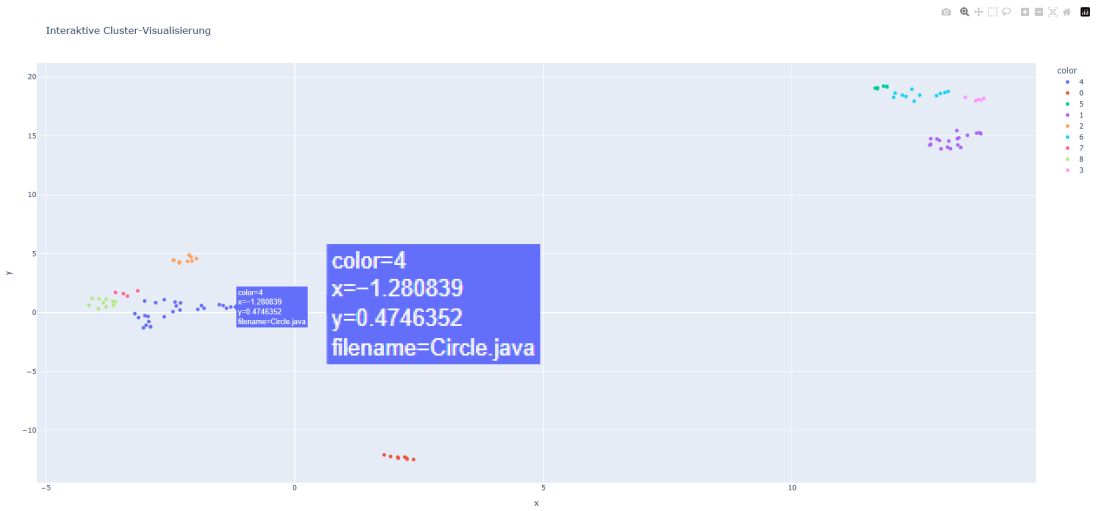

Abbildung 3.2: Interaktives Clustering-Diagramm einer Clusterung von 147 Java-Dateien. Durch das Halten der Maus über die Punkte werden Informationen über sie angezeigt (im Bild wiederholt vergrößert dargestellt). Am rechten Rand sind die Mengen der Cluster angezeigt.

Probleme bei der Cluster-Konsistenz und getroffene Maßnahmen

Beim mehrmaligen Austesten verschiedener Dateimengen und Überprüfen der ausgegeben Punkte, fiel auf, dass Cluster bei größeren Datenmengen nicht mehr konsistent sind, obwohl akzeptable Ergebnisse im Evaluationsmetriken erreicht wurden. Genauer genommen wurden vereinzelt unterschiedliche Dateien in einem gemeinsamen Cluster platziert (gezeigt in 4.2 in Abschnitt Forschungsergebnisse). So kamen beispielsweise Point.java-Dateien in einem Cluster mit sonst nur Circle.java-Dateien vor.

Da Clustering-Algorithmen nach ähnlicher Syntax und Semantik clustern, kann solch ein Verhalten durchaus vor kommen, ist hier jedoch nicht von praktischen Nutzen. Andererseits könnten ebenso Fehler in den Algorithmen oder Inkonsistenzen in den gegebenen Datensätzen die fehlerhafte Clusterung verursachen. Selbst Dateien die abhängig von der gestellten Aufgabe zu den Einreichungen bereits gegeben waren, nicht bearbeitet werden sollten und überall identisch waren, wurden in verschiedenen Clustern angeordnet.

Als erste Maßnahme gegen dieses Problem, wurden die nicht zu bearbeitenden Dateien ignoriert, indem nicht mehr allgemein nach.java-Dateien gesucht wird, sondern nur noch nach bestimmten Namen.

Im weiterem Verlauf wurde zudem in der Pipeline eine Schleife ergänzt, die die Prozesse ab Embedding bis zur Visualisierung für die gewünschten, in der Konfigurationsdatei festgelegten Namen bzw. Java-Dateien wiederholt. Dadurch werden gemischte Cluster verhindert und für jede unterschiedene Java-Datei ein Diagramm erstellt.

Weiterhin ist beim Testen einer niedriger Anzahl von Dateien ($n \leq 4$) aufgefallen, dass der Embedding-Algorithmus nicht mehr funtkioniert, jedoch ist solche eine Clusterung ohnehin nicht von Nutzen.

Experimentierungspipeline

Um die Evaluationsmetriken einfach testen zu können, wurde eine separate Experimentierungspipeline erstellt. Der Vorteil bestand darin, dass die verschiedenen Kombinationen der Dimensionsreduktions- und Clustering-Algorithmen mit ihren Parametern über Dictionaries innerhalb der Pipeline definiert wurden. Die Evaluationsergebnisse wurden anschließend in einer CSV-Datei im Projektverzeichnis festgehalten.

In den Tabellen 4.1 und 4.2 im Abschnitt Forschungsergebnisse wurde gezeigt, welche Algorithmen-Kombination für verschiedene Dateimengen geeignet sind.

Visuelle Erweiterung

Um den Punkten im Diagramm mehr Informationen entnehmen zu können, wurde neben den Dateinamen nun noch der Name des Einreichungsordners hinzugefügt. Da die erreichten Punktzahlen der Einreichungen Teil des Ordnernames sind, konnten die Clusterungen jetzt besser nachvollzogen und überprüft werden. Weiterhin wurde das `interactive_plot.py`-Modul um die Option das Diagramm

als dreidimensionale Umgebung darzustellen erweitert. Sollten mehrere Cluster im 2D-Diagramm aufeinanderliegen, so kann die dritte Achse bessere Einsicht gewährleisten.

3.3.3 Dritter Implementierungsabschnitt

Konkatenation gesuchter Dateien

Auch wenn für jede durch Namen getrennte Art Datei ein separates Diagramm erstellt wird, besteht die Möglichkeit dass die vollständige Einreichung bzw. Lösung zu den gestellten Aufgaben gesamt betrachtet werden sollte, da es sonst zu verminderter Information führen könnte. Um die Dateien als ein Ganzes zu betrachten, wurde das `data_loader.py`-Modul um eine Konkatenations-Funktionalität ergänzt. Die entstandene Methode konkateniert alle gesuchten Java-Dateien eines Einreichungsordners. Der im Diagramm gezeigte Dateiname eines Punktes, setzt sich nun aus den konkatenierten Namen der Dateien zusammen. Die Schleife in der Pipeline die die Prozessschritte für jede gesuchte Art Datei wiederholte, wurde daraufhin entfernt.

Probleme bei der Cluster-Visualisierung und Punktzuordnung

Theoretisch wäre das Programm in der Lage, Embeddings ohne Dimensionsreduktionsalgorithmen zu verarbeiten. Zum Testen wird jedoch weiterhin eine geeignete Visualisierung verwendet, bei der allerdings durch die Anzeige der Ordnernamen pro Punkt ein weiteres Probleme auffiel. So werden konkatenierte Dateien in einen Cluster gesteckt, dessen Ordnernamen sich durch stark variierende Punktzahlen unterscheiden, wie in Abbildung 3.3 zu sehen ist. Auch hier kann solch ein Verahlten durchaus vorkommen, ist aber nicht von praktischen Nutzen, da sich das zu generierende Feedback nach den erreichten Punktzahlen unterscheiden sollte.

Punktzahlenintervalle

Um diesem Problem entgegenzuwirken, wurde erneut die Vorgehensweise der wiederholten Prozessschritte mittels einer Schleife angewendet, indem nach Ebenen bzw. Punktzahlenintervallen (Score-Bins) geclustert wird. Es können beliebig viele Intervalle definiert werden (z. B. 0-49, 50-79, 80-89, 90-100), um das Clustering weiter zu präzisieren. Dafür wurde dem `data_loader.py`-Modul eine Methode hinzugefügt, die die Punktzahlen aus den Ordnernamen, in denen die Java-Dateien enthalten sind, ausliest.

Weiterhin wurde das Modul `score_binning.py` entwickelt, die den ausgelesenen Punktzahlen ein Intervall zuordnet. Anschließend werden in der Pipeline für jedes Intervall bzw. für alle Dateien die zu diesem Bereich gehören, die Prozesschritte ab Embedding wiederholt. Anstatt daraufhin pro Intervall ein Diagramm zu erstellen, wird zur besseren Übersichtlichkeit alle separaten Clusterungen in einem Diagramm visualisiert. Die Ergebnisse der Evaluationsmetriken könnten dadurch leicht verfälscht.

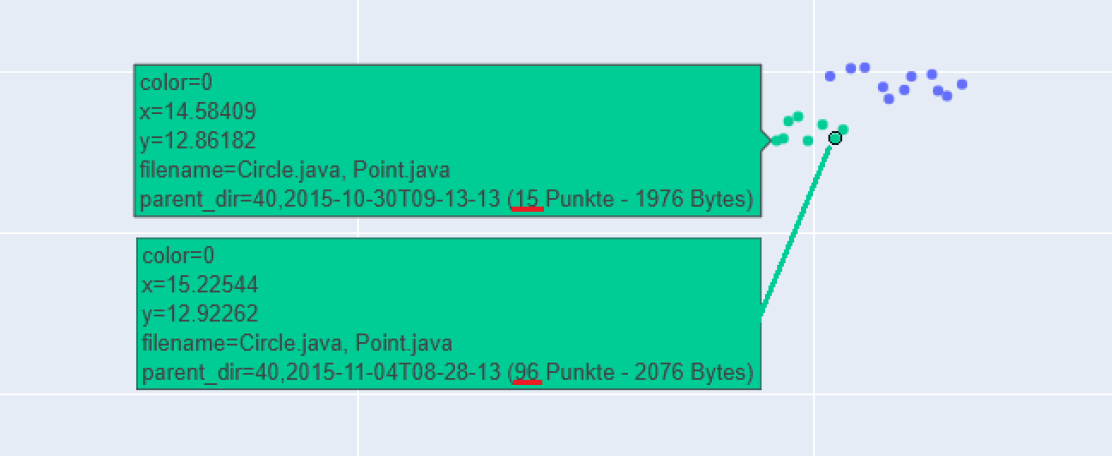

Abbildung 3.3: Interaktives Clustering-Diagramm einer Clusterung von 80 bzw. 40 konkatenierten Java-Dateien. Die rote Unterstreichung zeigt das Problem gemischert Cluster mit unterschiedlichen Punktzahlen.

Verarbeitung von Cluster-Ergebnissen

Um die Dateimenge weiter einzugrenzen, wurde ein zusätzlicher Parameter der Konfigurationsdatei hinzugefügt, der zu exkludierende Dateinamen speichert. Zudem wurden die Cluster-Ergebnisse mittels eines `report_generator.py`-Moduls für weiterführende Prozesse vorbereitet, indem sie nach Score-Bins und Cluster sortiert, in eine CSV-Datei abgespeichert werden. Da es pro Studierenden mehrere Einreichungen geben kann, wurde zudem der Überordnername ergänzt. Folgende Auflistung zeigt diese Struktur. Ein Element eines Clusters besteht aus: Überordner - Einreichungsordner - (konkatenierte) Java-Dateien.

Score-Bin: 0-49

- **Cluster 0:**
  - 0D01AAB9 – 2015-11-03T19-55-35 (28 Punkte – 1485 Bytes) – Circle.java, Point.java
  - 0D1C6395 – 2015-10-30T09-10-34 (15 Punkte – 1976 Bytes) – Circle.java, Point.java
- **Cluster 1:**
  - 0B87094E – 2015-11-03T13-16-47 (0 Punkte – 1983 Bytes) – Circle.java, Point.java
  - ...

  ...

Im folgendem Abschnitt wird die finale Implementierung bzw. dessen Module vorgestellt.

3.4 Finale Implementierung

In Abbildung 4.1 ist die Ordnerstruktur des Projekts zu sehen. Cache-Dateien, welche automatisch von VSC generiert wurden, wurden nicht mit aufgelistet.

3.4.1 Pipeline

Die Pipeline besteht zu diesem Zeitpunkt aus folgenden Hauptbestandteilen:
Daten laden → Punktzahlenintervalle → Einbetten → Dimensionen reduzieren → Clustern → Evaluieren → Visualisieren → Bericht erstellen

3.4.2 Ablauf in Pipeline

Zuerst wird die Konfigurationsdatei eingelesen (Abschnitt 3.4.3, # load config).

Danach werden für die Funktionalität des Programms unwichtige Warnungen ignoriert, um die Ausgabe übersichtlich zu halten (# ignore warnings). Sie beziehen sich auf auf veraltete Parameter, eingeschränkte Parallelverarbeitung durch feste zufällige Status (Random States) und automatische Anpassung von Nachbarparametern bei kleinen Datenmengen.

Als nächstes wird der Eingabepfad bestimmt, indem der Pfad der studentischen Einreichungen mit dem des aktuellen Projekts vereint wird (# dynamically determine path).

Dieser Pfad wird daraufhin neben anderen Parametern zum laden der studentischen Einreichungen genutzt und als Code-Schnipsel abgespeichert (Abschnitt 3.4.4, # load data).

Danach werden die Punktzahlenintervall vorbereitet (Abschnitt 3.4.5, # score bins). Die aus den Einreichungsorder extrahierten Punktzahlen werden einem Punktzahlenintervall eingeordnet und sortiert diese dann alphabetisch.

Nun werden vorbereitend ein Embedding-Objekt und der Name der zwischengespeicherten (cached) Embeddings erstellt (# Embedding object). Zudem werden Listen angelegt, die Informationen über die Einreichungen aus der darauffolgenden Schleife speichert, um sie später in einem Diagramm darstellen zu können (# save all information . . .).

Folgend wird über alle Punktzahlenintervalle iteriert (# filter solutions by . . .). Darin werden die Indizes aller Einreichungen über eine Liste gemerkt, dessen Punktzahl im aktuellen Punktzahlenintervall liegt. Bei einer niedrigen Dateimenge, kann nicht sinnvoll geclustert werden. Darum werden Punktzahlenintervalle ignoriert, zu denen weniger als vier Einreichungen zugeordnet werden konnten. Danach werden für die ausgewählten Indizes die passenden Code-Schnipsel, Datei- und Ordnernamen aus den jeweiligen Listen herausgefiltert und in neue Listen für diese Punktzahlenintervalle gespeichert.

Im nächsten Teil der Schleife werden die Embeddings berechnet (Abschnitt 3.4.6, # Embedding in loop). Es wird geprüft ob bereits ein Embedding-Cache vorliegt und ob dessen Größe die Anzahl der aktuell zu prüfenden Dateimenge einrahmt. Wenn ja, werden die Embeddings aus dem Cache geladen, ansonsten werden sie neu berechnet und im Cache abgespeichert.

Danach werden nacheinander die Embeddings in ihren Dimensionen reduziert (Abschnitt 3.4.7, # Dimension reduction), die reduzierten Embeddings geclustert (Abschnitt 3.4.8, # Clustering of the ...) und diese dann evaluiert (Abschnitt 3.4.9, # Evaluation). Anschließend werden alle Daten in der vor der Schleife definierten Listen angehängt (# save all information ...).

Nach der Schleife werden die gesammelten Daten in einem Diagramm visualisiert (Abschnitt 3.4.10, # Advanced interactive plot). Daneben kann noch auf das statische Diagramm zugegriffen werden (dient als Reserve und wird hier nicht genauer erklärt).

Als letztes wird der im Implementierungsverlauf erwähnte Bericht über das Clustering erstellt (Abschnitt 3.4.11, # Reporting).

3.4.3 `config.yaml`

Hier sind alle Parameter gehalten, die in der Pipeline für die verschiedenen Algorithmen gebraucht werden. Es sind die drei vorgestellten Dimensionsreduktionsalgorithmen, sowie die beiden Clustering-Algorithmen mit vollständigen möglichen Parametern angegeben. Weitere Einstellungen werden durch den Parameter "data"getroffen, in den neben Punktzahlen-Intervalle (Score-Bins), die gesuchten Dateien, zu exkludierende Dateien, etc. angegeben werden können. Um Algorithmen zu wechseln, müssen sie entsprechend ent- und auskommentiert werden.

3.4.4 `data_loader.py`

Die Klasse nimmt beim Erstellen den Dateinamen, den Pfad zu den studentischen Einreichungen und die zu exkludierenden Datei- und Ordnernamen entegegen. Zudem werden Listen zum speichern aller Dateinamen und Einreichungsordner gespeichert, um sie später im Diagramm anzeigen zu können. Folgende weitere Methoden sind gegeben:

- load_code_files(): Lädt alle passenden Dateien aus den Einreichungsordnern, liest deren Inhalt, speichert Datei-, Einreichungsordner- und Überordnernamen und entscheidet, ob die Inhalte der Dateien zusammengeführt werden sollen.
- get_scores(): Liest aus den Ordnernamen die Punktzahl heraus und gibt sie als Liste zurück.
- get_filenames() und get_parent_dirs(): Ermöglichen Zugriff auf die zwischengespeicherten Datei-, Einreichungsordner- und Überordnernamen.

3.4.5 `score_binning.py`

Die Klasse enthält die statische Methode bin_scores(), die eine Liste von Punktzahlen aus get_scores() des `data_loader.py`-Moduls nimmt und anhand der definierten score_bins aus der Konfigurationsdatei jeder Punktzahl ein passendes Label (z. B. „90-94“) oder „Unassigned“zuweist, falls sie in kein Intervall passt.

3.4.6 `embedding_model.py`

Die Klasse nimmt beim Erstellen den Namen des Embedding-Modells entgegen und speichert Tokenizer und Transformermodell (hier CodeBERT). Diese erhalten über die Methode get_embedding() Code-Schnipsel, zerlegen sie weiter in Einheiten und wandeln sie in einen numerischen Vektor (Embedding) um. Anschließend wird das Embedding in einem Numpy-Array zurückgegeben.

3.4.7 `dimension_reducer.py`

Ein Objekt der Klasse nimmt den Namen des Algorithmus und ein Dictionary mit Parametern entgegen, welche den Algorithmus manipulieren. Die Methode reduce() nimmt Embeddings entgegen, wendet dann den gewählten Algorithmus darauf an und gibt dimensionsreduzierte Embeddings zurück.

3.4.8 `clustering_engine.py`

Hier werden die dimensionsreduzierten Embeddings aus der Klasse des Moduls `dimension_reducer.py` geclustert und deren Cluster-Zugehörigkeit als Labels zurückgegeben. Der Aufbau ist zudem weitestgehend gleich (siehe Abschnitt 3.4.7).

3.4.9 `evaluation_metrics.py`

Die Klasse enthält die statische Methode evaluate(), welche die dimensionsreduzierten Embeddings und die Clustering-Labels entgegennimmt. Wenn mehr als ein Label bzw. Cluster vorhanden ist, werden die importieren Evaluationsmetriken darauf angewendet, dessen Ergebnisse in einem Dictionary gespeichert und anschließend zurückgegeben.

3.4.10 `advanced_iteractive_plot.py`

Die Klasse enthält eine statische Methode plot(), die die dimensionsreduzierten Embeddings, Clustering-Labels, Datei- und Ordnernamen und Punktzahlenintervalle entgegennimmt. Danach werden diese Daten in eine Tabellenstruktur (DataFrame) umgewandelt und damit dann ein Streudiagramm erstellt. Durch entsprechendes ent- und auskommentieren, kann zwischen zwei- und dreidimensionalen Diagrammen gewechselt werden.

3.4.11 `report_generator.py`

Die Klasse enthält eine statische Methode generate_report(), welche die Datei- und Ordnernamen, Clustering-Labels, Punktzahlenintervalle und den Ausgabepfad entgegennimmt. Über eine Schleife wird für jedes Punktzahlenintervall ein neues Dictionary und darin für jeden Cluster eine Liste erstellt. Danach werden diese Container erneut durchgegangen, um damit eine CSV-Datei nacheinander zu füllen. Anschließend wird die Datei im Ausgabepfad abgespeichert.

4

Forschungsergebnisse

4.1 Rangliste der Evaluationsverfahren

In den folgenden beiden Tabellen 4.1 und 4.2 sind die Ergebnisse der Evaluationsverfahren für den zweiten Implementierungsabschnitt enthalten, welche nach Rang bzw. aufsteigend nach bester Algorithmus-Kombination geordnet sind. Der Rang ergibt sich dabei aus dem Mittelwert der normalisierten Werte der Metriken (Davies-Bouldin Index invertiert normalisiert). Die optimalen Werte für die verschiedenen Verfahren sind wie folgt:

- Silhouette Score: 0,5 oder höher
- Caliński-Harabasz Index: höchster Wert aus allen Tests
- Davies-Bouldin Index: niedrigester Wert aus allen Tests (gut zwischen 0,3 und 0,7)

Tabelle 4.1 legt nahe, dass UMAP das geeigneteste Dimensionsreduktionsverfahren ist, unabhängig vom Clustering-Algorithmus, wobei t-SNE und PCA mittelwertige Ergebnisse mit k-Means und schlechtere Ergebnisse mit HDBSCAN liefern. Tabelle 4.2 zeigt, dass sich die Eignung nicht großartig ändert. So sind sowohl für kleine, als auch große Dateimengen beide Clustering-Algorithmen zusammen mit UMAP geeignet.

Kombination	**Silhouette**	**Calinski-Harabasz**	**Davies-Bouldin**	**Rang**
umap_kmeans	0.763	3593.558	0.317	1
umap_hdbscan	0.728	3248.739	0.425	2
tsne_kmeans	0.640	173.360	0.273	3
pca_kmeans	0.609	168.268	0.444	4
pca_hdbscan	0.573	85.872	0.777	5
tsne_hdbscan	0.600	2.318	5.856	6

Tabelle 4.1: Evaluationsergebnisse mit 40 Dateien.

Kombination	Silhouette	Calinski-Harabasz	Davies-Bouldin	Rang
umap_kmeans	0.595	6554.361	0.390	1
umap_hdbscan	0.671	1398.606	0.608	2
tsne_kmeans	0.579	490.225	0.609	3
pca_kmeans	0.716	36.378	0.863	4
pca_hdbscan	0.408	201.754	0.780	5
tsne_hdbscan	0.421	267.712	0.814	6

Tabelle 4.2: Evaluationsergebnisse mit 320 Dateien.

4.2 Clustern unterschiedlicher Dateien in einem Diagramm

Folgende Abbildungen zeigen Clusterungen mit steigender Anzahl Dateien, die sich auf den zweiten Implementierungsabschnitt beziehen. Dabei trat das Problem auf, dass bei größeren Dateimengen die Wahrscheinlichkeit für gemischte Cluster, also mit unterschiedlichen Dateien in einem Cluster anstieg. In den Abbildungen 4.1 und 4.2 ist die Clusterung von 160 Point.java- (orangene Punkte) und 160 Circle.java-Dateien (blaue Punkte) und das beschriebene Problem zu sehen. Die genutzte Algorithmen-Kombination war UMAP mit HDBSCAN. Andere Kombinationen wie PCA mit k-Means ergaben deutlich andere Ergebnisse, jedoch stieg hier die Anzahl der gemischten Cluster ebenso an.

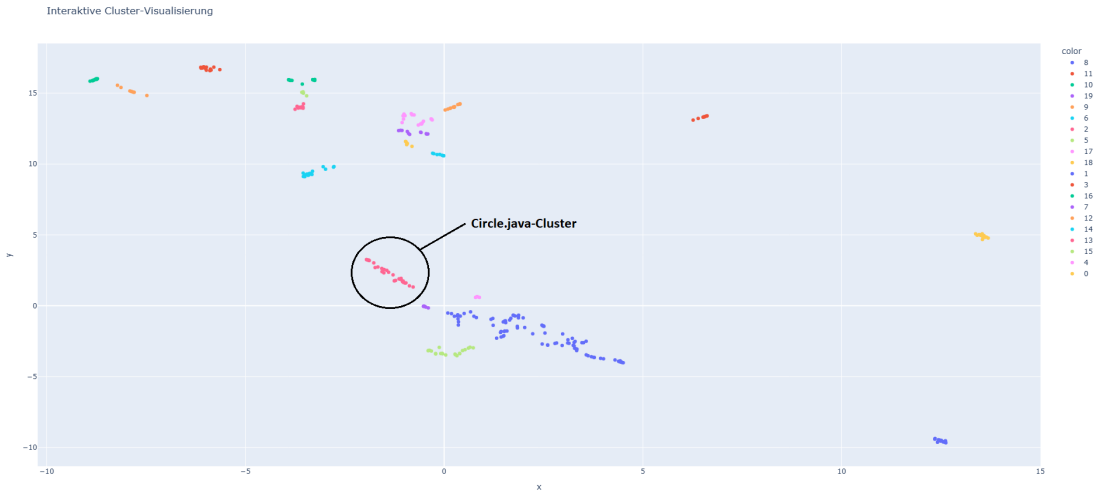

Abbildung 4.1: Clustering-Diagramm einer Clusterung von 320 Java-Dateien. Der eingekreiste Cluster ist ein Circle.java-Cluster.

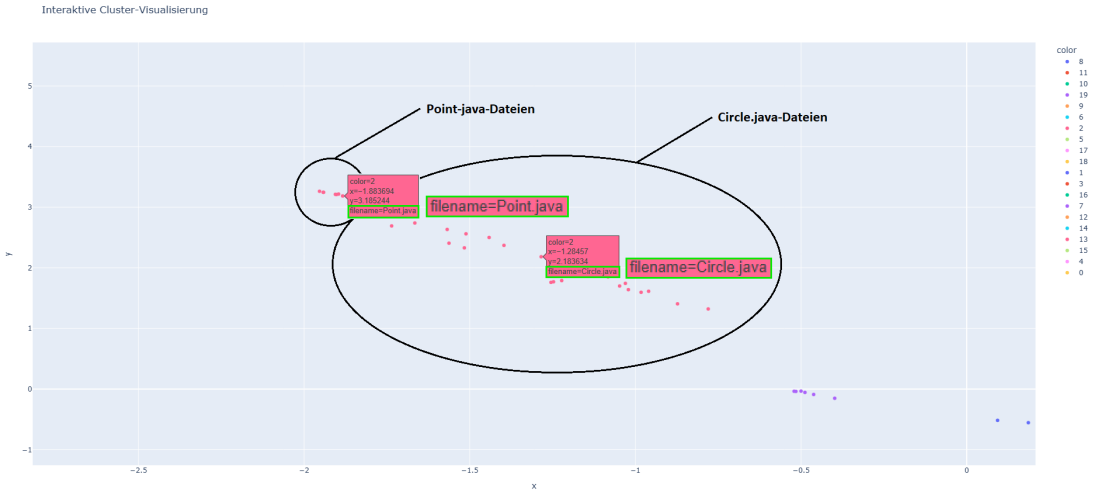

Abbildung 4.2: Vergrößerter Circle.java-Cluster aus Abbildung 4.1

5

Zusammenfassung und Ausblick

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die z.B. Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

6

Anhang

Literaturverzeichnis

Cal74. Calinski, T. and Harabasz, J.: *A dendrite method for cluster analysis.* Communications in Statistics, 3(1):1–27, 1974.

CMS. Campello, Ricardo J. G. B., Davoud Moulavi und Joerg Sander: *Density-Based Clustering Based on Hierarchical Density Estimates.* Seiten 160–172.

DB79. Davies, D. L. und D. W. Bouldin: *A Cluster Separation Measure.* IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2):224–227, 1979.

FGT+. Feng, Zhangyin, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang und Ming Zhou: *CodeBERT: A Pre-Trained Model for Programming and Natural Languages.*

HBV01. Halkidi, Maria, Yannis Batistakis und Michalis Vazirgiannis: *On Clustering Validation Techniques.* Journal of Intelligent Information Systems, 17(2-3):107–145, 2001.

JS21. Jerrim, John und Sam Sims: *When is high workload bad for teacher wellbeing? Accounting for the non-linear contribution of specific teaching tasks.* Teaching and Teacher Education, 105:103395, 2021.

Kar01. Karl Pearson: *On lines and planes of closest fit to systems of points in space.* Philosophical Magazine, 2(11):559–572, 1901.

Lau08. Laurens van der Maaten und Geoffrey Hinton: *Visualizing Data using t-SNE.* Journal of Machine Learning Research, 9:2579–2605, 2008.

Mac67. MacQueen, J.: *Some methods for classification and analysis of multivariate observations.* In: Le Cam, Lucien M. und Jerzy Neyman (Herausgeber): *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Band 5.1, Seiten 281–298. University of California Press, 1967.

MBKS. Messer, Marcus, Neil C. C. Brown, Michael Kölling und Miaojing Shi: *Automated Grading and Feedback Tools for Programming Education: A Systematic Review.*

MHM. McInnes, Leland, John Healy und James Melville: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.*

OJM. Orvalho, Pedro, Mikoláš Janota und Vasco Manquinho: *InvAASTCluster: On Applying Invariant-Based Program Clustering to Introductory Programming Assignments*.

PLF24. Paiva, José Carlos, José Paulo Leal und Álvaro Figueira: *Clustering source code from automated assessment of programming assignments*. International Journal of Data Science and Analytics, Seiten 1–12, 2024.

Rou87. Rousseeuw, Peter J.: *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*. Journal of Computational and Applied Mathematics, 20:53–65, 1987.

TWH+. Tang, Xiaohang, Sam Wong, Marcus Huynh, Zicheng He, Yalong Yang und Yan Chen: *SPHERE: Scaling Personalized Feedback in Programming Classrooms with Structured Review of LLM Outputs*.

You24. Youssef Lahmadi, Mohammed Zakariae El Khattabi, Mounia Rahhali, Lahcen Oughdir: *Optimizing Adaptive Learning: Insights from K-Means Clustering in Intelligent Tutoring Systems*. International Journal of Intelligent Systems and Applications in Engineering, 12(3):1842–1851, 2024.

A

Glossar

DisASTer	Distributed Algorithms Simulation Terrain, eine Plattform zur Implementierung verteilter Algorithmen [**?**]
DSM	Distributed Shared Memory
AC	Atomic Consistency (dt.: Linearisierbarkeit)
RC	Release Consistency (dt.: Freigabekonsistenz)
SC	Sequential Consistency (dt.: Sequentielle Konsistenz)
WC	Weak Consistency (dt.: Schwache Konsistenz)

B

Eigenständigkeitserklärung

☐ Die vorliegende Arbeit wurde als Einzelarbeit angefertigt.

☐ Die vorliegende Arbeit wurde als Gruppenarbeit angefertigt. Mein Anteil an der Gruppenarbeit ist im untenstehenden Abschnitt *Verantwortliche* dokumentiert:

☐ Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige Hilfe Dritter angefertigt habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche kenntlich gemacht. Darüber hinaus erkläre ich, dass ich die vorliegende Arbeit in dieser oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht habe.

☐ Es ist keine Nutzung von KI-basierten text- oder inhaltgenerierenden Hilfsmitteln erfolgt.

☐ Die Nutzung von KI-basierten text- oder inhaltgenerierenden Hilfsmitteln wurde von der/dem Prüfenden ausdrücklich gestattet. Die von der/dem Prüfenden mit Ausgabe der Arbeit vorgegebenen Anforderungen zur Dokumentation und Kennzeichnung habe ich erhalten und eingehalten. Sofern gefordert, habe ich in der untenstehenden Tabelle *Nutzung von KI-Tools* die verwendeten KI-basierten text- oder inhaltgenerierenden Hilfsmittel aufgeführt und die Stellen in der Arbeit genannt. Die Richtigkeit übernommener KI-Aussagen und Inhalte habe ich nach bestem Wissen und Gewissen überprüft.

Datum

Unterschrift der Kandidatin/des Kandidaten

Verantwortliche

Die Tabellen unten führen auf, wer als Autor für die einzelnen Kapitel der vorliegenden Dokumentation beziehungsweise für einzelne Teile des Quellcodes hauptverantwortlich ist.

Insgesamt beteiligt sind die folgenden Personen:

- Autor 1
- Autor 2
- Autor 3

Dokumentation

Kapitel	**Überschrift**	**Autor**
1	Einleitung	Autor 1, Autor 2, Autor 3
2	Problemstellung	Autor 1, Autor 2, Autor 3
3	Aufgabenstellung und Zielsetzung	Autor 1, Autor 2, Autor 3
4	Übrige Abschnitte (Kapitel und Absätze)	Autor 1
4.1	Abschnitt	Autor 3
4.1.1	Unterabschnitt	Autor 2, Autor 3
4.2	Abbildungen und Tabellen	usw.
4.3	Mathematische Formel	
4.4	Sätze, Lemmas und Definitionen	
4.5	Fußnoten	
4.6	Literaturverweise	
5	Beispiel-Kapitel	
5.1	Warum existieren unterschiedliche Konsistenzmodelle?	
5.2	Klassifizierung eines Konsistenzmodells	
5.3	Linearisierbarkeit (atomic consistency)	

Quellcode

Paket	**Autor**
algorithms.search	Autor 1
algorithms.sort	Autor 3

Nutzung von KI-Tools

KI-Tool	**Genutzt für**	**Warum?**	**Wann?**	**Mit welcher Eingabefrage bzw. -aufforderung?**	**An welcher Stelle der Arbeit übernommen?**
ChatGPT	Konzept XY erklären lassen	Erklärung von Verständnisfragen zu...	Bei der Bearbeitung des Theorieteils der Arbeit	Welches sind die zentralen Merkmale des Konzepts XY?	S. 25, 30 ff.
DeepL Write	Neuformulierung meiner Textentwürfe	Bessere Lesbarkeit	Über die gesamte Arbeit hinweg	Formuliere die Kapitel 2 und 3 neu in einfachen und leicht verständlichen Sätzen!	S. 45 ff. , S. 67