Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Ansätze zur Gruppierung von Anfragen über Ereignisströmen

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)

eingereicht von: Morris Baude
geboren am: 15.09.2003
geboren in: Berlin

Gutachter/innen: Prof. Dr. Matthias Weidlich
Dr. rer. nat. Patrick Schäfer

eingereicht am: ..... verteidigt am: .....

Abstract. Complex Event Processing beschäftigt sich mit der Erkennung relevanter Situationen in kontinuierlichen Ereignisströmen. Um festzustellen, ob bestimmte Situationen eintreten, werden fortlaufend Anfragen über diese Ströme ausgeführt. Zur Optimierung werden ähnliche Anfragen gemeinsam verarbeitet oder die Reihenfolge der Ereignisverarbeitung angepasst. Das Ziel dieser Arbeit unterscheidet sich jedoch, da hier die Anfragen bereits vor ihrer Ausführung optimiert werden sollen. Zu diesem Zweck werden Methoden untersucht, um Anfragen anhand ihrer Ähnlichkeit zu clustern und in Gruppen zu ordnen. Anstatt jede Anfrage einzeln auszuführen, kann so eine repräsentative Anfrage pro Cluster getestet, was den Rechenaufwand verringert würde. Darüber hinaus bietet das Clustering Rückschlüsse auf die Relevanz der Anfragen. Anfragen, die isoliert in einer Gruppe platziert werden und sich stark von den anderen unterscheiden, könnten weniger relevant sein, da sie nur selten geprüft werden. Im Gegensatz dazu deuten größere Cluster auf häufig vorkommende und zentrale Ereignisstrukturen hin, die möglicherweise eine Schlüsselrolle spielen. Zur Untersuchung der Clusterung wurden drei Ansätze zu der Berechnung der Ähnlichkeit geprüft: die Ähnlichkeit von Anfragen in einer hierarchischen Struktur, die syntaktische Ähnlichkeit und die semantische Ähnlichkeit von Anfragen. Der hierarchische Ansatz zeigte nur begrenzte Einsatzmöglichkeiten und ist höchstens in Sonderfällen nützlich. Der syntaktische Ansatz erwies sich als vielversprechend, da er konsistente Cluster bildete und Anfragen basierend auf ihrer strukturellen Ähnlichkeit sinnvoll gruppierte. Der semantische Ansatz blieb hinter den Erwartungen zurück, könnte jedoch in Zukunft großes Potenzial entfalten.

Inhaltsverzeichnis

1 Einleitung

Ereignisse, oder auch Events im Englischen, spielen eine zentrale Rolle in modernen Computern und Informationssystemen, da sie zur Steuerung des Verhaltens und der Reaktionen von Systemen genutzt werden. In den letzten Jahren ist die Zahl der generierten Ereignisse kontinuierlich gestiegen, was auf mehrere Faktoren zurückzuführen ist. Ein wesentlicher Grund dafür ist der Wandel von nutzeranfragegesteuerten Interaktionen im Web hin zu dynamischen, ereignisgesteuerten Interaktionen. Darüber hinaus wächst die Nachfrage nach der Automatisierung von Geschäftsprozessen sowie nach ereignisgesteuerten Architekturen. Auch rechtliche und betriebliche Anforderungen, die eine Echtzeitüberwachung von IT-Systemen vorschreiben, tragen zu diesem Anstieg bei. Vor diesem Hintergrund beschäftigt sich der Bereich des Complex Event Processing (CEP) mit der Analyse und Verarbeitung dieser Ereignisse, um aus der Vielzahl an Daten hochwertiges Wissen und bedeutungsvolle Erkenntnisse abzuleiten und das möglichst in Echtzeit [9]. CEP wird in vielen Bereichen eingesetzt, um auf Verhaltensmuster in proaktiven Anwendungen zu reagieren. Ziel von CEP ist es, bedeutsame Situationen zu identifizieren, die Fehler, sicherheitsrelevante Vorfälle oder informative Ereignisse sein können. So wird CEP beispielsweise im Bereich des Energiemanagements genutzt, um Verbrauchsmuster zu analysieren und Maßnahmen zur Reduktion des Energieverbrauchs zu ergreifen [37]. Im Finanzsektor wird CEP eingesetzt, um durch die Analyse von Transaktionsdaten Betrugsfälle frühzeitig zu erkennen [33], während im Bereich der Verkehrsüberwachung Verkehrsdaten verarbeitet werden, um in Echtzeit Vorhersagen über den Verkehrsfluss zu ermöglichen [35]. Ereignisströme werden dabei kontinuierlich mit einer Vielzahl von Anfragen abgeglichen, um festzustellen, ob eine solche Situation bereits eingetreten ist oder sich anbahnen könnte.

Für die Identifikation geeigneter Anfragen beschreibt [29] ein Verfahren, das auf einem Datensatz historischer Ereignisströme basiert. Dieser enthält Beispiele relevanter Situationen, aus denen automatisch Anfragen generiert werden, um vergleichbare Muster künftig in Echtzeit zu erkennen. Die zentrale Idee besteht darin, dass eine Anfrage, die typische Gemeinsamkeiten mehrerer vergangener Ereignisströme identifiziert, auch künftig zuverlässig ähnliche Situationen erkennen kann. Diese Anfragen beschreiben Abfolgen von Ereignissen, bei denen einzelne Ereignisse entweder durch konkrete Typen festgelegt sind oder Platzhalter enthalten, die keine festen Vorgaben machen. Ergänzend wird jeder Anfrage ein Zeitfenster zugewiesen, das bestimmt, innerhalb wie vieler Ereignisse die in der Anfrage beschriebenen Muster im Datenstrom auftreten müssen, um als Hinweis auf eine bedeutsame Situation zu gelten. Da jedoch unklar ist, welche Anfragen tatsächlich die relevanten Muster erfassen, erzeugt der Algorithmus eine Vielzahl unterschiedlicher Varianten. Jede dieser Anfragen enthält unterschiedliche Gemeinsamkeiten der Ereignisströme. Je nachdem, wie lange der Algorithmus läuft und wie der Datensatz aussieht, können Hunderte oder sogar Tausende von Anfragen generiert werden. Diese Anfragen sind einzigartig und keine Anfrage ist eine Teilmenge einer anderen, weshalb sie alle disjunkt sind. Trotzdem stellt sich die Frage, ob es Anfragen gibt, die mehr miteinander zu tun haben als andere. Eine Möglichkeit, dies zu untersuchen, besteht darin, die Anfragen in Gruppen zu unterteilen, sodass Anfragen innerhalb einer Gruppe als ähnlich betrachtet

werden, während Anfragen aus verschiedenen Gruppen als unterschiedlich gelten.
Dies führt zu der zentralen Problematik dieser Arbeit. Um die Anfragen trotz ihrer Disjunktheit sinnvoll zu gruppieren, ist es notwendig, sie in Vektoren umzuwandeln. Auf diese Weise können ähnliche Anfragen im Vektorraum nahe beieinander liegen, während unterschiedliche Anfragen weiter voneinander entfernt sind. Ziel dieser Arbeit ist es, eine geeignete Methode zur Vektorisierung und anschließenden Clusterung von Anfragen zu entwickeln. Hierzu werden verschiedene Ansätze vorgestellt und hinsichtlich ihrer Eignung bewertet, um am Ende eine effektive Vektorisierungsmethode für Anfragen über Ereignisströme zu identifizieren, die sich zur Clusterbildung eignet. Dafür ist die Arbeit wie folgt gegliedert. Zunächst werden die Grundlagen für die Notation und Erstellung von Anfragen über Ereignisströme erläutert, um ein grundlegendes Verständnis für die Thematik zu vermitteln. In Abschnitt 3 wird der aktuelle Stand der Forschung dargestellt. In Abschnitt 4 werden die theoretischen Grundlagen für die Ansätze und die Clusterung behandelt, die später im praktischen Teil von Bedeutung sind. Hier werden für die drei einzelnen Ansätze verschiedene Möglichkeiten zur Vektorisierung vorgestellt. Zudem wird in diesem Kapitel die Theorie hinter verschiedenen Clusterverfahren erläutert und ein Clusteralgorithmus detaillierter betrachtet. In Kapitel 5 wird der praktische Teil der Arbeit behandelt. Zunächst wird das Setup und der verwendete Datensatz vorgestellt, gefolgt von einer detaillierten Betrachtung der Implementierung. Im Anschluss werden die Parameter definiert, die für die einzelnen Algorithmen genutzt werden. Danach werden die Ergebnisse der verschiedenen Clusterungen präsentiert, die in der Auswertung zusammengefasst und anschließend mit einem Ausblick auf zukünftige Entwicklungen abgeschlossen werden. Die Arbeit endet mit einem Fazit.

2 Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen für die verwendete Notation sowie für das genutzte Modell zur Beschreibung von Anfragen vermittelt. Grundkenntnisse mathematischer Konzepte und Notationen sind dabei für das Verständnis hilfreich. Zunächst werden Ereignisse und Ereignisketten definiert, anschließend folgt die Beschreibung und Erstellung von Anfragen über diese Strukturen. Für die Erstellung dieses Abschnitts wurde das Paper [16] herangezogen, das die Grundlagen für eindimensionale Anfragen über Ereignisketten beschreibt. Die Erweiterung auf multidimensionale Ereignisströme wird in [17] erläutert. In [29] wird schließlich der Algorithmus zur Entdeckung solcher Anfragen vorgestellt.

Teilfolgen Für ein Alphabet Γ wird die Menge aller Zeichenketten durch Γ^* beschrieben. Die Länge einer Zeichenkette s wird mit $|s|$ angegeben. Für eine Position $i \in [|s|]$ bezeichnet $s[i]$ das Zeichen an der Stelle i in s. Ist eine Zeichenkette $t = t_1 t_2 \cdots t_n$ mit $t_i \in \Gamma$ gegeben, so heißt eine weitere Zeichenkette $s = s_1 \cdots s_m$ eine Teilfolge von t, wenn $m \leq n$ gilt und es Indizes $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ gibt, sodass $s_j = t_{i_j}$ für alle $j \in [m]$. Anschaulich bedeutet dies, dass die Zeichen von s in derselben Reihenfolge in t vorkommen. Die Zuordnung der Zeichen erfolgt über eine Funktion $e : [m] \to [n]$ mit

$e(j) = i_j$ für alle $j \in [m]$.

Ereignisketten Sei Γ ein Alphabet mit mindestens zwei Symbolen, also $|\Gamma| \geq 2$. Die Elemente von Γ werden als Typen bezeichnet. Zudem sei $k \in N_{\geq 1}$ eine natürliche Zahl. Ein Ereignis e ist ein geordnetes k-Tupel über dem Alphabet Γ, wobei k die Anzahl der Attribute eines Ereignisses angibt. Daher wird e auch als k-dimensionales Ereignis bezeichnet. Zur besseren Lesbarkeit werden Ereignisse in runden Klammern geschrieben und als Einheit betrachtet. Eine k-dimensionale Ereigniskette ist eine endliche, nichtleere Sequenz solcher Ereignisse, wobei jedes Ereignis dieselbe Anzahl an Attributen, nämlich k, aufweist. Beispiel 1 veranschaulicht die zuvor eingeführten Begriffe der Ereigniskette sowie der Teilfolge zwischen Ereignisketten.

Beispiel 1:
Es werden dreidimensionale Ereignisse betrachtet, bei denen die Attribute unterschiedliche Bedeutungen haben können, beispielsweise könnten sie für JobID, Status und Priorität stehen. Für dieses Beispiel ist die genaue Bedeutung jedoch nicht relevant. Wichtig ist lediglich, dass es sich bei den Ereignissen um geordnete 3-Tupel handelt, bei denen die Reihenfolge der Elemente von Bedeutung ist. Die Werte der Tupel stammen aus einer Menge $\Gamma := \{high, low\} \cup \{0, 1, \ldots, 9\}$. *Ein konkretes Ereignis könnte somit beispielsweise* $(1, 0, low)$ *lauten. Eine Ereigniskette ist eine Sequenz solcher Ereignisse. Beispielsweise kann eine Ereigniskette* s *aus zwei Ereignissen bestehen:* $(1, 0, low)$ *und* $(2, 3, high)$. *Eine andere Ereigniskette* t *könnte drei Ereignisse enthalten:* $(1, 0, low)$, $(1, 4, low)$ *und* $(2, 3, high)$. *Ein zentrales Element bei der Entdeckung von Anfragen über verschiedene Ereignisketten ist das Erkennen von Teilfolgen. In diesem Fall beschreibt die Abbildung* $e : [2] \to [3]$ *mit* $e(1) = 1$, $e(2) = 3$, *dass die beiden Ereignisse von* s *in den Positionen 1 und 3 von* t *wiederzufinden sind. Somit ist* s *als Teilfolge in* t *enthalten.*

Anfragen über Ereignisketten Die Grundlagen für das Modell der Anfragen mit Platzhaltern und Abstandsgrenzen wurden 2022 in dem Paper [16] gelegt. In dieser Arbeit konzentrierten sich die Autoren auf sogenannte eindimensionale SWG-Anfragen und entwickelten ein neues Anfragemodell, um Ereignisströme präziser zu beschreiben. Zur Definition dieses Modells wird zunächst eine neue Menge möglicher Variablen VAR eingeführt, die disjunkt zur Menge Γ ist. Eine SWG-Anfrage q besteht aus drei Hauptelementen $q = (s, w, c)$:

- Anfragezeichenkette s: Diese setzt sich aus Variablen aus VAR und Typen aus Γ zusammen und ist formal definiert als $s \in (\text{VAR} \cup \Gamma)^+$.
- Globale Fenstergröße w: Sie gibt an, wie groß der maximale Abstand zwischen dem ersten und letzten Ereignis innerhalb einer Ereigniskette sein darf, damit die Anfrage als Treffer gilt. Formal: $w \in \{1, 2, 3, \ldots\} \cup \{\infty\}$.
- Lokale Abstandsgrenzen c: Diese Menge beschreibt die erlaubten Abstände zwischen benachbarten Ereignissen innerhalb der Anfrage. Sie wird als $c = \{c_1, c_2, \ldots, c_{|s|-1}\}$

definiert, wobei jedes c_i ein Tupel $(c_i^-, c_i^+) \in \{0, 1, 2, \dots\} \times (\{0, 1, 2, \dots\} \cup \{\infty\})$ ist. Dabei steht c_i^- für den minimalen und c_i^+ für den maximalen Abstand zweier aufeinanderfolgender Ereignisse. Es gilt stets $c_i^- \leq c_i^+$.

Falls für einen Abstand keine Einschränkungen bestehen, wird dieser mit $c_i = (0, \infty)$ angegeben. Zusätzlich muss die Summe der minimalen Abstände gemeinsam mit der Länge der Anfragezeichenkette kleiner oder gleich der globalen Fenstergröße w sein. Während eindimensionale Anfragen ausschließlich auf Ereignisketten mit einer Dimension bezogen sind, spricht man bei Anfragen mit einer Dimension $k > 1$ von multidimensionalen Teilfolgenanfragen (MSWG-Anfragen). In diesem Fall stimmt die Dimension der Ereignisse in der Anfrage mit der Dimension der Ereignisse in den zugrunde liegenden Ereignisketten überein.

Matching In einer MSWG-Anfrage fungieren Variablen als Platzhalter und stehen für beliebige Typen aus dem Alphabet Γ. Sie werden mithilfe eines '$' in dem Modell kenntlich gemacht. Eine MSWG-Anfrage q wird als Treffer für eine k-dimensionale Ereigniskette t gewertet, wenn identische Variablen in der Anfrage durch gleiche konkrete Typen aus Γ ersetzt werden können. Zusätzlich müssen auch die Platzhalter durch konkrete Typen aus Γ ersetzt werden können, sodass die daraus resultierende k-dimensionale Ereigniskette q' eine Teilfolge von t bildet und die folgenden Bedingungen erfüllt:

- Lokale Gap-Size Constraints: Jede Lücke c_i in q' muss durch eine Sequenz von Ereignissen in t ersetzt werden, deren Anzahl mindestens c_i^- und höchstens c_i^+ beträgt.
- Globale Fenstergröße: Es existiert ein Teilstring t' von t, dessen Länge höchstens der globalen Fenstergröße w entspricht und der q' als Teilfolge enthält.

In Beispiel 2 wird ein passendes Beispiel für eine Anfrage näher erläutert.

Beispiel 2: *Angenommen, es liegen 3-dimensionale Ereignisse vor, deren Attribute die Bedeutung JobID, Status und Priorität haben. Dabei stammen die Typen aus der Menge* $\Gamma := \{high, low\} \cup \{0, 1, \dots, 9\}$. *Eine Anfrage der Form* $(\$x0, \_, low)$ $(\$x0, \_, low)$ *mit einer Fenstergröße von drei würde mit jeder Ereignisketten übereinstimmen, die innerhalb von drei Ereignissen zweimal dieselbe JobID und die Priorität 'low' enthalten. Formal entspricht dies einer 3-SWG-Anfrage* $q = (((\$x0, \_, low)\,(\$x0, \_, low)), 3, c)$. *Da keine speziellen Abstandsgrenzen zwischen den Ereignissen vorgegeben sind, wird* c *als* $c = ((0, \infty))$ *definiert. Aus Beispiel 1 würde die Ereigniskette* s *nicht mit der Anfrage übereinstimmen, da weder die Platzhalter noch die beiden Variablen der Anfrage so mit konkreten Typen ergänzt werden können, dass die Anfrage eine Teilfolge von* s *bildet. Bei der Ereigniskette* t *hingegen lässt sich für die Variable* $\$x0$ *der Wert 1 einsetzen, und die übrigen Typen können entsprechend den ersten beiden Ereignissen von* t *ergänzt werden. Dadurch ergibt sich eine gültige Abbildung der Anfrage* q *mit* $e(1) = 1$ *und* $e(2) = 2$, *sodass* q *eine Teilfolge von* t *ist. Da sich zudem beide passenden Ereignisse innerhalb eines Fensters von drei Ereignissen befinden, erfüllt* t *sowohl die globale Fenstergröße als*

auch die lokalen Abstandsgrenzen, wodurch q mit t matched. Eine weitere Ereigniskette, die mit der Anfrage übereinstimmen würde, wäre zum Beispiel ($(5, 1, low)$ $(2, 3, high)$ $(5, 2, low)$*). Trotz der unterschiedlichen Typen in der Ereignissekette werden hier ebenfalls alle Bedingungen erfüllt, sodass q auch mit dieser Ereigniskette übereinstimmen würde.*

In dieser Arbeit werden weder die lokalen Abstandsgrenzen noch die globale Fenstergröße der Anfragen weiter betrachtet. Für die Clusterung ist lediglich die Struktur der Zeichenketten der Anfragen von Bedeutung.

Beschreibende Anfragen Im Folgenden wird die Teilmenge von Anfragen betrachtet, die in dieser Arbeit im Mittelpunkt steht und letztlich geclustert werden soll. Dabei handelt es sich um sogenannte beschreibende Anfragen. Ähnlich wie reguläre Anfragen beziehen sie sich auf eine bestimmte Menge X von Ereignisketten innerhalb eines Datensatzes. Der wesentliche Unterschied besteht jedoch darin, dass keine andere, spezifischere Anfrage existiert, die dieselbe Menge X beschreibt. Eine Spezifizierung einer Anfrage liegt dann vor, wenn ein zusätzliches Ereignis eingefügt oder ein bestehender Platzhalter in einem Ereignis durch eine konkrete Variable oder ein konkretes Attribut ersetzt wird. Anders ausgedrückt beschreiben alle Anfragen, die spezifischer als eine beschreibende Anfrage sind, nicht dieselbe Menge an Ereignisketten und werden daher nicht als repräsentative Anfragen in die Menge der beschreibenden Anfragen aufgenommen.

Algorithmus zur Anfrageerkennung In dem Paper [29] wurde ein neuer Algorithmus vorgestellt, um Anfragen über Ereignisströme zu identifizieren. Der Algorithmus nutzt einen Datensatz von Ereignisketten, einen Schwellenwert, der angibt, wie viele Ereignisketten eine Anfrage mindestens matchen muss, eine Menge häufiger Symbole, die in genügend Ereignissen vorkommen, um den Schwellenwert abzudecken, zwei Mengen zur Speicherung der gefundenen Anfragen sowie der Beziehungen zu ihren jeweiligen Elternanfragen und einen Anfragestack, der jene Anfragen enthält, von denen ausgehend der Algorithmus nach spezifischeren Anfragen sucht. Solange der Stack zur Suche nach spezifischeren Anfragen nicht leer ist, wird eine Anfrage vom Stack genommen. Anschließend wird geprüft, ob diese Anfrage im Datensatz mehr Ereignisketten matched als der festgelegte Schwellenwert vorgibt. Ist dies der Fall, wird die Anfrage als gültige Anfrage in der Anfragemenge gespeichert. Danach werden zu dieser Anfrage die sogenannten „Kinder“ ermittelt. Damit sind spezifischere Anfragen gemeint, die durch gezielte Verfeinerung der ursprünglichen Anfrage entstehen. Diese Kinderanfragen werden zusammen mit der Ausgangsanfrage in der Menge der Eltern-Kind-Beziehungen gespeichert. Abschließend werden alle Kinderanfragen dem Stack hinzugefügt, sodass die Suche von dort aus weiter fortgesetzt werden kann. So werden für die im Stapel übergebenen Anfragen alle möglichen Spezifizierungen ermittelt, solange die jeweiligen Anfragen weiterhin den Schwellenwert erfüllen. Der Algorithmus erzeugt dabei eine hierarchische Struktur, in der jede Anfrage auf eine oder mehrere übergeordnete Anfragen zurückgeführt werden kann, beginnend bei den ursprünglichen Anfragen, die zusammen mit dem Stapel an die Funktion übergeben wurden. In dieser Hierarchie sind jene Anfragen, die selbst nicht als Eltern für andere dienen, die spezifischsten. Für diese Anfragen konnten keine weiteren Spezifizierungen

gefunden werden, die ebenfalls den festgelegten Schwellenwert im Datensatz erreichen. Diese Anfragen stellen die beschreibenden Anfragen dar, die im weiteren Verlauf dieser Arbeit analysiert und geclustert werden sollen.

3 Literaturübersicht

In diesem Kapitel wird ein Überblick über den aktuellen Stand der Forschung gegeben. Zunächst wird dargestellt, welche Ansätze bislang zur Optimierung der Erkennung bedeutsamer Ereignisse entwickelt wurden. Anschließend folgen Methoden zur Berechnung von Ähnlichkeit in anderen Anwendungsbereichen sowie Verfahren zur Bildung von Clustern.

Berechnungsoptimierung CEP beschäftigt sich mit der Erkennung von bedeutsamen Ereignissen in Echtzeit. Dazu werden kontinuierlich eintreffende Ereignisströme mit Anfragen überprüft, die solche bedeutsamen Ereignisse identifizieren sollen. Da dabei eine große Menge an Daten verarbeitet wird, ist die Optimierung dieser Verarbeitung ein wichtiges Thema [9]. Ein Ansatz dafür ist die sogenannte Multi-Query-Optimierung, wie sie zum Beispiel bei SAP ESP verwendet wird. Dabei wird versucht, mehrere Anfragen gemeinsam zu verarbeiten, um Rechenleistung zu sparen. Es wird geprüft, ob mehrere Anfragen ähnliche Ereignisse überwachen, sodass die Berechnungsergebnisse gemeinsam genutzt werden können und nicht mehrfach berechnet werden müssen [36]. Eine weitere Möglichkeit ist die Join-Optimierung, bei der die Reihenfolge, in der die Ereignisse betrachtet werden, angepasst wird, um die Berechnung der Ereignismenge effizienter zu gestalten. So lohnt es sich zum Beispiel, mit seltenen Ereignissen zu beginnen, da dadurch die Ergebnismenge früh eingeschränkt wird, bevor häufige Ereignisse geprüft werden [18]. Es existieren weitere Ansätze, um die Verarbeitung von Anfragen effizienter zu gestalten. Im Unterschied zu diesen Methoden setzt die vorliegende Arbeit jedoch bereits vor der eigentlichen Ausführung an. Ziel ist es, Ähnlichkeiten zwischen Anfragen zu erkennen und diese zu gruppieren. Anfragen innerhalb einer Gruppe könnten beispielsweise dieselbe Situation prüfen, obwohl sie unterschiedliche Attributwerte verwenden. In solchen Fällen wäre es unter Umständen nicht notwendig, alle Anfragen einzeln auszuführen. Vielmehr könnte es ausreichen, stellvertretend nur einen Teil davon zu verarbeiten. Dafür werden in dieser Arbeit Ansätze zur Clusterung von Anfragen vorgestellt, basierend auf ihrer Position in einer hierarchischen Struktur, ihrer Syntax und ihrer Semantik, um mögliche Ähnlichkeiten zu erkennen.

Strukturelle Ähnlichkeit In einem Graphen existieren zahlreiche Ansätze zur Bestimmung von Ähnlichkeiten zwischen Knoten. Eine Möglichkeit basiert auf Random Walks. Dabei bewegt man sich zufällig durch den Graphen. Knoten, die viele Pfade miteinander teilen, werden in solchen Random Walks häufig gemeinsam besucht, was auf eine hohe strukturelle Ähnlichkeit hinweist [11]. Ein alternativer Ansatz vergleicht Knoten anhand ihrer lokalen Struktur. Zwei Knoten gelten als strukturell ähnlich, wenn sie beispielsweise dieselbe Anzahl an Nachbarn besitzen oder ihre jeweiligen Nachbarschaften ähnliche Muster aufweisen. In Baumstrukturen gelten Knoten als ähnlich, wenn sie identische

Teilbäume aufweisen oder in vergleichbaren Substrukturen eingebettet sind [12]. Da es sich bei der hier betrachteten Struktur um eine hierarchische Struktur mit wenigen Pfaden zwischen den Knoten handelt, ist der Einsatz von Random Walks wenig zielführend. Auch die Analyse lokaler Strukturen liefert keine zusätzlichen Informationen, da in dieser Arbeit ausschließlich Knoten betrachtet werden, die Blätter des Baumes sind und somit dieselbe Unterstruktur aufweisen. Ein besonders vielversprechender Ansatz zur Bestimmung der Knotenähnlichkeit beruht auf distanzbasierten Maßen. In allgemeinen Graphen kann dies etwa über die Berechnung des kürzesten Pfads zwischen zwei Knoten erfolgen. In Baumstrukturen hingegen wird häufig der kleinste gemeinsame Vorfahre (LCA) verwendet. Befindet sich dieser tief im Baum, deutet das auf einen spezifischen gemeinsamen Kontext hin. Ein kombinierter Ansatz, der sowohl den kürzesten Pfad als auch den LCA berücksichtigt, wurde von Wu und Palmer [34] vorgeschlagen. Ihr Ziel war es, die semantische Ähnlichkeit zwischen Verben zu messen, basierend auf deren Einordnung in einer hierarchischen Struktur, in der Verben semantischen Klassen zugeordnet sind.

Syntaktische Ähnlichkeit Es gibt mehrere Möglichkeiten, die Ähnlichkeit zwischen Zeichenketten basierend auf ihrer Syntax zu bestimmen. Ein bekannter Ansatz ist die Editierdistanz nach Levenshtein [20]. Dabei wird die minimale Anzahl an Einfüge-, Lösch- und Ersetzoperationen gezählt, die notwendig sind, um eine Zeichenkette in eine andere zu überführen. Eine weitere Möglichkeit zur Bestimmung der Ähnlichkeit ist die Jaccard-Ähnlichkeit. Dabei werden die Zeichenketten in Mengen zerlegt, wobei die Elemente beispielsweise n-Gramme, Wörter oder andere Einheiten sein können. Jede Zeichenkette wird als Menge von solchen Elementen dargestellt. Die Ähnlichkeit ergibt sich aus dem Verhältnis der Anzahl gemeinsamer Elemente zur Gesamtanzahl aller Elemente beider Zeichenketten [7]. Schließlich lassen sich Zeichenketten auch durch Vektorrepräsentationen vergleichen, wie es im Information Retrieval üblich ist. Dabei wird jede Zeichenkette in einen Vektor überführt, in dem jede Dimension ein n-Gramm oder ein Wort repräsentiert. Als Wert pro Dimension kann etwa die bloße Existenz, die Häufigkeit des Auftretens oder ein normiertes Maß wie tf-idf verwendet werden. Die Ähnlichkeit lässt sich anschließend beispielsweise über die Cosinus-Ähnlichkeit oder die euklidische Distanz berechnen [22].

Semantische Ähnlichkeit Um die semantische Ähnlichkeit zwischen Zeichenketten zu bestimmen, existieren verschiedene Ansätze. Früher wurden häufig Ressourcen wie WordNet verwendet, in denen Wörter als Knoten in einem Netzwerk organisiert und über Kanten verbunden sind, die semantische Beziehungen wie Synonyme oder Oberbegriffe abbilden [24]. Ein wesentlicher Nachteil besteht darin, dass WordNet nur bekannte Wörter enthält und manuell gepflegt wird. Dadurch ist es ungeeignet für neue, fachspezifische Begriffe oder eigene Grammatiken, wie sie bei den Anfragen in diesem Kontext auftreten. Moderne Methoden nutzen maschinelles Lernen, um die Semantik von Wörtern automatisch zu erfassen. Dabei kommen Word Embeddings zum Einsatz, bei denen Wörter als Vektoren in einem mehrdimensionalen Raum dargestellt werden. Modelle wie Word2Vec [23] oder GloVe [26] lernen diese Repräsentationen aus umfangreichen Textkorpora. Wörter mit ähnlicher Bedeutung erhalten ähnliche Vektoren und liegen in diesem Raum entsprechend nahe beieinander.

Clusterung Nach der Vektorisierung der Anfragen folgt im nächsten Schritt die eigentliche Clusterung. Dabei existieren verschiedene Ansätze, um Punktmengen in Gruppen zu unterteilen. Ein bekannter Ansatz ist das dichtebasierte Clustering, wie DBScan. Hierbei wird für jeden Punkt überprüft, wie viele Nachbarpunkte sich innerhalb eines bestimmten Radius befinden. Übersteigt diese Anzahl einen vordefinierten Schwellenwert werden die Punkte in der Region als ein Clusters betrachtet. Ein typisches Merkmal dieses Verfahrens ist, dass nicht alle Punkte einem Cluster zugeordnet werden. Punkte, die nicht genügend Nachbarn aufweisen, gelten als Ausreißer und bleiben ungruppiert [10]. Da in dieser Arbeit jedoch alle Anfragen einem Cluster zugeordnet werden sollen, wird dieser Ansatz nicht weiter betrachtet. Stattdessen liegt der Fokus auf distanzbasierten Verfahren, bei denen alle Punkte zwingend in Cluster eingeteilt werden. Eine Gruppe solcher Verfahren ist das hierarchische Clustering, bei dem eine Clusterhierarchie aufgebaut wird. Entweder durch schrittweises Zusammenführen oder durch Aufspalten der Punktmengen. Grundlage ist dabei stets ein Distanzmaß zwischen Punkten oder Clustern [2]. Ein weiteres distanzbasiertes Verfahren ist das K-Means-Clustering, das zur Klasse der partitionierenden Algorithmen zählt. Die Anzahl der Cluster wird dabei im Voraus festgelegt. Der Algorithmus initialisiert Clusterzentren und weist jedem Punkt das nächstgelegene Zentrum zu. Anschließend werden die Zentren iterativ angepasst, bis eine passende Zuordnung erreicht ist. Aufgrund der Nutzung von Clusterzentren spricht man hier auch von zentroidbasierter Clusterung [13]. KMeans wird auch das Verfahren sein, welches in dieser Arbeit vorgestellt und verwendet wird.

4 Clustern von Anfragen

4.1 Strukturansatz

In diesem Abschnitt wird die zugrunde liegende hierarchische Struktur erläutert, die bei der Generierung der Anfragen entsteht. Anschließend wird untersucht, wie sich Knoten innerhalb dieser Struktur vergleichen lassen. Dazu werden zunächst grundlegende Konzepte der Ähnlichkeitsberechnung vorgestellt, gefolgt von einer Methode zur Bestimmung der Ähnlichkeit zwischen den Knoten. Zudem wird eine Möglichkeit vorgestellt, die Differenzierung von Knoten zu verfeinern, bevor abschließend gezeigt wird, wie dieser Ansatz genutzt werden kann, um die relevanten Punkte für die Clusterung zu bestimmen.

Hierarchische Struktur Ein Graph G besteht aus einer Menge von Knoten V und einer Menge von Kanten E. Hierbei ist die Menge $E \subseteq V \times V$, sodass jede Kante als Tupel aus zwei Knoten dargestellt wird. Wenn ein Tupel (v_1, v_2) in der Menge der Kanten enthalten ist, so sind die Knoten v_1 und v_2 durch eine Kante miteinander verbunden [30]. Mithilfe des im Paper [29] vorgestellten Algorithmus wird ein Graph entwickelt, der alle Anfragen miteinander in Verbindung setzt. Ein Merkmal dieses Graphen ist, dass er auf einer hierarchischen Struktur basiert. Ähnlich zu einer Baumstruktur stehen die Knoten in einer Eltern-Kind-Beziehung, jedoch mit dem Unterschied, dass ein Kindknoten

nicht nur einen Elternknoten, sondern mehrere haben kann. Zwei Anfragen stehen in einer Eltern-Kind-Beziehung, wenn die Kind-Anfrage eine spezifischere Definition als die Eltern-Anfrage aufweist. Das bedeutet, dass entweder in der Eltern-Anfrage eine Variable oder ein Attribut hinzugefügt wird, oder weitere Ereignisse zum Elternknoten hinzukommen. Die Wurzel des Graphen entspricht dabei der leeren Anfrage " [29].

Distanzmetrik Die Struktur stellt die Anfragen bereits in eine Beziehung, jedoch fehlt eine geeignete Metrik, um diese Beziehungen gezielt zu vergleichen. Um dies zu ermöglichen, wird eine Distanzmetrik definiert, die die Position der Anfragen im Baum berücksichtigt. Die Grundlage bildet der kürzeste Pfad, der die minimale Länge des Pfades von einem Startknoten s zu einem Endknoten v beschreibt, wobei alle Kanten ein konstantes Gewicht von 1 haben [30]. Dieser Pfad ermöglicht den Vergleich von Knoten anhand der Distanz zwischen ihnen. Zusätzlich wird der kleinste gemeinsame Vorfahre (LCA) als weitere Grundlage für die Distanzmetrik herangezogen. Der LCA zwischen zwei Knoten u und v ist der am weitesten von der Wurzel entfernte Knoten, der beide Knoten als Nachkommen hat. Ein LCA höher im Baum weist auf größere Unterschiede hin, während ein tiefer gelegener LCA eine engere Verwandtschaft signalisiert [4]. Der LCA kann durch eine Breitensuche ermittelt werden, bei der schrittweise die Vorfahren beider Knoten gespeichert werden, bis ein gemeinsamer Vorfahre gefunden wird.
Mithilfe des LCA und des kürzesten Weges lässt sich nun eine komplexere Metrik nutzen. Wu und Palmer [34] haben eine Methode zur Berechnung der semantischen Ähnlichkeit von Konzepten in einer Baumstruktur entworfen. In dem Paper wird die Ähnlichkeit zwischen zwei Konzepten C_1 und C_2 innerhalb einer Struktur berechnet. Dafür wird ein drittes Konzept C_3 eingeführt, das als least common superconcept von C_1 und C_2 definiert ist. C_3 ist somit das kleinste gemeinsame Konzept, das beide Konzepte in der Baumstruktur teilen. Die Pfade zwischen den Konzepten werden durch N_i beschrieben. N_1 steht für die Anzahl der Knoten entlang des Pfades von C_1 zu C_3, N_2 für die Anzahl der Knoten entlang des Pfades von C_2 zu C_3, und N_3 für die Anzahl der Knoten von C_3 bis zur Wurzel des Baumes. Die Ähnlichkeit zwischen den beiden Konzepten C_1 und C_2 wird durch die folgende Formel berechnet: $\text{ConSim}(C_1, C_2) = \frac{2 \cdot N_3}{N_1 + N_2 + 2 \cdot N_3}$ Für die hierarchische Struktur der Ereignisanfragen kann die Formel modifiziert werden, indem die Begriffe an das Projekt angepasst werden. Das least common superconcept entspricht dem kleinsten gemeinsamen Vorfahren. Die Länge eines Pfades wird mit length() abgekürzt und bezieht sich auf den kürzesten Pfad, während die Tiefe eines Knotens mit depth() beschrieben wird, was die Länge des Pfades von einem Knoten zu der Wurzel des Baumes bezeichnet. Die umformulierte Gleichung lautet dann: $ConSim(C_1, C_2) = \frac{2 \cdot \text{depth(LCA)}}{\text{length}(C_1,\text{LCA}) + \text{length}(C_2,\text{LCA}) + 2 \cdot \text{depth(LCA)}}$
Diese Anpassung macht die Berechnung für die vorliegende Struktur anwendbar und ermöglicht somit die Bestimmung der Ähnlichkeiten zwischen den Anfragen.
Ein wesentlicher Vorteil dieser Berechnung liegt in der expliziten Berücksichtigung der Tiefe des LCA. Dadurch kann bewertet werden, ob die gemeinsame Basis der beiden Konzepte eher allgemein oder spezifisch ist. Ein flacher LCA deutet auf eine eher allgemeine Beziehung hin, während ein tiefer LCA auf eine engere und spezifischere Verbindung zwischen den Knoten hindeutet. Zusätzlich ermöglichen die in der Formel enthaltenen Pfadlängen die Einbeziehung der Distanz zwischen den Konzepten. So lässt sich bestim-

men, wie nah oder entfernt sich zwei Konzepte innerhalb der Baumstruktur sind. Einen niedrigen $ConSim()$-Wert erhalten Knotenpaare, wenn sie sowohl eine geringe Distanz zueinander als auch einen gemeinsamen Vorfahren in einer tiefen Baumebene haben, was auf eine hohe Ähnlichkeit hinweist. Durch die Berücksichtigung sowohl der hierarchischen Struktur als auch der Distanzen zwischen den Knoten wird die Wu/Palmer Metrik in diesem Projekt zur Berechnung der Distanzen eingesetzt.

Wichtungen Der berechnete Wu/Palmer-Wert zwischen zwei Knoten in einer hierarchischen Struktur kann oft zu identischen Ergebnissen führen, insbesondere wenn die Knoten auf derselben Ebene liegen. Dadurch erhalten mehrere Anfragen die gleiche Distanz. Um eine differenziertere Bewertung zu ermöglichen, wird die Metrik durch den Einsatz von Gewichten angepasst. Wie bereits beschrieben, unterscheiden sich zwei Anfragen, wenn sie durch eine Kante verbunden sind. Dies kann durch die Erweiterung von Terminalsymbolen, Variablen oder durch eine veränderte Länge geschehen. Letzteres ergibt sich zwangsläufig, wenn ein weiteres Ereignis zwischen Eltern- und Kindknoten hinzugefügt wird. Diese verschiedenen Variationen ermöglichen die Zuweisung von Gewichten zu den Kanten, sodass die Wu/Palmer-Metrik nicht mehr die Weglänge zwischen den Knoten erfasst, sondern stattdessen die Summe der zugewiesenen Gewichte nutzt. Dadurch können Knoten, die in derselben Entfernung zueinander stehen, differenzierte Ähnlichkeitswerte erhalten. Zur Berechnung des Gewichts g einer Kante werden die Unterschiede zwischen dem Eltern- und dem Kindknoten erfasst und gezählt. Dabei wird bestimmt, wie viele Variablen (var), Terminale (term) und Ereignisse (ev) hinzugekommen sind. Das Gewicht g ergibt sich anschließend aus der Formel: $g = \alpha \cdot \text{var} + \beta \cdot \text{term} + \gamma \cdot \text{ev}$. Dadurch ergeben sich drei Anpassungsmöglichkeiten, um das endgültige Gewicht einer Kante zu beeinflussen. Das entsprechende Gewicht muss reduziert werden, wenn der jeweilige Unterschied nur eine geringe Auswirkung auf die Ähnlichkeit haben soll. Die genaue Initialisierung wird später im Praxisteil erläutert. Bereits im Vorfeld kann jedoch festgelegt werden, in welcher Reihenfolge die Gewichte vergeben werden. Der Unterschied in der Länge erhält das kleinste Gewicht, da er die geringste Auswirkung auf die Unterscheidung von Anfragen hat. Dies liegt daran, dass eine größere Länge keine zusätzlichen Einschränkungen für die Werte in den Ereignissen mit sich bringt. Variablen werden mit einem höheren Gewicht versehen, da ihr Wert noch veränderlich ist und somit keine strikte Abgrenzung zwischen einer Anfrage mit und ohne Variablen besteht. Die höchste Gewichtung erhält die Kante, an der sich die Anfragen durch ein Terminalsymbol unterscheiden, da dies die größte Einschränkung darstellt, bei der der Wert des Attributes fest definiert wird. Nachdem die gewichteten Kanten eingeführt wurden, muss auch die Funktionsweise der bestehenden Metriken betrachtet und angepasst werden. Zwar wird weiterhin eine Form des kürzesten Weges genutzt, doch durch die Gewichtung verändert sich dessen Definition. Der kleinste gemeinsame Pfad ist nun nicht mehr derjenige mit den wenigsten dazwischenliegenden Knoten, sondern der mit dem geringsten Gesamtgewicht (siehe Beispiel 3).

Beispiel 3: *Angenommen, der Graph $G = (V, E)$ (Abbildung 1) ist gegeben, wobei $V = \{v_1, v_2, v_3\}$ und $E = \{(v_1, v_2, 1), (v_2, v_3, 1), (v_1, v_3, 4)\}$. Ziel ist es, den Wert des*

kürzesten Pfades von v_1 nach v_3 zu berechnen. Durch die Einführung der Gewichte ist der kürzeste Pfad nun nicht mehr der direkte Weg von v_1 nach v_3, da dieser ein hohes Gewicht von 4 aufweist. Stattdessen führt der kürzeste Pfad über v_2, da die Summe der Kantengewichte mit einem Wert von 2 geringer ist als der direkte Weg.

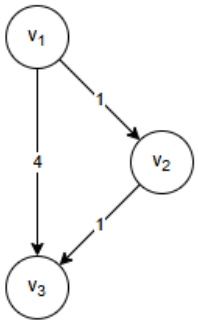

Abbildung 1: Gerichteter Graph mit Gewichten

Der kleinste gemeinsame Vorfahre bleibt weiterhin gleich definiert, die einzige Ausnahme ist, wenn es zwei Vorfahren in der selben Ebene gibt, so wird der Vorfahre gewählt, bei dem der Kombinierte Pfad zu den beiden Knoten am geringsten ist.

Multidimensionale Skalierung Mithilfe von Wu/Palmer und den Wichtungen können die Distanzen zwischen den Knoten berechnet und eine Distanzmatrix erstellt werden. Für die spätere Clusterung sind jedoch Koordinaten erforderlich, da der Clusteralgorithmus den euklidischen Abstand verwendet (siehe Abschnitt 4.4). Um aus der Distanzmatrix eine Punktematrix zu erstellen, kann ein MDS-Algorithmus verwendet werden. MDS befasst sich mit dem Problem, aus einer Tabelle von Distanzen (Abbildung 2) einen Raum zu generieren, in dem die einzelnen Objekte als Punkte dargestellt werden (Abbildung 3). Dafür wird eine Tabelle verwendet, die zeigt, wie ähnlich oder unterschiedlich die Objekte zueinander sind. Das Ergebnis ist eine geometrische Anordnung der Punkte in einem multidimensionalen Raum, wobei jeder Punkt einem Objekt aus der Tabelle entspricht. Bei der Berechnung gilt, dass eine größere Distanz zwischen den Objekten auch eine größere räumliche Trennung im Raum zur Folge hat. Die Berechnung, die ein MDS-Algorithmus verwendet, ist recht komplex und kann in den einfachsten Versionen nicht ohne die Unterstützung von Computern durchgeführt werden [19]. Eine vollständige Beschreibung des Prozesses für das klassische MDS kann in dem Artikel [32] nachgelesen werden.

CITIES	ATLA.	CHIC.	DENV.	HOUS.	L.A.	MIAMI	N.Y.	S.F.	SEAT.	WASH D.C.
ATLANTA		587	1212	701	1936	604	748	2139	2182	543
CHICAGO	587		920	940	1745	1188	713	1858	1737	597
DENVER	1212	920		879	831	1726	1631	949	1021	1494
HOUSTON	701	940	879		1374	968	1420	1645	1891	1220
LOS ANGELES	1936	1745	831	1374		2339	2451	347	959	2300
MIAMI	604	1188	1726	968	2339		1092	2594	2734	923
NEW YORK	748	713	1631	1420	2451	1092		2571	2408	205
SAN FRANCISCO	2139	1858	949	1645	347	2594	2571		678	2442
SEATTLE	2182	1737	1021	1891	959	2734	2408	678		2329
WASHINGTON D.C.	543	597	1494	1220	2300	923	205	2442	2329	

Abbildung 2: Distanzmatrix, Städte in USA "Multidimensional Scaling", Kruskal und Wish, 1978

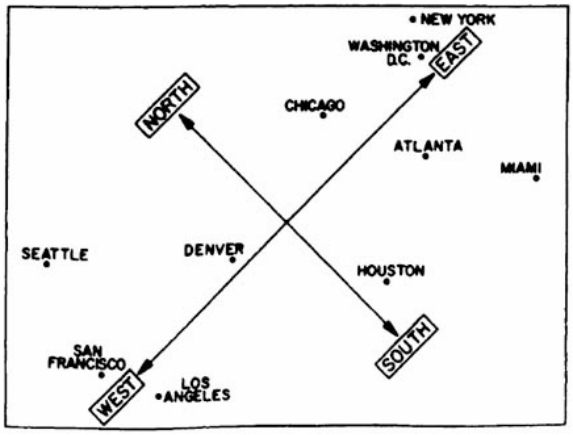

Abbildung 3: Ausgabe eines MDS-Algorithmus "Multidimensional Scaling", Kruskal und Wish, 1978

4.2 Syntaktischer Ansatz

In diesem Abschnitt wird der syntaktische Ansatz zur Vektorkonstruktion näher erläutert. Dabei erfolgt die Ähnlichkeitsbewertung der Anfragen auf Grundlage ihrer strukturellen Merkmale. Ziel ist es, dass ähnlich aufgebaute Anfragen im Vektorraum nahe beieinanderliegen, während sich strukturell unterschiedliche Anfragen weiter voneinander entfernen. Zur syntaktischen Gruppierung eignet sich ein Exkurs in das Information Retrieval. Information Retrieval beschäftigt sich mit der gezielten Suche nach relevanten Informationen in großen Mengen unstrukturierter Daten, etwa in Dokumentensammlungen. Ein zentrales Anwendungsfeld ist die Websuche, dort geben Nutzer Schlagwörter ein und die Suchmaschine soll passende Ergebnisse liefern. Aufgrund des Umfangs des Webs ist die effiziente Identifikation relevanter Inhalte eine Herausforderung. Daher kommen im Information Retrieval verschiedene methodische Verfahren zum Einsatz, um Treffergenauigkeit und Relevanzbewertung zu verbessern [22].

Vector Space Model Ein Konzept, das später adaptiert wird, um syntaktische Cluster zu bilden, ist das Vector Space Model. Die grundlegende Idee dieses Modells besteht darin, Anfragen und Dokumente als Vektoren in einem hochdimensionalen Raum darzustellen und dort zu vergleichen. Grundlage ist ein festes Vokabular K, das alle in den Dokumenten vorkommenden Wörter umfasst. Jede Anfrage und jedes Dokument wird als Punkt in einem $|K|$-dimensionalen Raum interpretiert, wobei jede Dimension einem bestimmten Begriff entspricht. Dokumente mit ähnlichem Wortbestand liegen näher beieinander, da angenommen wird, dass sie auch inhaltlich verwandt sind [22]. Ein Beispiel zur Veranschaulichung findet sich in Beispiel 4. Ein zentraler Vorteil des Modells liegt darin, dass auch bei teilweiser Übereinstimmung eine Ähnlichkeitsbewertung möglich ist. Diese erfolgt über den Abstand oder häufiger über den Winkel zwischen den Vektoren. Letzterer berücksichtigt nicht die Länge der Vektoren, wodurch auch

unterschiedlich lange Dokumente vergleichbar bleiben. Die Ähnlichkeit zwischen einer Anfrage q und einem Dokument d wird dabei wie folgt berechnet: $\cos(\theta) = \frac{\langle q,d\rangle}{\|q\|\cdot\|d\|}$ Dabei bezeichnet $\langle q, d\rangle$ das Skalarprodukt der Vektoren q und d, während $\|q\|$ und $\|d\|$ deren euklidischer Norm entspricht. Um den Winkel θ vollständig zu berechnen, wendet man die Arkuskosinus-Funktion an: $\theta = \arccos\left(\frac{\langle q,d\rangle}{\|q\|\cdot\|d\|}\right)$
Je kleiner der Winkel θ zwischen den Vektoren q und d, desto ähnlicher ist die Anfrage zu dem verglichenem Dokument [22].

Beispiel 4: *Gegeben seien drei Sätzen:*
s_1 *: „Das Auto parkt vor dem Haus.“* s_2 *: „Das Auto parkt hinter dem Gebäude.“*
s_3 *: „Das Tier steht im Tierpark.“*

Schritt 1 Erstellung des Vokabulars: Zunächst wird das Vokabular K definiert, das die Menge aller in den Sätzen vorkommenden Wörter umfasst. Dabei wird jedes Wort nur einmal aufgenommen, sodass sich die folgende Liste ergibt:

$$K = [\text{Das}, \text{Auto}, \text{parkt}, \text{vor}, \text{dem}, \text{Haus}, \text{hinter}, \text{Gebäude}, \text{Tier}, \text{steht}, \text{im}, \text{Tierpark}]$$

In einer realen Anwendung würde man typischerweise eine Normalisierung durchführen, um unterschiedliche Formen wie „parken“ und „parkt“ als identisch zu behandeln. Für dieses Beispiel verzichten wir jedoch darauf, da es nicht notwendig ist.

Schritt 2 Konstruktion der Vektoren: Jeder Satz wird nun als Vektor in einem $|K|$-dimensionalen Raum dargestellt. Eine Dimension wird erhöht, wenn das entsprechende Wort im Satz vorkommt. Daraus ergeben sich die folgenden Vektoren:

$$\begin{aligned} s_1 &= [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T \\ s_2 &= [1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]^T \\ s_3 &= [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1]^T \end{aligned}$$

Schritt 3 Berechnung der Ähnlichkeit: Zur Quantifizierung der Ähnlichkeit (sim) zwischen den Sätzen wird paarweise der Kosinus $sim(s_i, s_j)$ für jedes Satzpaar berechnet: $sim(s_i, s_j) = \frac{\langle s_i,s_j\rangle}{\|s_i\|\cdot\|s_j\|}$. Die resultierende Ähnlichkeitsmatrix lautet:

$\text{sim}(s_i, s_j)$	s_1	s_2	s_3
s_1	1	0.816	0.408
s_2	0.816	1	0.447
s_3	0.408	0.447	1

Es gilt, je näher der Wert an 1 ist, desto kleiner der Winkel und desto ähnlicher sind sich die Vektoren. Die Matrix zeigt, dass die Sätze s_1 und s_2 einen höheren Wert aufweisen als zu s_3. Es zeigt, dass s_1 und s_2 sich strukturell ähnlicher sind, weil sie mehr gemeinsame Wörter teilen und daher im Vektorraum näher beieinanderliegen als zu s_3.

Ereigniswörterbuch Das vorgestellte Konzept wird nun auf das Problem der An-

frageclusterung übertragen. Im Kern geht es darum, Anfragen als Punkte in einem Vektorraum zu repräsentieren und später ein Clustering-Verfahren anzuwenden. Da keine separaten Dokumente existieren, reduziert sich der Aufwand erheblich, sodass die Anfragen die Rolle der Dokumente übernehmen. Der Schwerpunkt liegt auf der Tokenisierung der Anfragen, wobei die enthaltenen Ereignisse wie Wörter in Dokumenten behandelt werden. Das Vokabular setzt sich hierbei aus der Menge aller unterschiedlichen Ereignisse zusammen, die in den Anfragen vorkommen, wobei die Größe des Vektorraums der Anzahl dieser einzigartigen Ereignisse entspricht. Nach der Definition des Vokabulars wird jede Anfrage schrittweise verarbeitet. Tritt ein Ereignis in einer Anfrage auf, wird die entsprechende Dimension im Vektorraum um eins erhöht. Dadurch erhält jede Anfrage eine Vektorrepräsentation, die anschließend für die Clusterung verwendet werden kann.

N-Gramme Häufig enthalten Suchanfragen zusammenhängende Wortgruppen, sogenannte Phrasenanfragen. Diese treten insbesondere bei Produktnamen, Organisationen oder komplexen Konzepten auf. Ein klassisches Beispiel wäre die Anfrage „Technische Universität Berlin“, die nicht mit dem Textausschnitt „Die technische Ausstattung einer Universität in Berlin wird immer schlechter“ übereinstimmen sollte. Um dieses Problem zu adressieren, stellen die Autoren den Ansatz des Biword-Index vor. Dabei werden alle Paare aufeinanderfolgender Wörter (Bigramme) in einem Dokument erfasst und als eigenständige Terme in das Vokabular aufgenommen. Bestehen die Phrasen aus mehr als zwei Wörtern so werden sie in aufeinanderfolgende Bi-Wörter zerlegt [22]. Dieser Ansatz wird für die Anfragen adaptiert. Denn aufeinanderfolgende Ereignisse können auch gemeinsame semantische Bedeutungen haben, beispielsweise wenn dieselbe Variable in mehreren Ereignissen vorkommt und letztlich das gleiche Attribut referenziert wird. Um diese Zusammenhänge zu erfassen, werden die Ereignisse innerhalb der Anfragen in eine Liste von Bigrammen zerlegt, auf deren Grundlage ein entsprechendes Wörterbuch erstellt wird. Im Praxisteil wird zudem untersucht, ob es sinnvoll ist, den Ansatz von Bigrammen auf N-Gramme zu erweitern. N-Gramme verallgemeinern das Konzept, indem sie nicht nur zwei, sondern beliebig viele aufeinanderfolgende Elemente erfassen. In dem Fall also Sequenzen von N Ereignisse.

Normalisierung Im Information Retrieval ist die Vorverarbeitung von Wort-Tokens essenziell, insbesondere die Normalisierung, die semantisch identische, aber formal unterschiedliche Tokens einander angleicht. Dazu zählen Verfahren wie die Vereinheitlichung von Groß- und Kleinschreibung sowie die Entfernung diakritischer Zeichen. So werden etwa „das“ und „Das“ als gleichwertig behandelt, da die Großschreibung am Satzanfang keinen Bedeutungsunterschied darstellt. Dies erhöht die Treffergenauigkeit und verbessert die Relevanz der Suchergebnisse [22]. Ein zentrales Problem bei Anfragen ist die Variablenbenennung semantisch identischer Ereignisse, etwa durch unterschiedlich eingeführte Variablen in langen und kurzen Anfragen (siehe Beispiel 5). Dies erschwert das Matching, obwohl die inhaltliche Bedeutung gleich bleibt. Eine Normalisierung dieser Variablen erhöht die Konsistenz im Vektorraum und verbessert die Vergleichbarkeit. Aus diesem Grund wird ein entsprechendes Normalisierungsverfahren auch für Anfragen eingeführt.

Beispiel 5: *Gegeben seien zwei 3-dimensionale Anfragen:*

$$a_1 = x_1;;x_1\ x_2;x_2; \qquad a_2 = x_1;x_1;$$

Ohne Normalisierung hätten diese Anfragen keine gemeinsamen Ereignisse und würden daher als unähnlich betrachtet werden. Allerdings erkennt man, dass $a_1(2) = a_2(1)$ *gilt, wenn die Variablen einheitlich benannt werden.*

Um solche Ähnlichkeiten korrekt abzubilden, werden die Ereignisse vor ihrer Aufnahme in das Vokabular normalisiert. Dabei bleiben die Zeichen des Terminalalphabets unverändert und die Variablen innerhalb der Ereignisse werden normalisiert, sobald sie in das Ereignis-Vokabular aufgenommen werden. Dazu wird eine neue Menge M definiert, die disjunkt von der bisherigen Variablenmenge VAR sowie dem Terminalalphabet Γ ist. Bei der Ersetzung werden die Variablen in den betrachteten Ereignissen systematisch durch ein festgelegtes Zeichen aus M getauscht, sodass identische strukturelle Muster zwischen den Anfragen beibehalten werden. Diese Methode lässt sich auch auf eine beliebige Länge von N-Grammen erweitern. Die konkrete Umsetzung dieses Normalisierungsverfahrens wird in Kapitel 5.2 detaillierter beschrieben.

Attributebene Der bisherige Ansatz ermöglicht einen effizienten Vergleich von Anfragen, die gemeinsame Ereignisse enthalten. Ein Problem tritt jedoch auf, wenn zwei Anfragen keine direkten Ereignis-Übereinstimmungen aufweisen (Beispiel 6). Um dieses Problem zu lösen werden auch Attribute in ein Vokabular gespeichert. Die grundlegende Methode bleibt dabei unverändert. Für jede Anfrage werden die enthaltenen Attribute extrahiert und in ein separates Vokabular überführt. Anschließend wird überprüft, welche Attribute eine Anfrage enthält, indem erneut über alle Attribute iteriert wird. Dabei ist es entscheidend, dass nur Attribute als übereinstimmend gelten, wenn sie in derselben Dimension der Anfrage vorkommen.

Beispiel 6: *Betrachtet werden zwei gegebene Anfragen* a_1 *und* a_2*, die jeweils aus zwei Ereignisse mit drei Attributen bestehen:*

$$a_1 = ;s_{10};\ ;s_9;s_8 \qquad a_2 = x_1;s_{10};\ x_1;s_9;$$

Nach der bisherigen Vektorisierung ergibt sich keine direkte Ähnlichkeit zwischen den Anfragen, da weder gemeinsame Ereignisse noch n-Gramme von Ereignissen vorhanden sind. Dennoch weisen die enthaltenen Attribute Übereinstimmungen auf, was für den Ansatz genutzt wird. Dafür wird zunächst ein Vokabular der einzelnen Attribute erstellt, wobei Variablen und Platzhalter unberücksichtigt bleiben, da sie keine direkten Einschränkungen für das Matching darstellen. Das resultierende Attribut-Vokabular lautet: $[s_{10}, s_9, s_8]$*. Woraus folgende Vektoren resultieren:*

$$v(a_1) = [1,1,1]^T \quad v(a_2) = [1,1,0]^T$$

Dies zeigt, dass durch diesen Ansatz die beiden Anfragen nicht mehr vollständig disjunkt sind, wodurch eine präzisere Ähnlichkeitsbewertung ermöglicht wird.

Ein wichtiger Aspekt zeigt sich bei der Erstellung der N-Gramme. Während bei den Ereignissen die sequentielle Reihenfolge der Elemente eine zentrale Rolle spielt, hängt die Bildung der N-Gramme für Attribute von deren Position innerhalb der Anfragen ab. Um

die Idee dahinter besser zu verstehen siehe Beispiel 7. Entscheidend ist also, in welchen Dimensionen die Attribute auftreten. Das Ziel besteht darin, Anfragen zu belohnen, die in bestimmten Dimensionen ähnliche Abfolgen von Attributen enthalten. Dadurch lassen sich Gemeinsamkeiten auf Attributebene erfassen, sodass Anfragen präziser miteinander verglichen werden können.

Beispiel 7:
Gegeben sind drei Ereignisse einer Anfrage: a;_;$x0, _;b;$x0 und a;_;_. Bisher wurde eine Attributfolge auf Basis ihres Auftretens innerhalb eines Ereignisses verglichen (siehe Abbildung 4, Kreis 1). Die neue Idee besteht darin, die Attributfolgen stattdessen entlang ihrer jeweiligen Dimensionen zu bilden (siehe Abbildung 4, Kreis 2). Dadurch können Anfragen, die zwar unterschiedliche Ereignisse enthalten, aber in bestimmten Dimensionen wiederholt gleiche Attributfolgen aufweisen, ebenfalls als ähnlich erkannt werden.

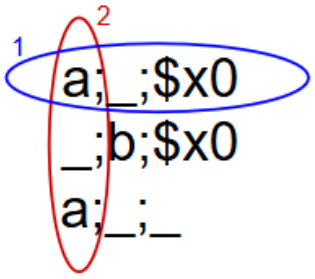

Abbildung 4

Neben den einzelnen Attribut-N-Grammen wird deshalb auch die jeweilige Dimension gespeichert, in der sie auftreten. Dadurch wird verhindert, dass identische Attributfolgen, die in unterschiedlichen Dimensionen vorkommen, fälschlicherweise als ähnlich gewertet werden (siehe Beispiel 8). Im Gegensatz zur Verarbeitung einzelner Attribute werden hier auch Variablen und Platzhalter berücksichtigt, da sie strukturelle Gemeinsamkeiten zwischen Anfragen aufzeigen können.

Beispiel 8: *Die Bi-Gramme für die beiden Anfragen aus Beispiel 6 lauten:*

$$\text{Für } a_1: \quad [(\_,\_),(s_{10},s_9),(\_,s_8)] \qquad \text{Für } a_2: \quad [(x_1,x_1),(s_{10},s_9),(\_,\_)]$$

Hierbei ist zu beachten, dass das Bi-Gramm $(\_,\_)$ aus der ersten Dimension von a_1 keinen Zusammenhang mit dem Bi-Gramm $(\_,\_)$ aus der dritten Dimension von a_2 hat. Da sie in unterschiedlichen Dimensionen auftreten, könnten in diesen Positionen völlig verschiedene Attribute verwendet werden. Die Aufmerksamkeit liegt auf der zweiten Dimension, in der beide Anfragen eine identische Attributfolge besitzen. So enthalten beide Anfragen in dieser Dimension die Sequenz $s_{10} \rightarrow s_9$. Daher wird in der Abschlussmatrix für diese Position eine „1“ gesetzt, wodurch die Vektoren der beiden Anfragen näher zusammenrücken und eine höhere Ähnlichkeit erhalten.

Wichtung In diesem Kapitel wurden verschiedene Merkmale betrachtet, um Anfragen syntaktisch zu vergleichen. Dazu gehören Ereignisse, Attributwerte und deren n-Gramme, die bisher gleich gewichtet werden. Allerdings sollte ein identisches Ereignis stärker zur Ähnlichkeit beitragen als ein gemeinsamer Attributwert, was eine differenzierte Gewichtung notwendig macht. Wie bei hierarchischen Strukturen lassen sich Gewichtungen auch im Vektorraum integrieren. Statt Kantengewichten werden dabei die entsprechenden

Vektorkomponenten mit einem Faktor k multipliziert. Dadurch verändern sich die Positionen der Punkte im Raum und rücken je nach Gewichtung näher zusammen oder weiter auseinander [5]. Sollen bestimmte Eigenschaften betont werden, werden ihre zugehörigen Dimensionen stärker gewichtet. Das führt dazu, dass Elemente mit dieser Eigenschaft näher beieinander liegen, während andere Elemente, ohne diese Eigenschaft, weiter entfernt sind. Im Praxisteil wird untersucht, welche Werte sich für die Gewichtung eignen. Die grundsätzliche Reihenfolge lässt sich jedoch bereits festlegen. Gemeinsame Attributwerte erhalten das geringste Gewicht, n-Gramme von Attributen und einzelne Ereignisse ein mittleres, während Ereignis-n-Gramme die höchste Bedeutung haben. Letztere deuten auf Übereinstimmungen über mehrere Dimensionen hinweg hin und weisen auf eine besonders hohe syntaktische Ähnlichkeit zwischen den Anfragen hin.

4.3 Semantischer Ansatz

Dieser Abschnitt beschäftigt sich mit der Methodik, Anfragen basierend auf ihrer semantischen Gleichheit zu vektorisieren. Zunächst wird der Unterschied zwischen Semantik und Syntaktik in Beispiel 9 veranschaulicht. Anschließend werden die Grundlagen neuronaler Netze erläutert, die eine entscheidende Rolle bei der semantischen Abbildung spielen. Daraufhin wird eine aktuell weit verbreitete Methode zur semantischen Repräsentation vorgestellt, die auf neuronalen Netzen basiert, bevor abschließend eine weiterentwickelte Methode präsentiert wird, die besser auf die Anforderungen dieses Projekts zugeschnitten ist.

Beispiel 9: *Angenommen, wir haben die drei deutschen Sätze s_1, s_2 und s_3: s_1 = "Der Hund jagt den Ball.", s_2 = "Der Hund jagt die Katze." und s_3 = "Die Katze wird vom Hund gejagt." Man erkennt, dass s_1 und s_2 syntaktisch sehr ähnlich sind, sich jedoch inhaltlich unterscheiden: In s_1 wird der Ball gejagt, in s_2 die Katze. s_3 trägt dieselbe Bedeutung wie s_2, ist aber syntaktisch anders strukturiert. Bei Anfragen verhält es sich ähnlich. Zwei Ereignisse können semantisch verwandt sein, wenn ihre Bestandteile ähnliche Bedeutungen tragen. So lassen sich beispielsweise die eindimensionalen Anfragen (a), ($x0), ($x0) und (a), (b), (b) als semantisch ähnlich interpretieren, obwohl sie unterschiedliche Ereignisse enthalten. Es gibt jedoch auch Fälle, in denen sich die semantische Ähnlichkeit erst aus dem Kontext ergibt. Etwa wenn beide Ereignisse regelmäßig von demselben anderen Ereignis eingeleitet werden.*

Neuronale Netze sind von der Struktur des menschlichen Nervensystems inspiriert und dienen der Simulation von Lernprozessen. Während im biologischen System Neuronen über Synapsen Signale weiterleiten und ihre Verbindungsstärke anpassen können, übernehmen in künstlichen Netzen Recheneinheiten diese Funktion. Die Verbindung zwischen ihnen erfolgt über Zahlengewichte, die analog zur synaptischen Stärke den Einfluss eines Signals auf nachfolgende Einheiten bestimmen. Das Netzwerk erzeugt auf Basis gegebener Eingaben eine Ausgabe. Lernen erfolgt dabei durch die Anpassung der Gewichte, sodass die Ausgaben schrittweise näher an die gewünschten Zielwerte heranrücken. Dafür wird ein Trainingsdatensatz mit bekannten Eingabe-Ausgabe-Paaren

verwendet. Je mehr solcher Paare zur Verfügung stehen, desto besser kann das Netz Muster erkennen, die Wahl seiner Gewichte verbessern und somit genauere sowie verallgemeinerbare Vorhersagen treffen. [1] Das einfachste neuronale Netzwerk besteht lediglich aus einer Eingabeschicht und einer Ausgabeschicht. Ein solches Modell wird Perceptron genannt (siehe Abbildung 5). Da die Eingabeschicht selbst keine Berechnungen durchführt, finden alle mathematischen Operationen im Ausgabeknoten statt.

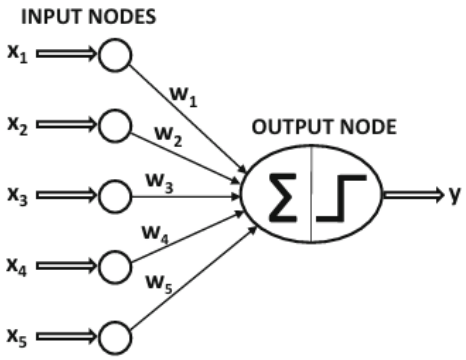

Abbildung 5: Perceptron "Neural Networks and Deep Learning", Charu C. Aggarwal, 2023

Da es sich um einen einzelnen Knoten handelt, wird im Perceptron eine lineare Funktion genutzt, um den Eingabevektor X mit dem Gewichtsvektor W zu verrechnen: $W \cdot X^T = \sum_{i=1}^{d} w_i \cdot x_i$. Das Ergebnis wird dann durch eine Aktivierungsfunktion weiterverarbeitet, die entscheidet, ob und in welcher Form das Signal weitergegeben wird. Eine gängige Funktion ist die ReLU-Aktivierung, die negative Werte auf null setzt, oder die Sign-Funktion, die den Wert auf +1 oder -1 abbildet und das Perceptron somit als binären Klassifikator verwendet. Weicht die Vorhersage von der tatsächlichen Klassifikation ab, wird der Fehler mit einer Loss-Funktion berechnet. Diese zeigt, wie stark die Vorhersage vom tatsächlichen Wert abweicht. Ziel des Trainings ist es, den Fehler über den gesamten Datensatz hinweg zu minimieren. Zu Beginn sind die Gewichte zufällig, aber während des Trainings lernt das Netzwerk durch Fehlerkorrekturen und passt die Gewichte so an, dass die Klassifikationsgenauigkeit steigt [1]. Neuronale Netzwerke mit mehr als einer berechnenden Schicht werden als Multilayer-Netzwerke bezeichnet. Die zusätzlichen Schichten, die sich zwischen der Eingabe- und der Ausgabeschicht befinden, nennt man verdeckte Schichten (Hidden Layers), da ihre Berechnungen für den Nutzer nicht direkt sichtbar sind (Abbildung 6). Die Architektur eines solchen Netzwerks ist dennoch relativ einfach zu verstehen. Es funktioniert wie eine Aneinanderreihung mehrerer Perceptrons. Die Ausgabe einer Schicht dient hierbei aber als Eingabe für die nächste Schicht, und dieser Prozess setzt sich fort, bis die finale Ausgabeschicht erreicht ist. Multilayer-Netzwerke sind leistungsfähiger und flexibler, da sie komplexe Zusammenhänge besser erfassen können. Während ein einfaches Perceptron nur linear trennbare Probleme lösen kann, ermöglichen versteckte Schichten die Modellierung nichtlinearer Zusammenhänge [1].

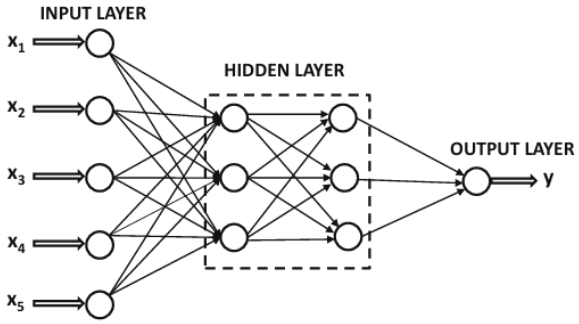

Abbildung 6: Neuronales Netz, "Neural Networks and Deep Learning", Charu C. Aggarwal, 2023

Um die semantische Bedeutung von Wörtern zu erfassen, werden Methoden verwendet, bei denen Wörter mithilfe neuronaler Netze in Vektoren überführt werden [31]. Ein gängiger Ansatz ist die Verwendung von Word Embeddings, bei denen Wörter in einem hochdimensionalen Raum so dargestellt werden, dass semantisch ähnliche Begriffe nahe beieinander liegen [23]. Da die verwendeten Anfragen auf einer eigenen, neuen Grammatik mit bisher nicht berücksichtigten Wörtern basieren, können vortrainierte Embeddings nicht direkt genutzt werden. Stattdessen ist es notwendig, ein eigenes Modell speziell für diese Grammatik zu trainieren, um die semantischen Beziehungen innerhalb des gegebenen Vokabulars angemessen abzubilden. Dazu werden bei dem Training des Modells die einzelnen Anfragen als normale Sätze betrachtet, wobei die Ereignisse innerhalb der Anfragen als Wörter fungieren. Das Ziel ist es, die Semantik der einzelnen Ereignisse zu erfassen und die Anfragen schließlich mithilfe der Wort- bzw. Ereignis-Embeddings zu vektorisieren.

Word2Vec ist ein vielversprechendes Modell zur semantischen Repräsentation von Wörtern, das 2013 von Mikolov et al. vorgestellt wurde. Es ermöglicht, ein neuronales Netz speziell für eigene Grammatiken zu trainieren. Der Ansatz basiert auf einem flachen neuronalen Netz mit nur einer versteckten Schicht, wodurch die Komplexität und der Rechenaufwand erheblich reduziert werden. Dadurch konnte das Modell effizient auf großen Datenmengen trainiert werden und übertraf frühere Methoden wie SemEval-2012 Task 2 oder Latent Semantic Analysis [23]. Zur Umwandlung von Wörtern in Vektoren stellte das Paper zwei Methoden vor. Eine davon ist die Continuous Bag-of-Words (CBOW) Methode (Abbildung 7 links). Hierbei wird versucht, das aktuelle Wort aus seinem Kontext vorherzusagen. Dazu werden die Vektordarstellungen der umliegenden Wörter in einem bestimmten Bereich vor und hinter dem Wort in eine Projektionsebene eingespeist. Diese kombiniert die Eingaben und erzeugt daraus den Vektor für das gesuchte Wort. Für ihre Datensätze wählten die Autoren einen Kontextbereich von jeweils vier Wörtern vor und nach dem Zielwort, da diese Konfiguration die besten Ergebnisse in diesem Fall lieferte. Die Reihenfolge der Wörter spielt in diesem Modell keine Rolle, da alle Vektoren der Kontextwörter auf dieselben Positionen projiziert werden. Aus diesem Grund ist der Name der Methode an das Bag-of-Words Prinzip angelehnt [23].

Der zweite Ansatz ist der Skip-Gram-Ansatz. Er funktioniert ähnlich wie CBOW, dreht jedoch das Prinzip um. Statt das aktuelle Wort aus dem Kontext vorherzusagen, wird versucht, mithilfe des aktuellen Wortes die Wortembeddings der umliegenden Wörter zu optimieren. Wie in der Abbildung 7 (rechts) dargestellt, wird auch hier die versteckte Schicht durch eine Projektionsebene ersetzt, um den Berechnungsaufwand zu reduzieren. Ein entscheidender Aspekt bei Skip-Gram ist die Wahl des Kontextbereichs. Während größere Distanzen zwischen Wörtern die Qualität der Embeddings verbessern können, erhöhen sie auch den Rechenaufwand erheblich. Deshalb wird eine maximale Wortdistanz c definiert. Für jedes Trainingswort wird zufällig eine Zahl R zwischen 1 und c gewählt, sodass das Modell die Wortembeddings für R Wörter vor und R Wörter nach dem aktuellen Wort nutzt [23].

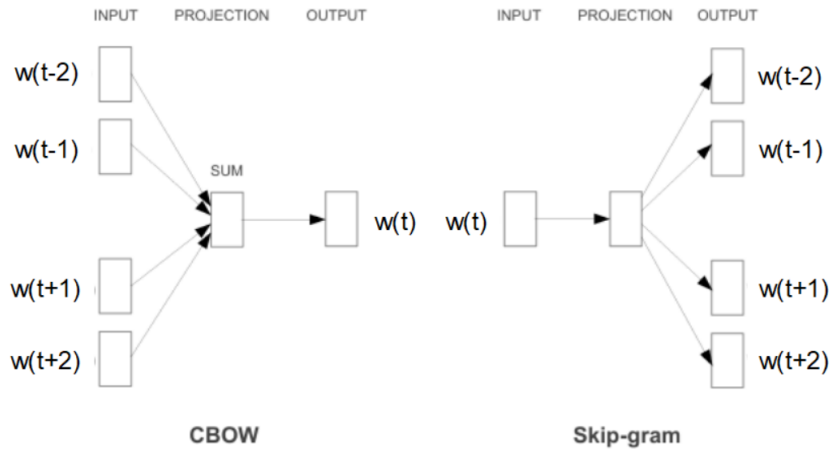

Abbildung 7: Model Architekturen Ëfficient Estimation of Word Representations in Vector Space", Mikolov et al, 2013

Ein zentrales Problem von Word2Vec ist jedoch, dass es für große Korpora mit Millionen oder sogar Milliarden von Wörtern optimiert wurde. Da die vorliegenden Anfragen bereits eine stark begrenzte Datenmenge darstellen, wäre es sinnvoll, alternative Methoden in Betracht zu ziehen.

FastText Ein Modell, das für die vorliegende Problemstellung vielversprechend sein könnte, ist eine Erweiterung von Word2Vec namens FastText. Dieses Modell wurde 2016 von einem Facebook-Team vorgestellt und baut auf der Skip-Gram-Methode von Mikolov et al. [23] auf. Nach [14] ist Skip-Gram besonders für kleinere Datenmengen und seltene Wörter geeignet, weshalb eine darauf basierende Erweiterung in diesem Fall vorteilhaft sein könnte. Die Autoren von FastText identifizierten ein zentrales Problem in den bisherigen Modellen darin, dass die innere Struktur von Wörtern nicht berücksichtigt wurde. Insbesondere Präfixe und Suffixe wurden bisher nicht berücksichtigt. Um dieses Problem zu lösen, führt FastText eine neue Methode ein, bei der jedes Wort als eine Sammlung von Buchstaben-N-Grammen dargestellt wird. Das Wort-Embedding, das in der Skip-gram-Methode verwendet wird, ergibt sich dann aus der Summe der N-Gram-

Embeddings [6]. Dazu wird an den Wortanfang das Zeichen '⟨' und an das Wortende das Zeichen '⟩' gesetzt, um Präfixe und Suffixe von normalen Buchstabensequenzen zu unterscheiden. Anschließend werden für eine feste Anzahl N die N-Gramme gebildet, wobei das gesamte Wort als zusätzliches N-Gramm in die Darstellung mit aufgenommen wird (siehe Beispiel 10).

Beispiel 10: *Gegeben sei das Wort 'Kuchen' und* $N = 3$ *. Damit ergeben sich die folgenden N-Gramme: [⟨Ku, Kuc, uch, che, hen, en⟩]. Zusätzlich wird das gesamte Wort selbst als N-Gramm hinzugefügt: ⟨Kuchen⟩*

Die Autoren wählen in der Regel N zwischen 3 und 6, da sich empirisch gezeigt hat, dass dieser Bereich gute Ergebnisse bei Wortähnlichkeits- und Analogieaufgaben liefert. Ein kleiner Wert wie N = 2 ist ungeeignet, da er keine vollständigen Suffixe erfassen kann. Ab N = 3 lassen sich bereits bekannte Präfixe und Suffixe abbilden, wodurch Gemeinsamkeiten zwischen Wörtern erkennbar werden. Größere Werte wie N = 5 oder N = 6 sind besonders für Sprachen mit langen Wortverkettungen wie dem Deutschen sinnvoll, da sie zusammengesetzte Wörter besser erfassen. Da die Anfragen in diesem Projekt ausschließlich aus Ereignissen fester Größe g bestehen, sollte N maximal g - 1 betragen, um sinnvolle Teilinformationen zu berücksichtigen. Gleichzeitig muss $N > 3$ sein, um den Kontext von mindestens zwei Attributen abzudecken. Entsprechend sollte N im Intervall (3, g - 1) gewählt werden.

Anfrageembeddings Mithilfe von FastText lassen sich Ereignis-Embeddings erstellen. In diesem Projekt sollen jedoch gesamte Anfragen als Vektoren repräsentiert werden. Dafür muss aus den Vektoren der einzelnen Ereignisse ein neuer Vektor konstruiert werden, der die Gesamtbedeutung der Anfrage erfasst. Dies geschieht durch den Prozess des Poolings, bei dem die individuellen Token-Embeddings der Ereignisse zu einem einzigen Anfrage-Embedding zusammengeführt werden. Dabei ist zu beachten, dass diese neuen Embeddings eine abstrakte Repräsentation darstellen, wodurch feine Details der kleineren Ebenen nicht vollständig abgebildet werden können. Eine häufig verwendete Methode zur Erzeugung solcher Embeddings ist das Mean Pooling. Hierbei wird über alle Ereignis-Embeddings iteriert und das arithmetische Mittel berechnet [21]. Diese Methode behandelt alle einfließenden Elemente gleich, ohne zusätzliche Gewichtung. Daher ist es entscheidend, nur die relevanten Elemente einzubeziehen. [25] Da bei einer Anfrage jedes Ereignis eine wesentliche Rolle spielt, ist dieser Punkt in diesem speziellen Kontext jedoch weniger kritisch. Zur Berechnung eines Anfrage-Embeddings für eine Anfrage s_1 wird zunächst ein Vokabular aller N-Gramme $n = \{n_1, \ldots, n_n\}$ der Größe $|n|$ erstellt. Zusätzlich wird die Menge der Embeddings $e(n)$ als $e = \{e(n_1), \ldots, e(n_n)\}$ mit der Größe $|n|$ erstellt. Anschließend wird s_1 in seine Menge an N-Grammen $n(s_1)$ zerlegt, sodass sich das Durchschnittsembedding durch $e(s_1) = \frac{1}{|n(s_1)|} \sum_{i \in n(s_1)} e(n_i)$ ergibt. Dieses Embedding dient anschließend als Punkt für die Clusterung.

4.4 Clusterung

In diesem Kapitel wird dargestellt, wie aus einzelnen Datenpunkten eine Clusterstruktur entsteht. Dazu wird zunächst erläutert, wie partitionierende Clustering-Verfahren grundsätzlich funktionieren. Daraufhin folgt die Vorstellung eines konkreten Algorithmus. Abschließend wird beschrieben, wie eine geeignete Anzahl an Clustern ermittelt werden kann.

Partitionierende Clusterverfahren beginnen mit einer initialen Gruppeneinteilung der Objekte und optimieren diese schrittweise durch Austauschverfahren, bis eine optimale Zuordnung erreicht ist. Objekte können dabei flexibel zwischen den Gruppen wechseln, was diese Methode anpassungsfähiger macht als andere Verfahren, bei denen Zusammenführungen oder Trennungen unumkehrbar sind. Aufgrund dieser Flexibilität kommen die partitionierenden Verfahren im Praxisteil zur Anwendung. Bei partitionierenden Clusterungen spielt die Wahl der Distanzmetrik eine entscheidende Rolle für das Ergebnis einer Clusterung. Die wohl bekannteste Distanzmetrik ist die euklidische Distanz. Sie basiert auf dem Satz des Pythagoras und misst die Luftlinie zwischen zwei Punkten. Diese Methode lässt sich auch auf mehrdimensionale Punkte anwenden. Die Formel lautet: $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ Hierbei bezeichnet n die Anzahl der Dimensionen, während x und y die Punkte im Vektorraum darstellen. Der Index i gibt den Wert in der jeweiligen Dimension an. Ein weiteres Distanzmaß ist die Manhattan-Distanz, auch City-Block-Distanz genannt. Sie berechnet den Abstand zwischen zwei Punkten mit der Einschränkung, dass Bewegungen nur entlang der Achsen erfolgen dürfen. Die Formel lautet: $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$ Die Variablen haben dieselbe Bedeutung wie bei der euklidischen Distanz, jedoch wird hier der absolute Betrag der Differenzen verwendet.
In dem Paper [28] werden die beiden Metriken zusätzlich mit der Minkowski-Distanz verglichen. Die Minkowski-Distanz ($d_p(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$)stellt eine Verallgemeinerung der anderen beiden Distanzen dar. Je nach Wahl des Parameters p kann sie unterschiedliche Metriken repräsentieren: Für $p = 1$ entspricht sie der Manhattan-Distanz, für $p = 2$ der euklidischen Distanz. Mithilfe der Auswertung des Papers, lässt sich sagen, dass die Manhatten-Distanz nicht im Praxisteil dieser Arbeit genutzt wird, da Abweichungen in der Clusterung sich meist in dieser Metrik unterscheiden. Weiterhin bestehen die zu clusternden Vektoren aus metrischen Werten. Laut Backhaus [2] ist die euklidische Distanz für diese Metrik gut geeignet. Daher wird im Praxisteil die euklidische Distanz verwendet.

Der **K-Means-Algorithmus** ist ein Vertreter der partitionierenden Verfahren zur Gruppierung von N Datenpunkten in k Cluster. Im Verlauf des Algorithmus wird die sogenannte „squared error-based“ Funktion E minimiert, um die bestmögliche Clusterzuordnung zu finden. In der Gleichung wird zunächst geschaut wie groß die Abweichungen innerhalb eines Cluster sind. Dazu wird der Abstand jedes Punktes v in einem Cluster C_i zum Mittelpunkt $\tilde{x_i}$ des Clusters berechnet. Je näher alle Punkte am Mittelpunkt des Clusters liegen, desto kleiner fällt das Ergebnis aus. Durch eine Quadrierung werden Punkte stärker gewichtet, die weit vom Mittelpunkt entfernt sind, da ihre Abweichungen überproportional

ansteigen. Die gesamte Funktion summiert schließlich alle Abweichungen innerhalb der Cluster auf, um die Qualität der gesamten Clusterung zu bewerten. Die Formel wird beschrieben mit: $E = \sum_{i=1}^{K} \sum_{v \in C_i} \left\| \tilde{X}_i - v \right\|^2$. Auch hier gilt: Je kleiner der Wert, desto besser [13]. Der K-means Algorithmus funktioniert dann wie folgt. Zunächst wird eine positive Zahl K festgelegt, die die Anzahl der Cluster definiert. Anschließend werden zufällig K Datenpunkte ausgewählt, die als anfängliche Zentroiden dienen [28]. Anschließend werden alle weiteren Datenpunkte den Clusterzentren zugeordnet. Hierfür wird der euklidische Abstand zwischen jedem Punkt, der einem Cluster zugeordnet werden soll, und jedem Clusterzentrum berechnet. Ein Datenpunkt wird dem Cluster zugeordnet, dessen Mittelpunkt ihm am nächsten liegt. Nach der Zuordnung werden die Clusterzentren aktualisiert, indem der Mittelwert der Punkte innerhalb eines Clusters neu berechnet wird. Dies führt zu neuen Abständen zwischen den Punkten und den neu definierten Mittelpunkten, was gegebenenfalls zu einer erneuten Umordnung der Datenpunkte führt. Dieser Vorgang wird so lange wiederholt, bis sich die squared-error-Funktion nur noch geringfügig ändert [13] oder die vordefinierte Anzahl an Iterationen erreicht ist. [2]

K-Wert Ein zentraler Aspekt des K-Means-Algorithmus ist die Wahl eines geeigneten K, also die Anzahl der Cluster. Da in dem Praxisteil kein Domänenwissen vorliegt und somit kein passender Wert für K vordefiniert werden kann, muss dieser ohne externe Informationen bestimmt werden. Stattdessen lassen sich interne Validierungsmaße nutzen, um die Clusterqualität zu bewerten. Dabei spielen zwei zentrale Aspekte eine Rolle. Der erste Aspekt ist der Cluster-Zusammenhalt (Kohäsion), dieser beschreibt, wie eng die Datenpunkte innerhalb eines Clusters beieinanderliegen. Eine hohe Kohäsion bedeutet, dass die Punkte eines Clusters nahe beieinander liegen und somit eine starke Gruppierung aufweisen. Der zweite Aspekt ist die Cluster-Trennung (Separation), diese misst, wie deutlich sich die Cluster voneinander unterscheiden. Eine hohe Separation deutet darauf hin, dass die Cluster weit auseinanderliegen und somit klar voneinander abgegrenzt sind. Die Kohäsion wird mit der 'Within-Cluster Sum of Squares' (WSS) berechnet. Diese Metrik summiert die quadrierten Abweichungen aller Punkte innerhalb eines Clusters von ihrem jeweiligen Clusterzentrum auf. Ein geringer Wert deutet auf eine hohe Kohäsion hin. Mathematisch wird die WSS durch die folgende Formel beschrieben: $WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$, wobei x die einzelnen Datenpunkte innerhalb des Clusters C_i sind und m_i das Zentrum dieses Clusters darstellt. [28]
Die Separation wird durch die 'Between-Cluster Sum of Squares' (BSS) gemessen. Diese Metrik beschreibt, wie stark sich die Clusterzentren vom Gesamtmittelpunkt aller Datenpunkte unterscheiden. Die Berechnung der BSS erfolgt durch die folgende Formel: $BSS = \sum_i |C_i|(m - m_i)^2$. Hierbei bezeichnet m den Gesamtmittelpunkt aller Datenpunkte, also den Mittelwert über alle Datenpunkte hinweg, unabhängig von ihrer Clusterzugehörigkeit. Das Zentrum des Clusters C_i wird durch m_i dargestellt, was dem Mittelwert aller Punkte innerhalb dieses Clusters entspricht. Die Größe $|C_i|$ gibt die Anzahl der Elemente im Cluster C_i an. Eine hohe BSS bedeutet, dass sich die Clusterzentren weit vom Gesamtmittelpunkt unterscheiden, was auf eine klare Trennung der Cluster hindeutet. Mithilfe der beiden internen Validierungsmaße lassen sich nun

die beiden folgenden Berechnungen durchführen: der Silhouetten-Koeffizient und die Elbow-Methode. Die Elbow-Methode ist hierbei eine gängige Technik zur Bestimmung der optimalen Clusteranzahl in einem K-Means-Algorithmus. Sie basiert auf der Analyse der Within-Cluster Sum of Squares. Die Elbow-Methode visualisiert die WSS für verschiedene Werte von K in einem Diagramm. Auf der x-Achse steht die Anzahl der Cluster, auf der y-Achse der WSS-Wert. Der optimale Wert für K wird dort vermutet, wo der Graph einen deutlichen Knick – also einen „Ellbogen“ – zeigt. Ab diesem Punkt bringt eine weitere Erhöhung der Clusteranzahl kaum noch eine signifikante Verbesserung der Clustering-Qualität. Allerdings hat die Methode eine Schwäche: Mit zunehmender Clusteranzahl sinkt die WSS zwangsläufig und nähert sich null, unabhängig davon, ob die Cluster tatsächlich sinnvoll sind. Daher kann man allein aus der WSS teilweise nur schwer die optimale Anzahl der Cluster ableiten.
Die Silhouette-Methode dient zur Bewertung der Clusterqualität, indem sie sowohl die Kohäsion innerhalb eines Clusters als auch die Trennung zwischen verschiedenen Clustern berücksichtigt. Der Silhouette-Koeffizient wird über das gesamte Cluster berechnet und liegt im Wertebereich von -1 bis 1. Er ergibt sich aus dem Verhältnis der Kohäsion (WSS) innerhalb eines Clusters zur Separation (BSS) zwischen den Clustern anhand der folgenden Formeln:
1. Wenn die Kohäsion kleiner als die Separation ist, gilt: $s = 1 - \frac{\text{WSS}}{\text{BSS}}$
2. Wenn die Kohäsion größer als die Separation ist, gilt: $s = \frac{\text{BSS}}{\text{WSS}} - 1$
Ein Silhouette-Wert nahe 1 deutet darauf hin, dass die Punkte der Cluster passend zu einem Cluster zugeordnet wurden. Um die optimale Anzahl der Cluster zu bestimmen, erstellt man ein Diagramm, in dem die X-Achse die Anzahl der Cluster K und die Y-Achse den Silhouette-Koeffizienten darstellt. Der höchste Wert im Diagramm gibt den besten K-Wert an, da er das optimale Verhältnis zwischen Kohäsion und Separation widerspiegelt. [28]
In der Quelle [28] werden die Elbow-Methode und die Silhouette-Methode gegenübergestellt. Oft führen beide zu ähnlichen Ergebnissen. Allerdings schlägt die Silhouette-Methode weniger K-Werte von 0 vor und ist daher robuster gegenüber schlecht definierten Clustern. Zudem erfolgt die Analyse in dieser Arbeit nicht visuell über Diagramme, weshalb im Praxisteil die Silhouette-Methode zur Bestimmung des optimalen K-Wertes genutzt wird.
Eine weitere Metrik zur Bewertung von Clusterergebnissen ist der Davies-Bouldin-Index. Dieser bewertet die Clusterung insbesondere im Hinblick auf die Kompaktheit innerhalb der Cluster und die Trennung zwischen verschiedenen Clustern. Zur Berechnung der Metrik wird zunächst für jedes Cluster C_i die durchschnittliche Streuung zum jeweiligen Mittelpunkt c_i bestimmt: $S_i = \frac{1}{|C_i|} \sum_{x \in C_i} \|x - c_i\|$. Der Abstand M_{ij} zwischen zwei Clustern i und j wird als euklidische Distanz zwischen ihren Mittelpunkten definiert: $M_{ij} = \|c_i - c_j\|$. Daraus ergibt sich die sogenannte Cluster-Ähnlichkeit zwischen zwei Clustern: $R_{ij} = \frac{S_i + S_j}{M_{ij}}$. Für jedes Cluster i wird die schlechteste Trennung zu einem anderen Cluster bestimmt, also der maximale Wert von R_{ij} über alle $j \neq i$. Der Davies-Bouldin-Index berechnet anschließend den Durchschnitt dieser Maximalwerte über alle k Cluster: $DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} R_{ij}$ Der Wert des Davies-Bouldin-Index ist stets größer oder gleich null. Je kleiner dieser Wert ausfällt, desto besser ist die Qualität der Clus-

terung. Die Cluster weisen dann eine geringe Streuung auf, sind somit kompakter, und gleichzeitig liegen die Cluster weiter voneinander entfernt. [8]

5 Implementation und Evaluation

5.1 Setup

Experimentelles Setup Für die Implementierung dieses Projekts wurde Python in der Version 3.12.4 verwendet. Dabei kamen mehrere essenzielle Bibliotheken zum Einsatz, darunter fasttext, UMAP, NumPy, pandas, scikit-learn sowie matplotlib. Das Projekt wurde auf einem Rechner mit dem Betriebssystem Windows 11 ausgeführt, ausgestattet mit einem Intel Core i5-1135G7 Prozessor der 11. Generation (2.40 GHz) und 8 GB RAM. Das vollständige Projekt ist unter [3] einsehbar. Die Projektstruktur ist wie folgt gegliedert:

- Im Ordner `data` befindet sich die CSV-Datei, in der sowohl die Anfragen als auch die hierarchische Struktur gespeichert sind.
- Im Ordner `preprocessing` sind die drei genutzten Vektorisierungsansätze implementiert.
- Der Ordner `clustering` enthält die Klassen zur Erstellung und Auswertung der Cluster.
- Die Ergebnisse der Clusteranalysen werden im Ordner `result` abgelegt.
- Der gesamte Ablauf des Projekts wird in der Datei `main.py` gesteuert.

Datensatz Für den praktischen Teil wird ein Datensatz von Google verwendet, der Traces aus Google-Clustern enthält [27]. Ein cluster wird hierbei als eine Menge von Maschinen beschrieben, welche über ein gemeinsames Netzwerk verbunden sind. Genauer wird sich hierbei ein 'Zelle' angeschaut, welche ebenfalls eine Menge von Maschinen, meist aus einem gemeinsamen Cluster sind und mithilfe eines Cluster-Managementsystems organisiert werden. Dieses System verteilt dabei die verschiedenen Aufgaben an die entsprechenden Maschinen. Die Traces stellen Informationen bereit, welche durch solches Management system entstehen und durch einzelnen Maschinen in den Zellen gewonnen werden. Die Ereignisse eines Traces bilden dabei fünf parameter ab. Zunächst gibt es die Job-ID, einen eindeutigen Bezeichner, der angibt, zu welchem Job die Aufgabe gehört. Das zweite Attribut ist der Aufgabenindex innerhalb des Jobs, welcher als ganze Zahl dargestellt wird. Das dritte Attribut gibt an, auf welcher Maschine der Job ausgeführt wird, wobei auch hier eindeutige Bezeichner verwendet werden. Die letzten beiden Attribute beziehen sich auf die Eigenschaften der Aufgabe. So wird im vierten Attribut der Status der Aufgabe gespeichert, wobei es insgesamt acht verschiedene Statuswerte gibt. Das fünfte Attribut gibt die Priorität an, mit der die Aufgabe ausgeführt wird. Dabei gilt: Je größer die Zahl, desto höher die Priorität der Aufgabe. Für die exakten Definitionen der Attribute und für weitere Informationen siehe [27].

5.2 Implementation

Struktur Das Projekt wurde unabhängig von der bisherigen Anfragegenerierung entwickelt, wodurch die neue Pipeline vollständig entkoppelt ist. Für eine erfolgreiche Nutzung ist lediglich die korrekte Übergabe an der Schnittstelle ausschlaggebend. Die Implementierung folgt einem klaren Ablauf (siehe Abbildung 8). Zunächst wird eine Schnittstelle definiert, über die alle für das Clustering relevanten Informationen bereitgestellt werden. Diese Daten werden anschließend ausgelesen und an die jeweiligen Vektorisierungsverfahren weitergegeben. Die Vektorisierung bereitet die Daten je nach Ansatz unterschiedlich auf. Der hierarchische Ansatz erfordert vorab den Aufbau einer Struktur, während die anderen Verfahren direkt mit den deskriptiven Anfragen arbeiten können. Alle Verfahren geben eine einheitliche Punktematrix aus. Diese konsistente Ausgabe ist notwendig, da sie eine einheitliche Weiterverarbeitung durch den K-Means-Algorithmus erlaubt, der auf dem euklidischen Abstand basiert. Nach dem Clustering werden die Ergebnisse analysiert, sowohl auf der Gesamtebene als auch innerhalb der einzelnen Cluster.

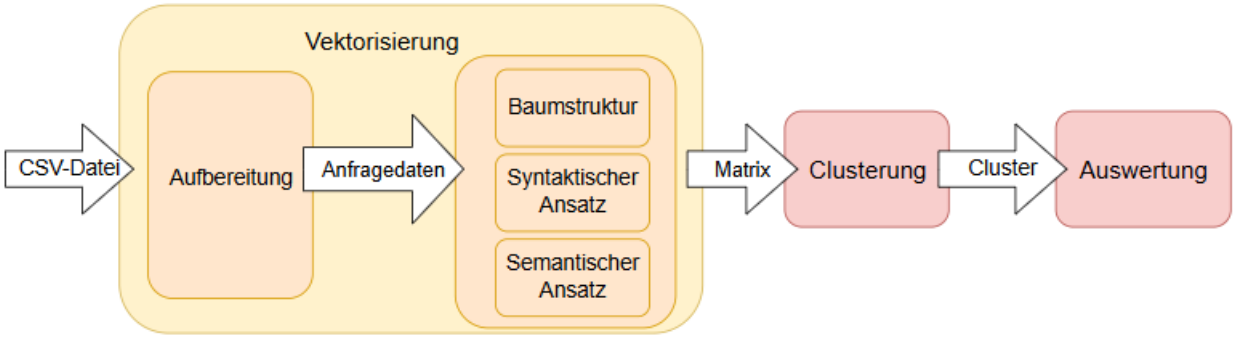

Abbildung 8: Pipeline des Praxisteil

Schnittstelle Damit die Clusterung unabhängig von der bisherigen Implementierung aus [15] funktioniert und nicht an den ursprünglichen Code zur Anfrageextraktion gebunden ist, musste eine Schnittstelle geschaffen werden, die die Anfragedaten in einer für die Clusterung nutzbaren Form bereitstellt. Zu diesem Zweck wurde der bestehende Code um die Ausgabe einer CSV-Datei erweitert. Diese CSV-Datei enthält alle relevanten Informationen zu den Anfragen, insbesondere auch deren hierarchische Struktur. Die Datei besteht daher aus zwei Spalten: In der ersten Spalte werden sämtliche Knoten der Struktur aufgeführt, während die zweite Spalte die jeweils zugehörigen Elternknoten angibt. Auf Basis der beiden Spalten kann die gesamte hierarchische Struktur zuverlässig rekonstruiert werden. Die zu clusternden deskriptiven Anfragen werden direkt aus der CSV-Datei extrahiert, indem jene Knoten identifiziert werden, die in der hierarchischen Struktur keine Eltern anderer Knoten sind. Diese bilden die Blätter der Struktur und sind somit die spezifischsten Anfragen.

Hierarchischer Ansatz Im hierarchischen Ansatz werden detaillierte Informationen über die Position der Anfragen in der Struktur benötigt. Alle Knoten werden dazu aus der CSV-Datei ausgelesen und ihren Elternknoten zugeordnet. Jeder Knoten wird als Objekt

gespeichert, das eine Liste seiner Elternknoten enthält. Diese Liste besteht aus Tupeln, die auf den übergeordneten Knoten und das zugehörige Kantengewicht verweisen. Die Kantengewichte werden anhand von Gewichtungsparametern berechnet, die die Relevanz struktureller Unterschiede zwischen Knoten und Elternknoten bestimmen, wie z. B. die Hinzufügung neuer Terminalzeichen, die Einführung von Variablen und die Veränderung der Länge. Diese Werte werden mit den Gewichtungsfaktoren verrechnet, um festzulegen, welche Merkmale bei der Clusterbildung stärker berücksichtigt werden. Nach der Erstellung der hierarchischen Struktur erfolgt die Vektorisierung. Zunächst wird eine Distanzmatrix erstellt, indem die Distanz zwischen allen Anfragen mit der Wu/Palmer-Distanz berechnet wird. Diese Distanzmatrix wird dann mithilfe des MDS-Algorithmus aus der Bibliothek 'scikit-learn.manifold' in Punkte eines euklidischen Raums umgewandelt. Dabei wird die Projektion auf zwei Dimensionen begrenzt und die Distanzmatrix als Eingabe genutzt. Die Ergebnisse werden anschließend weiterverarbeitet.

Syntaktischer Ansatz Zur Erstellung der Punkte im syntaktischen Ansatz werden zunächst die deskriptiven Anfragen extrahiert. Anschließend werden zwei Wörterbücher aufgebaut, zum einen für die Ereignisse und zum anderen für die Attribute. Zuerst wird das Ereignis-Wörterbuch erstellt. Jede Anfrage wird in seine Ereignisse zerlegt und anschließend werden daraus die normalisierten N-Gramme bis zu einer gewünschten Länge generiert. Dabei werden Variablen, die mit einem „$“ gekennzeichnet sind, durch durchnummerierte Platzhalter mit „%“ ersetzt. Wobei die gleichen Variablen auch die gleichen Nummern erhalten. Im nächsten Schritt wird das Attributwörterbuch erstellt, dazu werden die Anfragen in Ereignisse und diese in Attribute unterteilt. Nicht initialisierte Attribute werden durch einen Unterstrich markiert. Für jede Attribut-Dimension werden N-Gramme erstellt und im Attributwörterbuch abgelegt. Dabei wird berücksichtigt, dass Attribute nur dann als ähnlich gelten, wenn sie in derselben Position (Dimension) auftreten. Auch hier erfolgt eine Normalisierung der Variablen. Sobald beide Wörterbücher vorliegen, werden für jede Anfrage die Vektoren berechnet. Für Ereignisse werden entsprechende N-Gramme erzeugt, gezählt und mit dem Wörterbuch abgeglichen. Der Wert in der jeweiligen Dimension ergibt sich aus der Häufigkeit multipliziert mit dem festgelegten Gewicht. Das gleiche Verfahren gilt für die Attribute, wobei zusätzlich geprüft wird, ob die Position des Attributs mit der im Wörterbuch übereinstimmt. Zum Abschluss werden Ereignis- und Attributvektoren zusammengeführt. Die daraus entstehende Matrix dient als Grundlage für das anschließende Clustering.

Semantischer Ansatz Beim semantischen Ansatz wird die FastText-Bibliothek verwendet, die 2016 von Facebook AI Research veröffentlicht wurde. FastText-Modelle trainieren mit Textdateien, weshalb nach der Extraktion der deskriptiven Anfragen aus der CSV-Datei eine temporäre Textdatei erstellt wird. In dieser Datei werden die Anfragen zwischengespeichert, um sie für das Modell bereitzustellen. Da für die Daten keine externe Validierung vorliegt, muss das Modell die Muster und Strukturen in den Anfragen eigenständig erkennen und daraus lernen. Um dies zu ermöglichen, wird das Modell unüberwacht trainiert. Dazu wird die Funktion 'train_unsupervised' von FastText genutzt, wobei die Textdatei übergeben wird und das Modell auf den Skip-Gram-Algorithmus

eingestellt wird (siehe Kapitel 4.3). Die Embeddings der einzelnen Anfragen werden dann mit dem vortrainierten Modell berechnet, indem jede Anfrage in ihre Ereignisse aufgeteilt wird. Für jedes dieser Ereignisse wird dann das entsprechende Embedding ermittelt. Anschließend wird der Durchschnitt aller Ereignis-Embeddings einer Anfrage berechnet, um das gesamte Anfrage-Embedding zu erhalten. Dieser Prozess wird für alle Anfragen wiederholt, und die daraus resultierende Matrix wird für die Clusterung weitergegeben.

Clusterung Die Clusterung erhält von allen Ansätzen eine Punktematrix und hat die Aufgabe, die übergebenen Punkte in Cluster aufzuteilen. Dafür wird der KMeans-Algorithmus aus der Bibliothek 'scikit-learn.cluster' verwendet. Der Algorithmus wird mit der gewünschten Anzahl an Clustern initialisiert und mit einem 'random_state'-Seed versehen, um sicherzustellen, dass die Ergebnisse reproduzierbar bleiben. Da K-Means zufällig Startwerte wählt, ermöglicht der 'random_state'-Seed, die Zufallsgenerierung auf festgelegte Werte zu beschränken, sodass stets die gleichen Ergebnisse erzielt werden. Um die optimale Clusteranzahl K zu bestimmen, wird für einen definierten Bereich von K-Werten die Clusterbildung durchgeführt. Die resultierenden Cluster werden mithilfe des Silhouetten-Scores aus der Bibliothek 'scikit-learn.metrics' bewertet. Eine Abbildung zeigt die Silhouetten-Score-Werte für jedes K, sodass die Schwankungen visuell erfasst werden können. Die Clusteranzahl mit dem höchsten Silhouetten-Wert wird als optimal angesehen und zurückgegeben. Ist das optimale K bestimmt, wird mit diesem der KMeans-Algorithmus ein weiteres Mal ausgeführt, wobei der gleiche 'random_state'-Wert verwendet wird, um dasselbe Cluster zu erzeugen. Dieses Cluster wird dann für die weitergehende Auswertung übergeben.

Auswertung Bei der Auswertung werden sowohl die Gesamtstruktur der Cluster als auch deren innere Zusammensetzung analysiert. Zur Bewertung der Clusterqualität werden der Silhouetten-Score und der Davies-Bouldin-Index berechnet. Ergänzend werden die Punktverteilung und die Größenverhältnisse der Cluster grafisch dargestellt. Auf Mikroebene wird für jedes Cluster die Anzahl der enthaltenen Anfragen ermittelt und eine zentrale Anfrage als Repräsentant bestimmt. Weitere Kennzahlen wie Kohärenz, durchschnittliche Anfragelänge, Standardabweichung der Längen und bei hierarchischen Strukturen auch die mittlere Tiefe der Anfragen werden berechnet und gespeichert. Für die visuelle Analyse werden zusätzlich die Verteilung der Anfragelängen, die Häufigkeit einzelner Ereignisse sowie die Anzahl der Anfragen mit einem bestimmten Ereignis innerhalb eines Clusters dargestellt.

5.3 Variablenoptimierung

Dieser Abschnitt behandelt die Auswahl der einzelnen Variablen in den jeweiligen Implementierungen, um die bestmögliche Clusterung zu erzielen. Zur Bestimmung der Variablen für die Datensatzextraktion wird analysiert, wie viele Anfragen vorliegen und wie lang diese sind, um eine fundierte Entscheidung zu treffen. Für die Variablen der verschiedenen Ansätze werden zunächst Testläufe mit willkürlich initialisierten Werten durchgeführt, wobei die einzigen Einschränkungen bereits im Theorieteil 4 definiert wurden. Anschlie-

ßend wird die Qualität der gebildeten Cluster anhand des Silhouetten-Scores und des Davies-Bouldin-Indexes (DBI) bewertet. Schließlich werden die Variablen ausgewählt, bei denen die Cluster am deutlichsten voneinander abgegrenzt sind, um präzisere Aussagen über die Clusterarten und deren Ergebnisse treffen zu können.

Datensatz Um zunächst ein Sample aus den Rohdaten zu extrahieren, müssen die Anzahl und Länge der Traces bestimmt werden. Bei der Anfragegenerierung mit `bu_discovery_multidim()` kann zusätzlich der Support-Threshold angepasst werden. Ein Wert von 1.0 bedeutet, dass eine Anfrage alle Traces abdecken muss, wodurch möglichst allgemeine Queries entstehen, die Aussagen über das gesamte Sample erlauben. Die Anzahl und Länge der Traces lassen sich ebenfalls variieren. Zur Analyse ihrer Auswirkungen wurde eine Datei erstellt, in der unter anderem die Anzahl der generierten deskriptiven Queries gespeichert wird. Ebenfalls erfasst werden die Standardabweichung und die durchschnittliche Länge der Anfragen. Eine hohe Standardabweichung deutet auf eine größere Varianz in der Länge hin, was die Analyse vielseitiger macht. Längere Anfragen sind dabei besonders nützlich, da sie mehr Ereignisse enthalten und so detailliertere Aussagen innerhalb der Cluster ermöglichen. Abschließend wird die Ausführungszeit gemessen, um den Rechenaufwand einzelner Parameterkonfigurationen nachvollziehen zu können. Die Ergebnisse der Tests mit den verschiedenen Eingabeparametern sind im Projektverzeichnis unter `variable_tuning/dataset` abgelegt. Bei konstanter Tracelänge nimmt mit steigender Trace-Anzahl sowohl die Anzahl der gefundenen deskriptiven Anfragen als auch deren durchschnittliche Länge ab. Dies lässt sich damit erklären, dass bei einem Threshold von 1.0 jede Anfrage alle Traces abdecken muss und deshalb allgemeiner formuliert wird. Dadurch werden weniger spezifische Ereignisse verwendet und die Anfragen enthalten insgesamt weniger Einschränkungen. Auch die Ausführungszeit zeigt ein interessantes Verhalten. Sie ist bei sehr wenigen und bei sehr vielen Traces erhöht, während sie im mittleren Bereich deutlich sinkt. Dieses Muster konnte bei einer Tracelängen von 10 und 11 beobachtet werden, weitere Längen wurden nicht untersucht. Zusammengefasst führen mehr Traces zu weniger und kürzeren Anfragen. Für die Nutzung des Algortihmuses erscheint ein Wert im Bereich zwischen 50 und 500 Traces sinnvoll, da hier die Rechenzeit spürbar geringer ausfällt. Mit zunehmender Tracelänge verlängern sich die generierten Anfragen, da mehr Ereignisse in den Traces berücksichtigt werden müssen. Gleichzeitig steigt die Anzahl der Anfragen, da eine größere Vielfalt an Kombinationen möglich ist. Die Standardabweichung zeigt hierbei kein einheitliches Verhalten. Insgesamt nimmt die Ausführungszeit des Algorithmus mit längeren Traces ebenfalls zu. Für das finale Datenset wurde eine Tracelänge von 13 gewählt, da diese Länge die ausführlichsten Anfragen ermöglicht und gleichzeitig eine stabile Laufzeit zwischen zwei Minuten dreißig und acht Minuten dreißig gewährleistet, abhängig von der Anzahl der Traces. Diese wurde auf 400 festgelegt, wodurch 520 Anfragen erzeugt werden, die sich gut für eine anschließende Clusterung eignen. Eine geringere Trace-Anzahl würde die Laufzeit erhöhen und die Anzahl der Anfragen unnötig vergrößern. Eine höhere Anzahl würde die Laufzeit ebenfalls verlängern, dabei jedoch die Anfragen verkürzen und somit die inhaltliche Tiefe verringern. Aus diesen Gründen basiert das Datenset auf 400 Traces mit jeweils 13 Ereignissen.

Hierarchische Struktur Beim hierarchischen Strukturansatz kann man die später gebildeten Cluster dadurch beeinflussen, dass man die Gewichte zwischen zwei Knoten verändert. Dabei lassen sich drei Arten von Gewichten anpassen: das Termgewicht, das Variablengewicht und das Ereignisgewicht (siehe Kapitel 4.1). Um die geeigneten Gewichte für die Clusterung zu bestimmen, wurden Cluster mit unterschiedlichen Parameterwerten erstellt, ausgewertet und im Ordner `variable_tuning/structure` abgelegt. So werden neben den einzelnen Gewichten die optimale Clusteranzahl, der Silhouetten-Score, der DBI und die generellen Anzahlen der Cluster gespeichert. Zudem wird das Verhältnis zwischen dem Silhouetten-Score und dem Davies-Bouldin-Index berechnet. Da ein hoher Silhouetten-Score und ein niedriger DBI von Vorteil wären, wird das Verhältnis mit $\frac{\text{Silhouetten-Score}}{\text{Davies-Bouldin-Index}}$ gebildet. Für das Term-Gewicht wurden die Werte [0.1, 0.5, 1.0, 2.5, 5.0], für das Variablen-Gewicht die Werte [0.05, 0.2, 0.5, 1.0, 2.5, 5.0] und für das Ereignis-Gewicht die Werte [0.01, 0.1, 0.3, 1.0, 2.0] getestet. Dabei wurde jede Kombination nur dann berücksichtigt, wenn die Bedingung term_weight $\geq$ variablen_weight $\geq$ length_weight erfüllt war (siehe Kapitel 4.1). Somit wurden 52 Kombinationen getestet. Die optimale Clusteranzahl liegt nach den besten Silhouetten-Scores zwischen 3 und 7. Dabei wurde einmal die Clusteranzahl 3, 36-mal die Clusteranzahl 6 und 15-mal die Clusteranzahl 7 gewählt. Die Silhouetten-Scores liegen dabei zwischen 0.363 und 0.375, während der Davies-Bouldin-Index zwischen 0.74 und 0.88 schwankt. Die geringen Schwankungen in den Ergebnissen zeigen, dass die Veränderung der Variablen nur einen kleinen Einfluss auf die Bildung der besten Cluster haben. So lassen sich unabhängig von den gewählten Parametern ähnlich gut getrennte Cluster generieren. Die Wahl der Variablen hat offenbar einen Einfluss auf die Clusterbildung bei größeren K-Werten. Lässt man die Längenunterschiede weitgehend unberücksichtigt und setzt das Gewicht für die Länge auf 0.01, während die anderen Gewichte mit Werten von 5.0 eine hohe Bedeutung erhalten, zeigt sich in Abbildung 9 ein markanter Verlauf. Auf der X-Achse sind die jeweiligen Clusteranzahlen K dargestellt, auf der Y-Achse die zugehörigen Silhouetten-Werte. Es lässt sich erkennen, dass der Silhouetten-Score für $K > 8$ zunächst abnimmt und erst bei einem Minimum um $K \approx 60$ wieder ansteigt. Im Bereich zwischen $K = 150$ und $K = 250$ schwankt der Silhouetten-Wert bei etwa 0.35, bevor er erneut absinkt. Wird hingegen das Gewicht für die Länge auf 4.0 erhöht, zeigt Abbildung 10 einen ähnlichen Verlauf, allerdings steigen die Werte für $K > 150$ nicht weiter an, sondern fallen ab und pendeln lediglich bei etwa 0.32. Dies legt nahe, dass die gebildeten Cluster bei höherer Gewichtung der Länge weniger klar voneinander getrennt sind als bei niedriger Gewichtung. Ein möglicher Erklärungsansatz dafür ist, dass zu Beginn vor allem die hierarchische Struktur der Anfragen eine dominante Rolle spielt. Es könnten zwischen $K = 6$ und $K = 7$ besonders stark zusammenhängende Unterstrukturen existieren, wodurch jeweils die Blätter einer solchen Unterstruktur gemeinsam einem Cluster zugeordnet werden. In diesem Bereich wirken sich die Gewichtungen kaum auf die Clusterbildung aus, was sich erst bei größeren K-Werten ändert. Da jedoch der Schwerpunkt dieser Arbeit auf den Clusterungsmöglichkeiten liegt, wird auf eine tiefergehende Analyse dieses Phänomens verzichtet, um die Komplexität nicht unnötig zu erhöhen.
Im nächsten Schritt wird der Fokus wieder auf die 52 besten Clusterungen gelegt. Da

ein K-Wert von 3 nur geringe Aussagekraft besitzt und zudem lediglich in einer der 52 getesteten Kombinationen auftrat, wird diese Clusteranzahl aus der weiteren Analyse ausgeschlossen. Besonders auffällig ist, dass der K-Wert 6 in 36 Fällen als optimale Clusteranzahl ermittelt wurde. Dies weist auf eine hohe Relevanz dieser Anzahl innerhalb der hierarchischen Struktur hin, weshalb eine Clusterung gewählt wird, die K = 6 empfiehlt. Da die Unterschiede zwischen den verschiedenen Kombinationen in Bezug auf den Silhouetten-Score und den Davies-Bouldin-Index nur gering ausfallen, erfolgt die Auswahl der endgültigen Parameterkombination auf Basis des besten Verhältnisses dieser beiden Metriken. Die beste Kombination ergibt sich mit einem Verhältnis von 0.495 aus den Parametern term_weight = 5.0, variablen_weight = 0.5 und length_weight = 0.01. Diese Kombination liefert einen Silhouetten-Score von 0.375 sowie einen Davies-Bouldin-Index von 0.757. Für den Praxisteil des hierarchischen Ansatzes werden die Variablen daher wie folgt definiert: K = 6, term_weight = 5.0, variablen_weight = 0.5 und length_weight = 0.01.

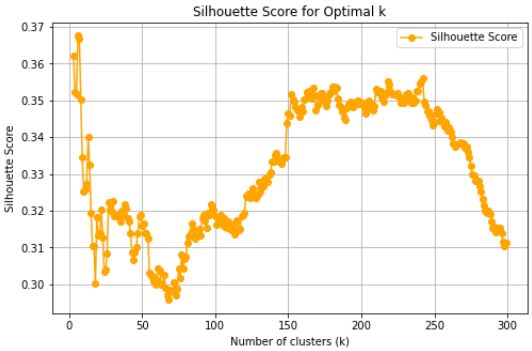

Abbildung 9: Silhouetten Scorewerte für Längengewicht = 0.1

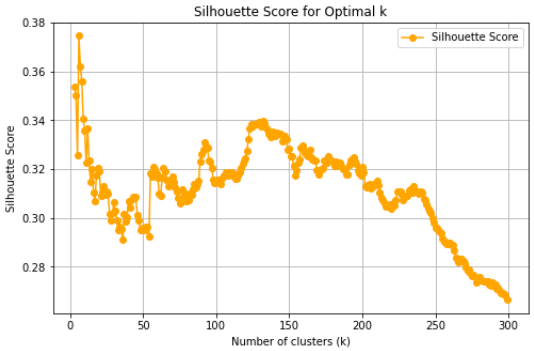

Abbildung 10: Silhouetten Scorewerte für Längengewicht = 4.0

Syntaktischer Ansatz Beim syntaktischen Ansatz können insgesamt sechs Variablen angepasst werden, was diesen Ansatz zu dem flexibelsten Ansatz macht. Zunächst lassen sich die Variablen n_ev und n_att bestimmen, die angeben, bis zu welcher Länge N-Gramme aus Ereignissen bzw. Attributen gebildet werden sollen. Die weiteren vier Variablen steuern die Gewichte und ermöglichen es, die Bedeutung einzelner Eigenschaften gezielt anzupassen. Dazu gehören weight_event, weight_event_ngram, weight_att_individual und weight_att_ngram. Wie bereits aus den Namen ersichtlich ist, lassen sich damit sowohl das Verhältnis zwischen Ereignissen und Attributen als auch die Gewichtung einzelner Elemente im Vergleich zu N-Grammen beeinflussen.
Zur Bestimmung einer geeigneten N-Gramm-Länge wurden verschiedene Werte bis zu einer definierten Obergrenze getestet. Entgegen der Erwartung führte die Verwendung längerer N-Gramme nicht zu besseren Ergebnissen, sondern verschlechterte die Clusterqualität. Zur detaillierten Analyse wurden in der Datei `variable_tuning/syntax/event_ngram_length_test` unterschiedliche Längen untersucht, wobei ausschließlich das Gewicht für Event-N-Gramme auf 1 gesetzt wurde. Alle anderen Gewichte blieben auf 0, sodass die Clusterbildung ausschließlich auf Ereignis-N-Grammen basierte. Zur Be-

wertung wurden der durchschnittliche Silhouetten-Score und der Davies-Bouldin-Index herangezogen. Da deren Mediane kaum von den Mittelwerten abwichen, wurden sie nicht weiter berücksichtigt. Die Ergebnisse zeigen klar, dass längere N-Gramme keinen Qualitätsgewinn bringen. Bei einer N-Gramm-Länge von 2 lag der Silhouetten-Score bei 0.2, was auf eine noch akzeptable Trennung der Cluster hinweist. Bereits bei einer Länge von 3 sank dieser auf 0.12, was deutlich überlappende Cluster signalisiert. Daher wird im Praxisteil eine N-Gramm-Länge von 2 für Ereignisse verwendet.
Bei den Tests für die optimale Attribut-N-Gramm-Länge zeigen sich höhere durchschnittliche Werte, was auf eine bessere Trennbarkeit der Daten hindeutet. Die Ergebnisse sind in der Datei `variable_tuning/syntax/attribut_ngram_length_test` gespeichert. Aber auch hier zeigt sich ein absteigender Trend, wobei eine N-Gramm-Länge von 2 mit einem durchschnittlichen Silhouetten-Score von 0.31 den besten Wert liefert. Die nächstbeste Länge von 3 erzielt nur noch 0.17 und wird daher als unzureichend bewertet. Aus diesem Grund wird auch bei den Attributen eine N-Gramm-Länge von 2 bevorzugt. Die im Theorieteil vorgeschlagene Gewichtungshierarchie (individuelle Attribute $\leq$ N-Gramm-Attribute $\leq$ individuelle Ereignisse $\leq$ Ereignis-N-Gramme) kann aufgrund der schlechten Ergebnisse für beide N-Gramme nicht übernommen werden. Ein zu hohes Gewicht dieser N-Gramme verschlechtert die Clusterung, da sich das Modell zu stark an ihrer Struktur orientiert. Daher werden kleinere Gewichtungen für N-Gramme getestet, während individuelle Attribute und Ereignisse mit höheren Werten geprüft werden. Es ist jedoch anzumerken, dass die schwachen Ergebnisse für N-Gramme stark vom Datensatz abhängen. Bei anderen Datensätzen könnten längere N-Gramme durchaus eine präzisere Differenzierung ermöglichen. Nachdem die Länge der N-Gramme festgelegt wurde, geht es nun um die Zuweisung von Werten für die Gewichte. Insgesamt wurden 162 Kombinationen getestet, wobei für die N-Gramme, wie bereits erwähnt, die kleinsten Werte verwendet wurden. So erhielt das Gewicht der Ereignis-N-Gramme die Werte [0.1, 0.25, 0.45], die Attribut-N-Gramme wurden mit den Werten [0.15, 0.25, 0.4] getestet. Schließlich wurden die Gewichte für individuelle Attribute und Ereignisse auf [0.2, 0.25, 0.35, 0.4, 0.5, 0.9] gesetzt. Um die Ergebnisse für die passende Variablenwahl einzugrenzen, wurden für die Ergebnisse zusätzliche Bedingungen festgelegt. So muss der K-Wert im Bereich zwischen 4 und 100 liegen, um eine sinnvolle Interpretation der Ergebnisse zu ermöglichen. Zudem sollte das Gewicht für die individuellen Ereignisse größer sein als das Gewicht für die Attribute, da Ereignisse mehrere gleiche Attribute vereinen. Durch die Festlegung eines erforderlichen Verhältnisses zwischen dem Davies-Bouldin-Index und dem Silhouette-Score von mehr als 0,5 kann sichergestellt werden, dass die Punkte in die richtigen Cluster verteilt wurden und die Clustertrennung ebenfalls gut ausgeprägt ist.
Nach der Eingrenzung verbleiben neun Clusterergebnisse, die in Tabelle 1) dargestellt sind. Es ist zu erkennen, dass alle aufgeführten Clusterungen gute Ergebnisse liefern. Die schlechteste Clusterung hat ein Verhältnis von 0.5, während die beste ein Verhältnis von 0.9 erreicht. Zudem fällt auf, dass keines der Cluster die höchsten N-Gramm-Gewichte von 0.4 bzw. 0.45 enthält, was darauf hindeutet, dass die N-Gramme in dieser Clusterung keine wesentliche Rolle bei der Verbesserung der Clustertrennung spielen. Unter den getesteten Clusterungen sticht eine Kombination besonders hervor. Mit einem Verhältnis von etwa 0.9, einem Silhouetten-Score von 0.62 und einem Davies-Bouldin-Index (DBI) von 0.69

weist eine Clusterung die besten Ergebnisse hinsichtlich der inneren Clusterqualität auf. Diese Clusterung empfiehlt einen K-Wert von 54. Zusammenfassend ergeben sich für den praktischen Ansatz folgende Parameter: Die N-Gramm-Längen für Attribute und Ereignisse werden auf 2 festgelegt. Der K-Wert für den K-Means-Algorithmus wird auf 54 gesetzt. Das Gewicht für einzelne Ereignisse beträgt 0.9, das Gewicht für Ereignis-N-Gramme 0.1. Für einzelne Attribute wird ein Gewicht von 0.5 und für Attribut-N-Gramme ein Gewicht von 0.15 verwendet.

K	**Weight Event**	**Weight Event Ngram**	**Weight Attribut**	**Weight Attribut Ngram**	**Silhouette Score**	**Davies Bouldin Index**	**Verhältnis**
60	0.5	0.10	0.40	0.15	0.488	0.759	0.644
54	0.4	0.10	0.35	0.15	0.455	0.902	0.504
69	0.9	0.10	0.25	0.15	0.548	0.656	0.835
79	0.9	0.10	0.25	0.25	0.441	0.773	0.571
54	0.9	0.10	0.35	0.15	0.565	0.729	0.775
62	0.9	0.10	0.35	0.25	0.481	0.785	0.613
54	0.9	0.10	0.50	0.15	0.621	0.691	0.899
68	0.9	0.10	0.50	0.25	0.506	0.786	0.644
55	0.9	0.25	0.50	0.15	0.469	0.881	0.533

Tabelle 1: Clusterergebnisse nach Filterung

Semantischer Ansatz Bei der Erstellung der Embeddings für den semantischen Ansatz können Parameter wie minCount, minn und maxn angepasst werden. Der Parameter minCount legt fest, wie oft ein Wort im Trainingskorpus vorkommen muss, um berücksichtigt zu werden. So werden seltenere Wörter ausgeschlossen. In diesem Fall betrifft das Ereignisse, die nur wenige Male in den Anfragen auftreten. Da der Datensatz nur 520 Anfragen enthält, können häufiger seltene Ereignisse entstehen, etwa durch unterschiedliche Variablen bei sonst identischem Aufbau. Dennoch lassen sich auch für solche Ereignisse verlässliche Embeddings erzeugen, da das begrenzte Alphabet die Wiederverwendung von N-Grammen in anderen Anfragen begünstigt. So können selbst seltene Ereignisse indirekt mittrainiert werden. Daher wird minCount auf 1 gesetzt, damit alle Ereignisse im Training berücksichtigt werden. Die Parameter minn und maxn legen die Mindest- und Maximallänge der im Training verwendeten N-Gramme fest. Ist ein Wort kürzer als minn, wird es ohne N-Gramm ins Training übernommen. Bei längeren Wörtern werden N-Gramme nur bis zur definierten Maximallänge gebildet.
Ein Beispiel mit zwei willkürlich gewählten Gewichtungen (siehe Abbildung 11) zeigt, dass der Silhouetten-Score mit steigender Clusteranzahl zunächst zunimmt, jedoch bei $K \approx 280$ stark abfällt. Der höchste Wert vor dem Einbruch wird als optimales K identifiziert. Bei genauerer Betrachtung zeigt sich jedoch eine Schwäche der Anfrageembeddings. Da jedes Embedding aus dem Durchschnitt aller enthaltenen Wörter berechnet wird, gehen Details verloren. Anfragen, die dieselben Ereignisse in unterschiedlicher Reihenfolge enthalten, weisen die größte Ähnlichkeit auf. Das führt dazu, dass Anfragen lediglich auf Basis

gemeinsamer Ereignisse gruppiert werden, vergleichbar mit einem Bag-of-Words-Modell. Ein hoher Silhouetten-Score entsteht somit vor allem dann, wenn diese oberflächlichen Übereinstimmungen maximiert werden. Da dies dem Ziel einer semantisch fundierten Gruppierung widerspricht, sollte stattdessen ein kleinerer K gewählt werden, um auch tiefergehende inhaltliche Gemeinsamkeiten zu erfassen.
Um eine gute Wahl für die Parameter minn und maxn zu treffen, wurde der K-Wert-Raum von K=40 bis K=60 genauer betrachtet. In diesem Bereich liegen die Silhouetten-Scores meist über 0.30, was ähnlich hohe Werte wie bei dem hierarchischen Strukturansatz liefert. Zudem ist die Anzahl der Cluster klein genug, damit verschiedene Anfragen, die nicht vollständig übereinstimmen, zumindest denselben Cluster zugeordnet werden. Dies ermöglicht, eine semantische Aussagekraft über die Anfragen zu erhalten. Für diese Untersuchung wurden die Daten mit beiden Parametern, minn und maxn, für Werte zwischen 2 und 9 getestet und in der Datei `variable_tuning/semantik/embedding_var_test_avg_40_60` gespeichert. Auffällig ist, dass Clusterungen mit einem minn-Wert von 3 im Durchschnitt die besten Silhouetten-Scores aufweisen, was darauf hindeutet, dass die Cluster die beste Trennung aufweisen. Zudem liefert der Davies-Bouldin-Index mit etwa 1.14 ebenfalls den besten Wert bei minn = 3. Bei einer Gruppierung der Ergebnisse nach den maxn-Werten, also der maximalen Länge der N-Gramme, zeigt sich, dass die besten Silhouetten-Scores bei maxn = 5 und maxn = 6 erzielt werden. Der Davies-Bouldin-Index hingegen ist bei maxn = 1 und maxn = 4 am besten, gefolgt von den beiden zuvor genannten Werten. Dies zeigt, dass der geeignete Wert für maxn nicht eindeutig bestimmbar ist. In Kombination mit minn = 3 ergibt sich jedoch, dass die Kombination minn = 3 und maxn = 4 den besten Silhouetten-Score von 0.35 sowie einen Davies-Bouldin-Index von etwa 1.12 liefert. Daher werden für die weiteren Experimente die Parameter minn = 3 und maxn = 4 gewählt, da sie in diesem Bereich die durchschnittlich besten Ergebnisse zeigen. Für diese Kombination muss nun die optimale Clusteranzahl gefunden werden. Zwar könnte man nach der Clusterqualität das höchste K im fokussierten Intervall von 40 bis 60 wählen, was in diesem Fall K = 60 wäre, jedoch gilt auch, dass je weniger Cluster existieren, desto besser können diese analysiert werden. Daher wird für das Intervall 40-60 die prozentuale Änderungsrate zwischen einem K und K-1 betrachtet. Zusätzlich wird auch die Änderungsrate des Verhältnisses zwischen Silhouetten-Score und Davies-Bouldin-Index einbezogen. In der Grafik 12 kann man auf den X-Achsen die Clusteranzahlen zwischen 40 und 60 sehen, die Y-Achsen enthält die Werte der Metriken. Von links an erst der Silhouetten Score dann der DBI und zuletzt das Verhältnis der beiden. Während der Vertikale Strich in jeder Abbildung für den Punkt mit der größten Änderungsrate steht. Dabei ergibt sich, dass die größte Änderung des Silhouetten-Scores bei K = 48 mit einer Änderungsrate von 6,2 Prozent auftritt, während die größte Änderungsrate des Davies-Bouldin-Index bei K = 45 mit 8,1 Prozent zu finden ist. Das Verhältnis der beiden Metriken zeigt ebenfalls die größte Änderung bei K = 48, weshalb dieser Wert für das weitere Vorgehen gewählt wird. Daraus ergibt sich die Wahl der folgenden Variablen: minCount = 1, minn = 3, maxn = 4 und K = 48.

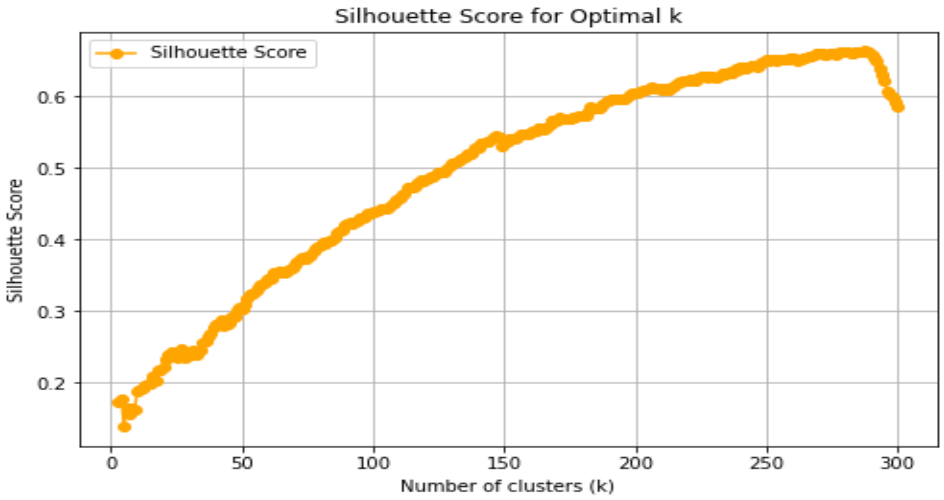

Abbildung 11: K-Werte für nmin=2 und nmax=3

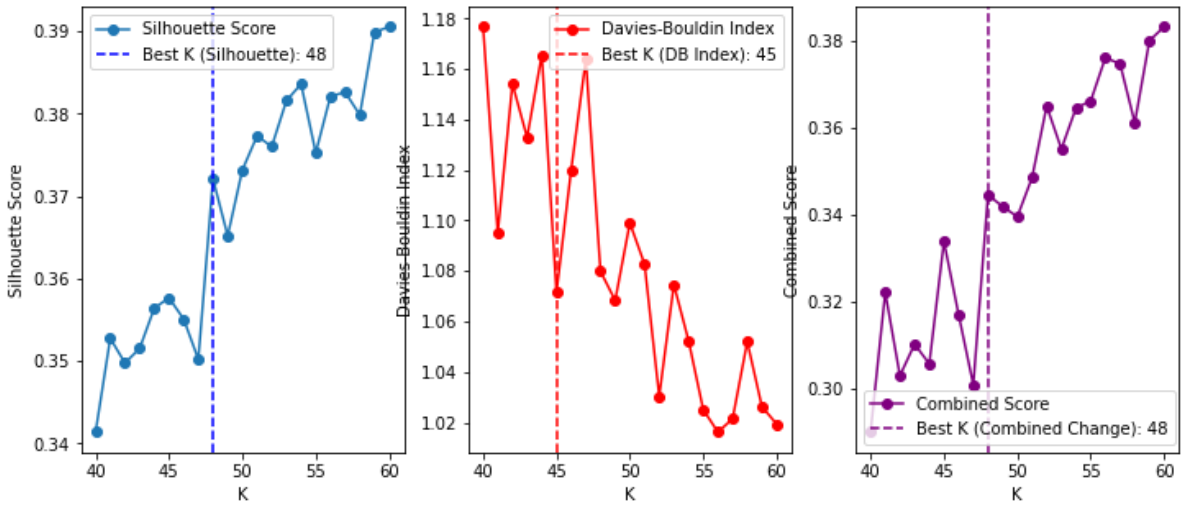

Abbildung 12: Evaluationsmetriken für K im Bereich von 40 bis 60

5.4 Ergebnisse

In diesem Abschnitt erfolgt die Bildung und anschließende Auswertung der Cluster auf Grundlage der in Kapitel 5.3 definierten Variablen. Zunächst wird jedes Cluster als Ganzes betrachtet, bevor im weiteren Verlauf eine detaillierte Analyse einzelner Cluster erfolgt. Die detaillierten Ergebnisse sind in dem Projekt [3] in dem Ordner `result` gespeichert.

Hierarchische Struktur Bei dem hierarchischen Strukturansatz werden die 520 Anfragen in sechs Cluster unterteilt. Im Mittel enthält jedes Cluster 86,67 Anfragen, wobei die Clustergrößen zwischen 82 und 90 Anfragen liegen. Die Clusterung erzielt einen Silhouettenwert von 0,375. Da dieser Wert größer als null ist, wurden die meisten Anfragen

sinnvoll einem Cluster zugeordnet. Der Wert bleibt jedoch unter 0,5, was darauf hindeutet, dass einige Datenpunkte auch plausibel anderen Clustern zugeordnet werden könnten. Darüber hinaus ergibt sich ein Davies-Bouldin-Index von 0,757. Ein Wert unter 1,0 zeigt, dass die Cluster im Durchschnitt gut voneinander getrennt sind. Die durchschnittliche Streuung innerhalb der Cluster ist somit geringer als die Distanz zwischen den jeweiligen Clustermittelpunkten, wenn gleich einzelne Ausnahmen möglich sind. In Abbildung 13 ist die Verteilung der Anfragen nach der Clusterung dargestellt. Die sechs Cluster lassen sich visuell voneinander unterscheiden, gleichzeitig zeigt sich jedoch, dass viele Punkte in unmittelbarer Nähe zu den Grenzen benachbarter Gruppen liegen. Auf diese Weise wird auch das Ergebnis der zuvor berechneten Metriken nachvollziehbar. Es lassen sich zwar grundlegende Strukturen erkennen, doch die Trennung zwischen den Clustern ist insgesamt nur schwach ausgeprägt. Die Visualisierung des Baums brachte keine weiteren Erkenntnisse, da sie aufgrund der vielen Knoten und Kanten zu unübersichtlich war und deshalb nicht weiter einbezogen wurde. Ein detaillierter Blick auf die einzelnen Cluster zeigt, dass diese hinsichtlich der durchschnittlichen Anzahl an Ereignissen sehr ähnlich sind. Die Mittelwerte bewegen sich in einem engen Bereich zwischen 5,74 und 5,78 Ereignissen pro Anfrage. Deutlichere Unterschiede zeigen sich jedoch bei der Streuung. Während zwei Cluster eine vergleichsweise geringe Standardabweichung von etwa 0,83 aufweisen, liegt sie in anderen Clustern bei bis zu 1,01. Ein Ausreißer zeigt sich in der Kompaktheit der Cluster. Während alle Cluster einen durchschnittlichen Abstand zum jeweiligen Mittelpunkt von mehr als 0,11 aufweisen, liegt dieser Wert für das als Cluster 0 definierte Zentrum in der Punkteverteilung bei unter 0,1. Dies lässt darauf schließen, dass die Punkte dieses Clusters eine geringere räumliche Ausdehnung aufweisen. Im Gegensatz dazu besitzen die Randcluster eine größere Streuung, wodurch sich der Abstand zum jeweiligen Mittelpunkt erhöht. Ein ähnliches Muster zeigt sich bei der durchschnittlichen Tiefe in der hierarchischen Struktur. Cluster 0, das sich zentral in der Punktverteilung befindet, weist eine durchschnittliche Tiefe von 3,85 auf. Die übrigen Cluster liegen mit Werten zwischen 4,26 und 4,34 deutlich darüber. Dies deutet darauf hin, dass Anfragen im Zentrum der Punktverteilung tendenziell höher in der hierarchischen Struktur angesiedelt sind. Zudem zeigt sich, dass sich neben dem Cluster mit den höherliegenden Anfragen fünf weitere Unterstrukturen ausbilden, die den übrigen Clustern entsprechen.
Im Folgenden werden zwei Cluster im Detail betrachtet. Zum einen Cluster 0, der zentrale Cluster in der Punkteverteilung, und zum anderen exemplarisch einer der fünf Randcluster. Cluster 0 weist, wie bereits beschrieben, sowohl die geringste durchschnittliche Tiefe in der hierarchischen Struktur als auch die höchste Kompaktheit auf. Der Repräsentant dieses Clusters, also die Anfrage, die dem Clustermittelpunkt am nächsten liegt, lautet: ‘$x0;;;$x1; ;;$x2;; $x0;;;$x1; ;;$x2;;‘. Die Anfragen innerhalb dieses Clusters umfassen zwischen drei und neun Ereignisse. Über ein drittel der Anfragen weisen dabei eine Länge von sechs Ereignissen auf (siehe Abbildung 14). In Abbildung 15 ist ersichtlich, wie oft ein Ereignis in einer Anfrage aufgetreten ist. In mehr als 35 Anfragen sind die Ereignisse ‘;;$x0;;‘ und ‘;;;$x0;‘ aufgetreten. Die vier weiteren häufigsten Ereignisse, die in mehr als einem Viertel der Anfragen vorkommen, haben die gleiche Form, jedoch mit einem erhöhten Variablenwert. Cluster 0 umfasst somit Anfragen, bei denen die Ereignisse häufig den gleichen Status teilen oder auf derselben Maschine ausgeführt werden. In über

20 Anfragen wurde zudem die Maschinen-ID ‘;;4;;‘ explizit angegeben, auf der ein Ereignis ausgeführt wird. Ein weiteres Cluster, das analysiert werden soll, ist Cluster 2. Cluster 2 enthält mit 90 Anfragen die meisten Anfragen und weist mit einer durchschnittlichen Länge von 5,88 auch die längsten Anfragen auf. Die Längen variieren zwischen 4 und 8. Die häufigsten Ereignislängen sind 5 und 7, wobei jede dieser Längen in mehr als 25 Anfragen vorkommt. Weniger als 5 Anfragen haben jeweils eine Länge von 4 oder 8. In der hierarchischen Struktur befinden sich die Anfragen bei einer durchschnittlichen Tiefe von 4,32, was sie im Durchschnitt zum zweit-tiefsten Cluster macht. Die Ereignisverteilung über die Anfragen ist in Abbildung 16 dargestellt. Auffällig ist, dass die ersten drei Ereignisse dieselbe Form von ‘;;;$x0;‘ aufweisen, was darauf hindeutet, dass in diesem Cluster häufig Ereignisse auftreten, die auf dem selben Status ausgeführt werden. In etwa einem Drittel der Anfragen tritt zusätzlich das Ereignis ‘;;$x0;;‘ auf, was zeigt, dass einige Ereignisse auf denselben Maschinen laufen. Das fünfthäufigste Ereignis in diesem Cluster ist eine Kombination der beiden vorherigen, nämlich ‘;;$x0;$x1;‘, was bedeutet, dass diese Ereignisse sowohl auf denselben Maschinen als auch mit demselben Status ausgeführt werden.

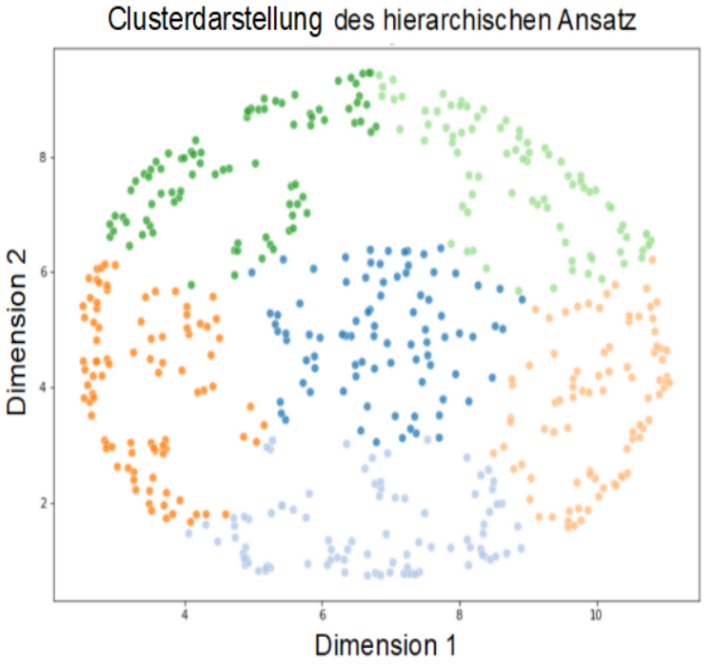

Abbildung 13: Visualisierung der Punkte des hierarchischen Ansatzes

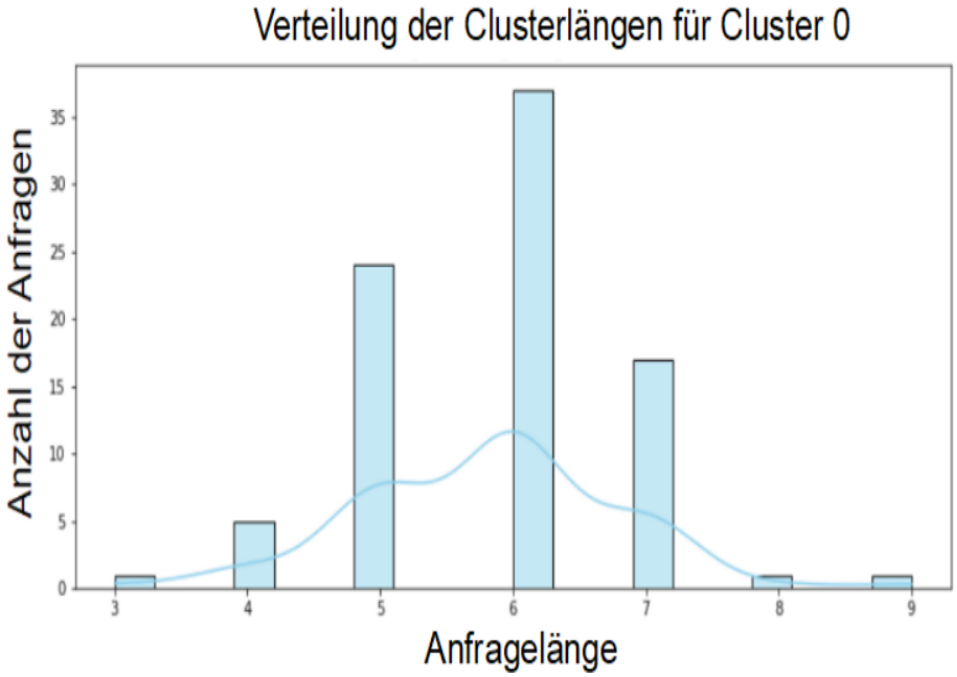

Abbildung 14: Längenverteilung Cluster 0

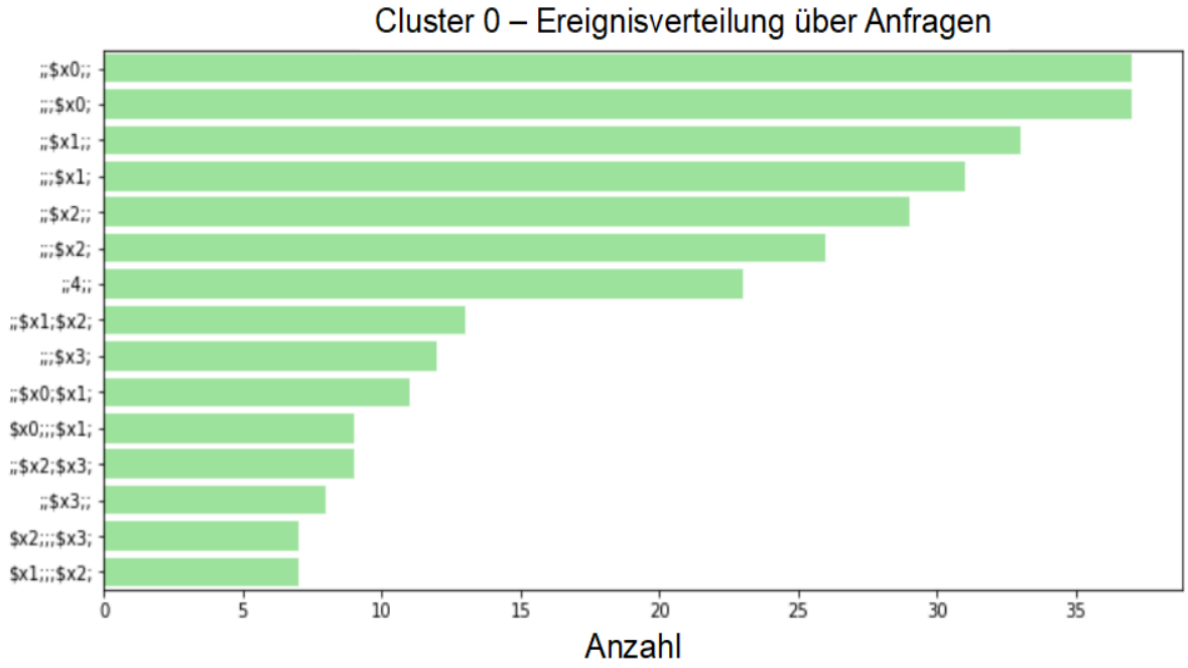

Abbildung 15: Anfragenhäufigkeit in Cluster 0

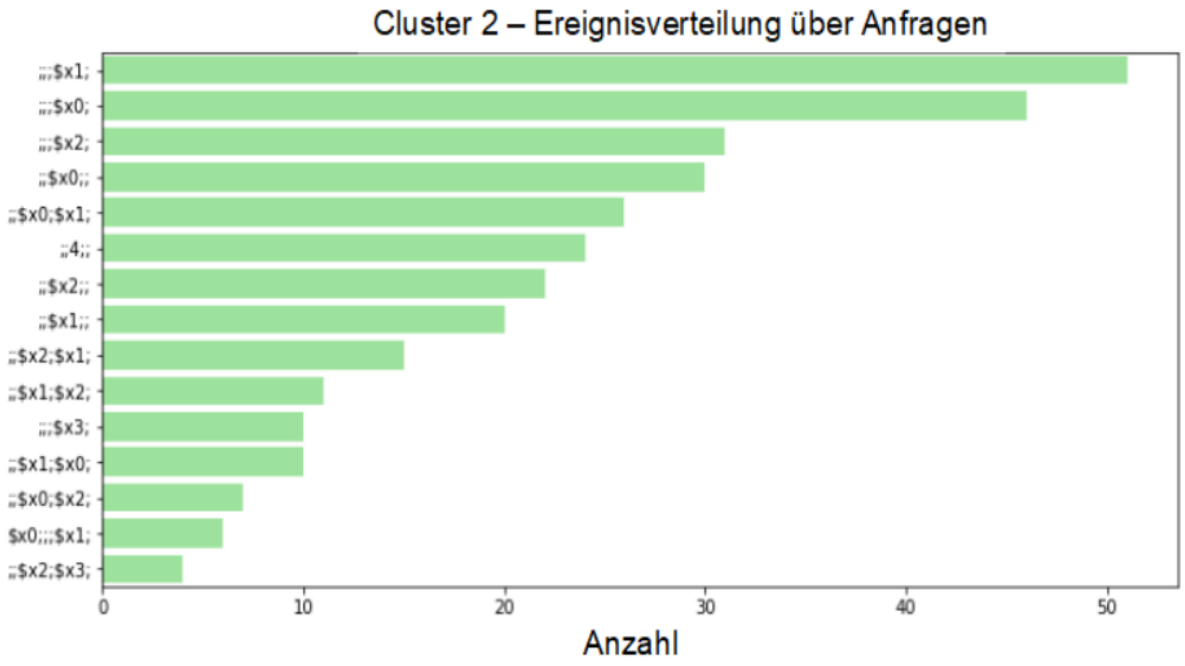

Abbildung 16: Anfragenhäufigkeit in Cluster 2

Syntaktischer Ansatz Bei der Clusterung der Anfragen anhand des syntaktischen Ansatzes werden 54 Cluster erstellt. Der erzielte Silhouetten-Score liegt bei 0,61, was auf eine gute Trennung der Cluster hindeutet. Die Cluster sind klar voneinander abgegrenzt und die Anfragen lassen sich weitgehend eindeutig zuordnen. Auch der Davies-Bouldin-Index von 0,7 spricht für eine qualitativ hochwertige Clusterung. So ist die durchschnittliche Streuung innerhalb der Cluster geringer als der Abstand zwischen den Clustern. In der Punkteverteilung 17 zeigt sich ebenfalls, dass einzelne Cluster deutlich voneinander getrennt sind, etwa das braune Cluster im Bereich um (0, 12) oder das blaue Cluster bei (–1, –3). Im Bereich rechts der X-Achse ab etwa 15 bilden sich hingegen viele kleinere Cluster, die in der zweidimensionalen Projektion eng beieinanderliegen, sodass visuell der Eindruck entsteht, sie könnten zu einer gemeinsamen Gruppe gehören. Die erstellten Cluster unterscheiden sich deutlich in ihrer Größe. In Abbildung 19 ist auf der X-Achse die Anzahl an Anfragen pro Cluster dargestellt, während die Y-Achse angibt, wie viele Cluster jeweils diese Größe aufweisen. Es existieren einzelne Cluster, die nur eine einzige Anfrage enthalten. Die häufigste Clustergröße liegt bei etwa fünf bis sieben Anfragen, danach nimmt die Häufigkeit sukzessive ab. Das größte Cluster umfasst 40 Anfragen und wird von zwei weiteren Clustern mit jeweils 32 Anfragen gefolgt. Auch die durchschnittliche Anfragelänge unterscheidet sich deutlich zwischen den Clustern und liegt zwischen 4 und 8,3 Ereignissen pro Anfrage. Ein Cluster mit lediglich drei Ereignissen im Durchschnitt wurde dabei nicht berücksichtigt, da es nur eine einzelne Anfrage enthält. Insgesamt zeigt sich, dass der syntaktische Ansatz stark von der Anfragelänge beeinflusst ist. In 30 der 54 Cluster liegt die Standardabweichung der Anfragelängen bei 0,0, was zeigt, dass alle Anfragen innerhalb dieser Cluster die gleiche Länge aufweisen. Dabei spielt die

Clustergröße keine Rolle. Es gibt sowohl große Cluster wie der mit 40 Anfragen als auch kleinere mit 1 bis 10 Anfragen, welche eine Standardabweichung von 0,0 aufweisen. Nur drei Cluster weichen signifikant davon ab und zeigen höhere Standardabweichungen von 0,816, 0,687 und 0,639. Alle übrigen Cluster liegen unterhalb von 0,5. In der Clusteraufteilung wurden drei Cluster erstellt, die jeweils nur eine einzige Anfrage enthalten. Dabei handelt es sich um die Anfragen ‘$x0;;;$x1; $x0;;;$x1; $x0;;;$x1;‘, ‘$x0;;;$x1; $x0;;;$x1; $x2;;;$x3; $x2;;;$x3; ;;4;;‘ und ‘;;;$x0; $x1;;;$x2; ;;;$x0; $x1;;;$x2;‘. Diese unterscheiden sich strukturell offenbar so stark von allen anderen Anfragen, dass sie keinem weiteren Cluster zugeordnet werden konnten. Im Fall der ersten Anfrage ist die geringe Länge mit nur drei Ereignissen vermutlich der Hauptgrund. Aufgrund der starken Längenabhängigkeit des syntaktischen Ansatzes reicht die Anzahl der Ereignisse nicht aus, um eine hinreichende Ähnlichkeit zu anderen, meist längeren Anfragen herzustellen. Selbst bei inhaltlicher Nähe wird die Differenz in der Länge als zu groß gewertet. Die beiden anderen Einzelanfragen mit jeweils vier bzw. fünf Ereignissen scheinen sich weniger durch ihre Länge, sondern vielmehr durch ihre einzigartigen Ereignisse zu unterscheiden. Ihre spezifische Struktur oder Kombination der Merkmale tritt offenbar in keiner anderen Anfrage in ähnlicher Form auf, sodass sie ebenfalls als eigenständig klassifiziert wurden. Es gibt aber auch mehrere Cluster, die trotz ihrer Größe eine bemerkenswert hohe Kompaktheit aufweisen. Drei Cluster mit jeweils mehr als zehn Anfragen zeigen eine durchschnittliche Distanz der Punkte zum Clustermittelpunkt von unter 0,3. Zum Vergleich: Der durchschnittliche Kohärenzwert für alle Cluster mit mehr als zehn Anfragen liegt bei 0,38, wodurch diese drei Cluster deutlich hervorstechen. Besonders auffällig ist dabei Cluster 15, das aus 15 Anfragen besteht und einen Kohärenzwert von lediglich 0,24 aufweist, was etwa 66 Prozent des Durchschnittswerts entspricht. Repräsentiert wird dieses Cluster durch die Anfrage ‘;;;$x0; ;;;$x1; ;;;$x1; ;;;$x2; ;;;$x2; ;;;$x1; ;;;$x0;‘, die dem Clustermittelpunkt am nächsten liegt. Alle Anfragen in Cluster 15 bestehen aus exakt sieben Ereignissen, wobei lediglich drei verschiedene Ereignistypen auftreten: ‘;;;$x0;‘, ‘;;;$x1;‘ und ‘;;;$x2;‘. Die Ereignisse verbinden sich ausschließlich in ihrem Status, während weitere Spezifikationen wie Maschinen-ID oder Priorität nicht vorkommen. Die hohe Kompaktheit des Clusters lässt sich somit durch die starke formale Ähnlichkeit der enthaltenen Anfragen erklären, die sich auf wenige, wiederkehrende Statusmuster beschränken.

Ein besonders auffälliger Fall unter den drei größten Clustern mit jeweils 32 beziehungsweise 40 Anfragen ist Cluster 4. Trotz seines Umfangs von 40 Elementen weist es mit einem Kohärenzwert von 0,287 eine ungewöhnlich hohe Kompaktheit auf. Auffällig ist zudem, dass sämtliche Anfragen innerhalb dieses Clusters exakt fünf Ereignisse enthalten. In Abbildung 18 zeigt die X-Achse, in wie vielen Anfragen ein bestimmtes Ereignis auftritt, während die Y-Achse die jeweiligen Ereignisse darstellt. Auffällig ist, dass viele Anfragen mit einem Ereignis beginnen, das eine bestimmte Maschinen-ID enthält, welche im weiteren Verlauf der Anfrage mindestens ein weiteres Mal referenziert wird. Dies lässt sich an den Ereignissen ‘;;$x0;;‘ und ‘;;$x0;$x1;‘ erkennen, die jeweils in mehr als 15 Anfragen vorkommen. Zudem ist zu beobachten, dass spätere Ereignisse häufig auf derselben Statusebene ausgeführt werden. Ein Beispiel hierfür ist das Ereignis ‘;;;$x3;‘, das ebenfalls in über 15 Anfragen erscheint. Darüber hinaus zeigen die häufigsten Ereignisse eine wiederkehrende Kombination aus identischer Maschinen-ID und gleichem

Status. So treten Ereignisse der Form ‘;;$x0;$x1;‘ sechsmal unter den zehn am häufigsten vorkommenden Ereignissen auf. Dies deutet darauf hin, dass die Anfragen in Cluster 4 eine starke strukturelle Ähnlichkeit aufweisen, sowohl hinsichtlich der Maschinen- als auch der Statuszuordnung der Ereignisse.

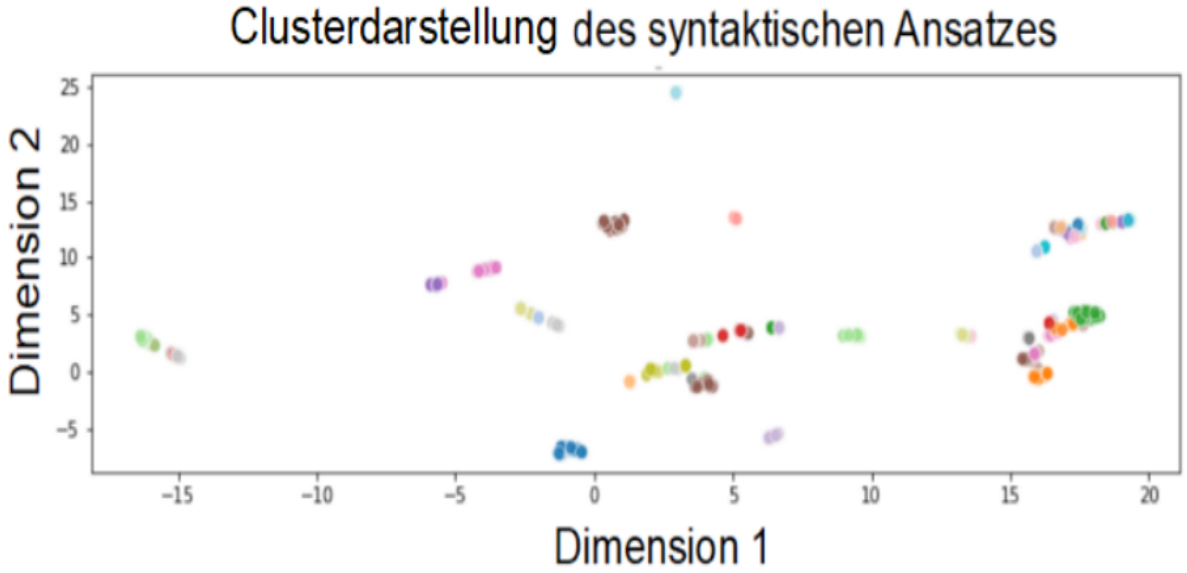

Abbildung 17: Visualisierung der Punkte des syntaktischen Ansatzes

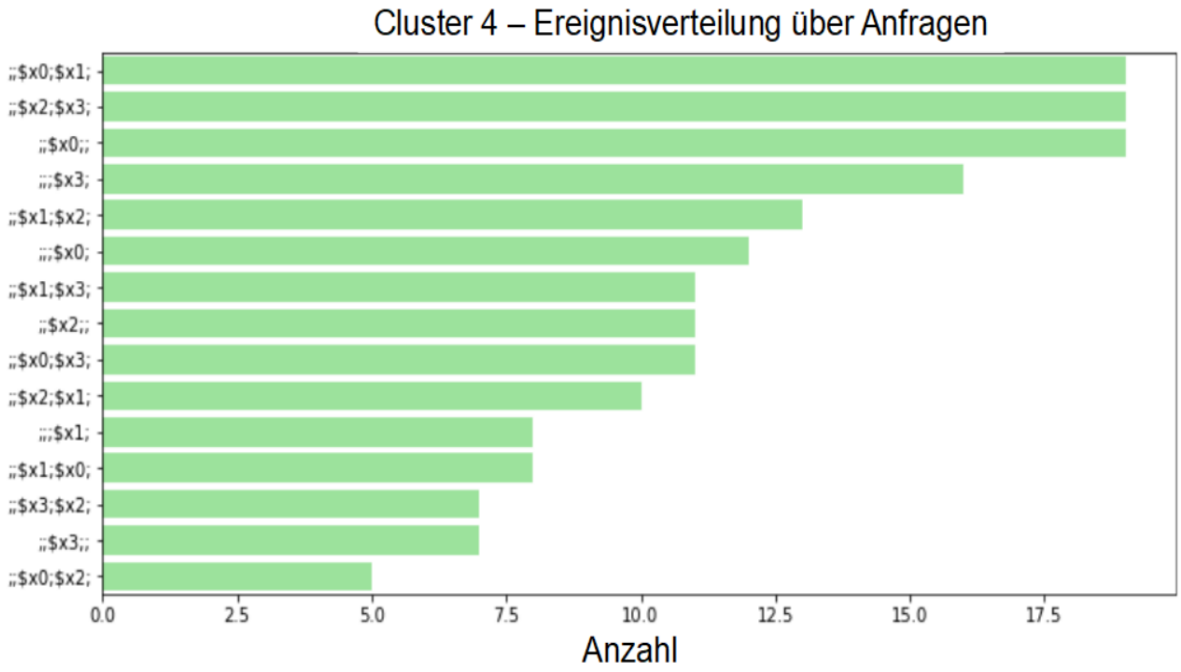

Abbildung 18: Anfragenhäufigkeit in Cluster 4

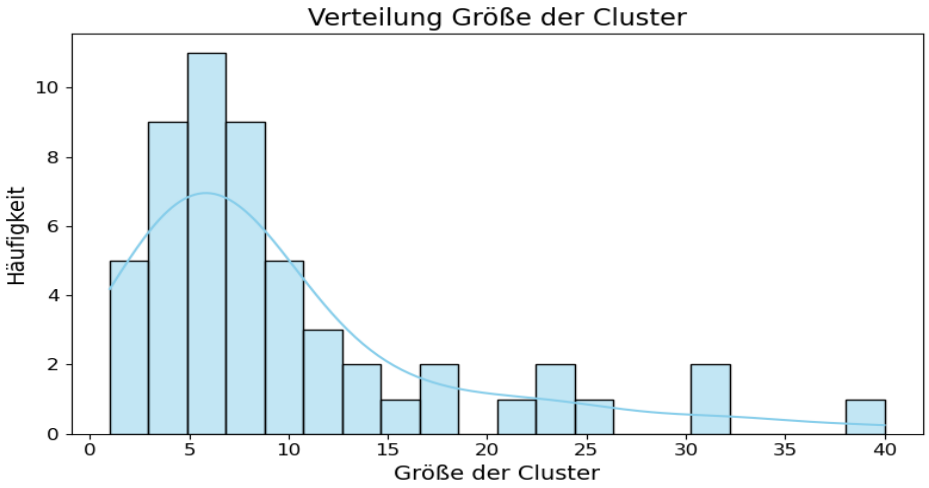

Abbildung 19: Größenverteilung der Cluster in dem syntaktischen Ansatz

Semantischer Ansatz Wie in Kapitel 5.3 erläutert, ermöglicht eine hohe Anzahl an Clustern zwar eine bessere Differenzierung, jedoch nimmt die Aussagekraft bei mehr als 200 Clustern deutlich ab. Viele Cluster enthalten dann nur noch Anfragen mit nahezu identischen Ereignissen, was den inhaltlichen Nutzen der Gruppierung stark einschränkt. Aus diesem Grund wurde die Clusterung mit einer Clusteranzahl von 48 durchgeführt. Der erzielte Silhouettenwert beträgt 0,36. Das deutet darauf hin, dass die Cluster grundsätzlich erkennbar sind, jedoch befinden sich viele Punkte nahe an den Grenzen zu anderen Clustern, wodurch ihre Zuordnung nicht eindeutig ist. Der Davies-Bouldin-Index beträgt 1,09, was darauf hinweist, dass die durchschnittliche Streuung innerhalb der Cluster größer ist als der Abstand zwischen den jeweiligen Mittelpunkten. In Abbildung 20 ist zu erkennen, dass einige Cluster klar voneinander getrennt und isoliert liegen, insbesondere bei Werten kleiner als 5 in der ersten Dimension. In diesem Bereich lassen sich einzelne Gruppen deutlich voneinander abgrenzen. Im Gegensatz dazu zeigt sich im Bereich zwischen 5 und 13 auf der ersten Dimension sowie zwischen 2 und 6 auf der zweiten Dimension eine dichte Ballung von Punkten. Die Cluster in diesem Bereich überlappen stark, was in der zweidimensionalen Darstellung eine klare Trennung erschwert. Diese Überlagerung könnte den vergleichsweise niedrigen Silhouetten Score erklären, da viele Punkte keiner eindeutig abgrenzbaren Gruppe zugeordnet werden können. Die Clustergrößen bei der semantischen Clusterung variieren zwischen 1 und 26 Anfragen, wobei der Großteil der Cluster eine Größe zwischen 10 und 13 aufweist (siehe Abbildung 21). Mehr als die Hälfte der Cluster enthält jedoch zehn oder weniger Anfragen. Auch die durchschnittliche Anzahl der enthaltenen Ereignisse pro Cluster variiert deutlich. Wie in Abbildung 22 dargestellt, gibt es Ausreißer in beide Richtungen. Ein Cluster umfasst Anfragen mit durchschnittlich nur drei Ereignissen, während ein anderes eine durchschnittliche Länge von 8,25 Ereignissen aufweist. Abgesehen von diesen Ausnahmen bewegen sich die durchschnittlichen Längen überwiegend zwischen vier und sieben Ereignissen. Über 50 Prozent der Cluster weisen dabei eine durchschnittliche Ereignisanzahl im Bereich zwischen 5 und 6,5 auf. Lässt man die vier Cluster außer Acht,

die eine Standardabweichung von 0 aufweisen, so variiert die Standardabweichung der Ereignislängen in den verbleibenden Clustern zwischen 0,2 und 0,8. Etwa die Hälfte der Cluster liegt dabei in einem Bereich zwischen 0,45 und 0,65. Das bedeutet, dass sich die Anfragen innerhalb eines Clusters im Schnitt um etwa ein halbes Ereignis in ihrer Länge unterscheiden. Diese Verteilung in Kombination mit der geringen Anzahl an Clustern, in denen alle Anfragen exakt die gleiche Länge haben, zeigt, dass der semantische Ansatz weniger stark von der Länge der Anfragen abhängt. Es ist also möglich, dass Anfragen mit unterschiedlichen Längen im selben Cluster landen, was darauf hindeutet, dass die Gruppierung eher auf inhaltlicher oder semantischer Ähnlichkeit basiert. Alle Cluster sind dabei sehr kompakt, mit Kohärenzwerten zwischen 0,0 und 0,003 liegen die Punkte innerhalb eines Clusters sehr nah am jeweiligen Mittelpunkt. Diese enge Anordnung wirkt auf den ersten Blick positiv, ist jedoch im Gesamtkontext eher kritisch zu sehen. Der mittelmäßige Silhouetten-Score sowie ein DBI-Wert größer als 1 deuten darauf hin, dass die Punkte insgesamt sehr dicht beieinander liegen, wodurch es schwierig ist, klare Clustergrenzen zu erkennen. Die hohe Kompaktheit der einzelnen Cluster ergibt sich somit eher aus der generellen Nähe der Datenpunkte zueinander als aus einer tatsächlich passenden Differenzierung. Dadurch verliert der Kompaktheitswert in diesem Fall an Aussagekraft. Nachdem die Makroanalyse der Clusterung abgeschlossen ist, können nun einzelne Cluster im Detail betrachtet werden. Auch bei diesem Ansatz existiert ein Cluster, das nur eine einzige Anfrage enthält. Konkret handelt es sich um die Anfrage ‘$x0;;;$x1; $x0;;;$x1; $x0;;;$x1;‘. Offenbar weist diese Anfrage in ihrer semantischen Struktur keine ausreichenden Ähnlichkeiten zu allen anderen Anfragen auf, sodass sie keinem weiteren Cluster zugeordnet werden konnte und daher als eigenständige Gruppe klassifiziert wurde. Ein weiteres auffälliges Cluster ist Cluster 10, das insbesondere durch die Länge der enthaltenen Anfragen hervorsticht. Es umfasst vier Anfragen, von denen drei jeweils acht Ereignisse enthalten, während eine Anfrage neun Ereignisse umfasst. Damit ergibt sich eine durchschnittliche Anfragelänge von 8,25, welches der höchste Wert aller Cluster ist. Trotz der vergleichsweise hohen Anzahl an Ereignissen nutzen die Anfragen lediglich fünf verschiedene Ereignistypen, die sich innerhalb der Anfragen mehrfach wiederholen. Charakteristisch für alle Anfragen ist zudem, dass sie mit dem Ereignis ‘;;4;;‘ enden, wodurch jede Anfrage mit einer Ausführung auf der Maschine mit der ID 4 abschließt. Davor folgen ausschließlich Ereignisse der Form ‘;;;$x0;‘, also Ereignisse, die auf identischen Statuswerten ablaufen. Innerhalb der Abläufe wechselt dieser Status mindestens zweimal, bei der längsten Anfrage sogar dreimal, bevor schließlich das abschließende Maschinenereignis erfolgt. Dieses Cluster zeichnet sich somit durch eine ausgeprägte Wiederholung statusbasierter Ereignisse aus, gefolgt von einem einheitlichen Abschluss auf einer spezifischen Maschine. Die starke Ähnlichkeit in der Struktur und dem Ablauf macht Cluster 10 zu einer inhaltlich zusammenhängenden und von den anderen Clustern unabhängigen Gruppe. Der Fokus liegt nun auf Cluster 9, dessen Werte häufig im Bereich der Durchschnitte liegen. Es umfasst 8 Anfragen mit einer durchschnittlichen Länge von 5,125 Ereignissen. Diese setzt sich zusammen aus einer Anfrage mit 4 Ereignissen, fünf Anfragen mit jeweils 5 Ereignissen und zwei Anfragen mit 6 Ereignissen. Insgesamt kommen nur vier verschiedene Ereignisse in diesem Cluster vor. Alle Anfragen enden in diesem Cluster mit dem Ereignis ‘;;4;;‘, das auf eine Ausführung

auf der Maschine mit der ID 4 beschreibt. Die übrigen Ereignisse sind ‘;;$x0;$x1;‘, ‘;;$x0;;‘ und ‘;;;$x1;‘. Alle Anfragen in diesem Cluster beginnen mit einem Ereignis, das entweder allein auf einer bestimmten Maschinen-ID oder zusammen mit einem bestimmten Status ausgeführt wird. Die restlichen Ereignisse nutzen dann entweder dieselbe Maschine, denselben Status oder beides. Was eine gemeinsame inhaltliche Ähnlichkeit zeigt.
Zuletzt wird Cluster 1 genauer untersucht. Cluster 1 ist mit 20 Anfragen eines der größeren Cluster. Es enthält eine Anfrage mit 5 Ereignissen, sechs Anfragen mit jeweils 6 Ereignissen, zwölf mit 7 Ereignissen und eine mit 8 Ereignissen. Daraus ergibt sich eine durchschnittliche Anfragelänge von 6,65. Insgesamt kommen acht verschiedene Ereignisse vor, wobei zwei davon nur in einer einzigen Anfrage enthalten sind. Die betroffene Anfrage lautet: ‘$x0;;;$x1; $x0;;$x2;$x1; ;;;$x1; ;;$x2;; ;;;$x1;‘ und hebt sich strukturell von den übrigen Anfragen im Cluster ab. Da die ersten beiden Ereignisse sonst nirgends im Cluster vorkommen, zeigt sich hier möglicherweise der Vorteil des semantischen Ansatzes. Die Anfrage scheint trotz struktureller Unterschiede inhaltlich gut in die Gruppe zu passen. Ohne diese spezielle Anfrage beschränken sich die restlichen Anfragen auf sechs verschiedene Ereignisse. Dabei beginnen alle mit dem Ereignis ‘;;;$x0;‘, gefolgt von mindestens einem Vorkommen von ‘;;;$x1;‘. Darauf folgt jeweils ein Ereignis der Form ‘;;$x2;;‘, ‘;;$x2;$x0;‘ oder ‘;;$x2;$x1;‘. Am Ende steht immer das Ereignis ‘;;4;;‘. Insgesamt folgt der Ablauf einem ähnlichen Muster. Die Anfragen beginnen mit einem bestimmten Status, der im Verlauf der Ausführung meist wechselt. Anschließend tritt ein Ereignis auf, das eine bestimmte Maschine festlegt, auf der mit einem der beiden Status der Job weiter ausgeführt wird, bis die Ereigniskette schließlich mit einer Ausführung auf der Maschine mit der ID 4 endet. Die sechs definierten Ereignisse können dabei mehrfach auftreten, doch der beschriebene Ablauf bildet das Grundgerüst aller Anfragen in diesem Cluster.

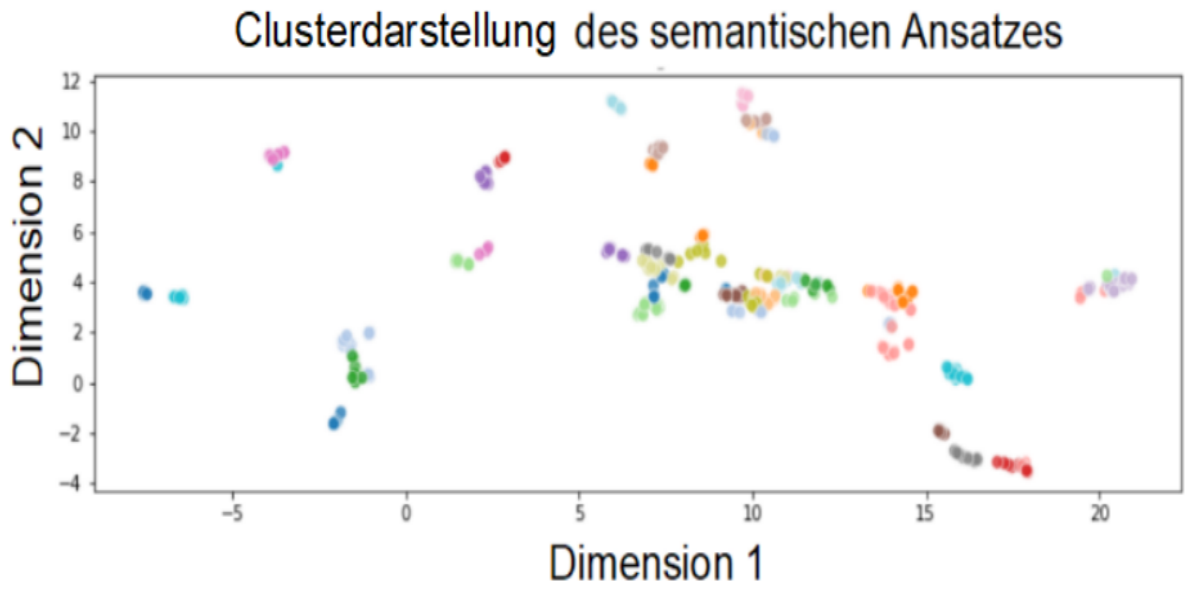

Abbildung 20: Visualisierung der Punkte des semantischen Ansatzes

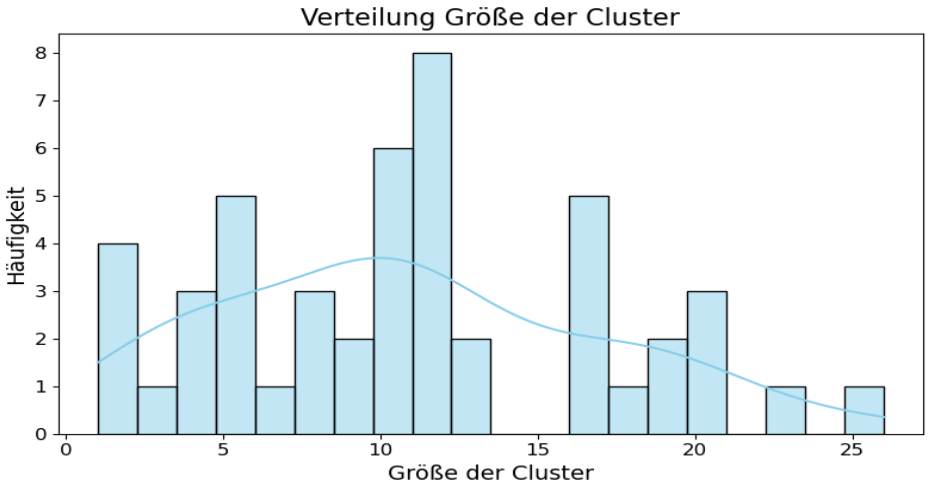

Abbildung 21: Verteilung der Clustergrößen bei dem semantischen Ansatz

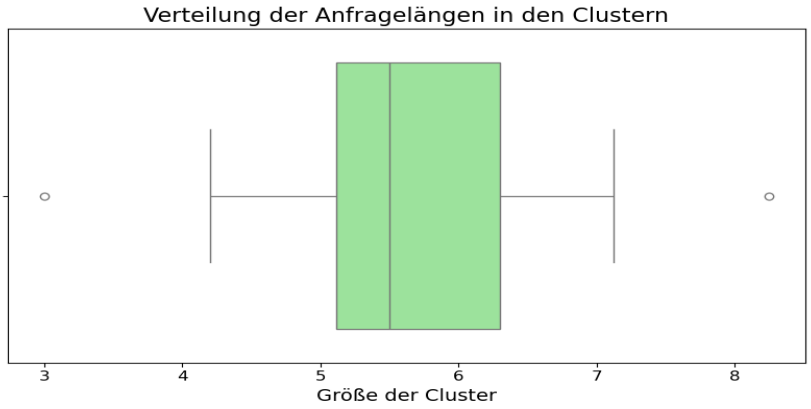

Abbildung 22: Verteilung der durchschnittlichen Anfragelänge bei dem semantischen Ansatzes

5.5 Zusammenfassung und Ausblick

In dieser Arbeit wurden drei Ansätze zur Clusterung von Anfragen über Ereignisströme untersucht, um geeignete Methoden für die Clusterbildung zu identifizieren. In diesem Abschnitt werden die gewonnenen Erkenntnisse zusammengefasst und ein Ausblick auf weitere Forschungsmöglichkeiten gegeben.

Zusammenfassung Bei dem hierarchischen Strukturansatz wurde eine vorgegebene Struktur genutzt, die alle zu clusternden Anfragen beinhaltete. Durch ihre Position in der Struktur wurde ein Zusammenhang zwischen den Anfragen hergestellt. Dabei bestand die Möglichkeit, an drei Parametern zu variieren, die die Gewichte der Kanten beeinflussten, um die Clusterbildung zu steuern. Für die besten Clusterungen waren

die Kantengewichte jedoch obsolet. Die besten Cluster, also jene mit der eindeutigsten Clustertrennung, wurden mit nur wenigen Clustern erreicht, was wahrscheinlich auf die Struktur zurückzuführen ist. Es schien, dass die Struktur so robust war, dass die Gewichte in den besten Clustern keinen Einfluss hatten, weil die Struktur so starke Unterstrukturen aufwies, dass die Anfragen auf Grundlage dieser Unterstrukturen geclustert wurden. Dadurch entsteht das Problem, dass die Relation zwischen zwei Anfragen nur schwer beeinflusst werden kann, da sie bereits festgelegt ist. Dies kann positiv sein, wenn die Struktur die Anfragen in eine passende Relation eingefügt hat, die der Intention des Forschers entspricht. Es kann jedoch auch negativ sein, da die Anfragen in eine Struktur „gezwungen“ wurden, in der die Relationen bereits feststehen und nur wenig Variationen zugelassen wird. In diesem Fall kann es vorkommen, dass Anfragen, die man als ähnlich bewertet, in der Struktur weit voneinander entfernt liegen und somit niemals in denselben Cluster eingeordnet werden. Erst bei einer höheren Anzahl an Clustern zeigte sich die Wirkung der Gewichte, was jedoch zu einer Verschlechterung der Clustertrennung führte, sodass diese Clusterung nicht weiter berücksichtigt wurden. In den wenigen erzeugten Clustern aus dem Praxisteil waren Anfragen unterschiedlicher Längen enthalten, was darauf hindeutet, dass dieser Ansatz besonders für Cluster geeignet ist, bei denen die Länge der Anfrage keine Rolle spielt. Abgesehen davon war es schwierig, Gemeinsamkeiten zwischen den Anfragen zu identifizieren, da diese vor allem auf der Position der Anfragen in der hierarchischen Struktur basierten. Die Anfragen in den Clustern ähnelten sich vor allem durch eine ähnliche Tiefe. Dies verdeutlicht, dass die gegebene hierarchische Struktur für die Clusterbildung von entscheidender Bedeutung war und der Ansatz somit seine Aufgabe erfüllt hat. Bei der Nutzung dieses Ansatzes muss man sich also für eine passende Anzahl an Clustern entscheiden, wobei die Clusterung dann entweder auf Grundlage der Unterstrukturen oder auf der Position der Anfragen in der gesamten Struktur basiert. Letzteres führt jedoch, wie bereits erwähnt, zu einer Verschlechterung der Clusterqualität, sodass diese Entscheidung eher ein Trade-off darstellt. Es gilt abzuwägen, ob man große Cluster bevorzugt, die viel Interpretationsspielraum bieten, aber keine klaren Aussagen zulassen, oder kleinere Cluster, die zwar eine schlechtere Trennbarkeit aufweisen, dafür jedoch vermutlich eindeutigere Aussagen ermöglichen.

Der syntaktische Ansatz hatte das Ziel, die Anfragen aufgrund ähnlicher Strukturen zu clustern, sodass in einem Cluster strukturell gleiche Anfragen enthalten sind. Anfragen wurden als ähnlich betrachtet, wenn sie dieselben Ereignisse und Ereignisfolgen teilten, aber auch, wenn sie dieselben Attribute und Attributketten in der gleichen Dimension aufwiesen. Dieser Ansatz war sehr flexibel, da er die Kontrolle über sechs verschiedene Parameter ermöglichte, mit denen die Clusterung gezielt verändert werden konnte. Nach zahlreichen Tests zeigte sich, dass die im Theorieteil gefundene Reihenfolge der Gewichte angepasst werden musste. Da die Struktur der Anfragen im Datensatz bei vielen Anfragen sehr ähnlich war, führte ein hohes Gewicht für N-Gramme zu einer erheblichen Verschlechterung der Clusterung. Mit den veränderten Gewichten konnte schließlich eine Clusterung erreicht werden, die eine klare Abgrenzung und eine gute Zuordnung der Punkte aufwies. Es wurden sowohl kleine als auch große Cluster gebildet, welche aufgrund ihrer Struktur als ähnlich galten. Auffällig war jedoch, dass die Elemente innerhalb der Cluster oft eine ähnliche Größe aufwiesen, was darauf hindeutet, dass dieser Ansatz stark längenabhängig

ist. Wenn das Ziel darin besteht, Anfragen mit unterschiedlichen Längen zu clustern, ist dieser Ansatz daher nicht zu empfehlen. Die größte Herausforderung bei diesem Ansatz bestand darin, die passenden Variablengewichte zu finden. Es empfiehlt sich daher, bereits im Vorfeld eine grobe Vorstellung davon zu haben, welche Eigenschaften einen großen Einfluss auf die Clusterung haben sollen und welche weniger relevant sind, wobei das wie im Praxisteil auch noch variieren kann. Andernfalls muss man durch zahlreiche Testdurchläufe die optimalen Parameter ermitteln, was einen erheblichen Zeitaufwand bedeutet. Die dabei entstandenen Cluster sahen jedoch vielversprechend aus. Sie waren deutlich voneinander getrennt, und auch die näher beschriebenen Cluster wiesen eine ähnliche Struktur auf, was auf eine konsistente und gut funktionierende Clusterung hinweist.
Der semantische Ansatz verfolgte das Ziel, Anfragen basierend auf ihrem Vorkommen in ähnlichen Kontexten zu clustern. Mithilfe der erweiterten Methode von FastText wurden dabei nicht nur die Embeddings für ein ganzes Ereignis berechnet, sondern auch die Embeddings für einzelne Zeichenketten in den Ereignissen. Dadurch ergab sich die Möglichkeit, auch für seltene Ereignisse passende Embeddings zu erstellen, die auf dem Vorkommen ähnlicher Zeichenfrequenzen in anderen Anfragen basierten. Ein Umstand, der für die wenigen, aber ähnlichen Anfragen von Vorteil war. Für die Vektorisierung wurde das vorgefertigte FastText-Modell von Facebook verwendet. Dies führte zu einer Clusterstruktur, die insgesamt gute Ergebnisse erzielte, jedoch eine Clusteranzahl von mehr als 200 erzeugte. Dadurch enthielten viele Cluster nur ein oder zwei Anfragen, die sich durch identische Ereignisse hervorhoben. Aufgrund der Durchschnittsberechnung der Ereignis-Embeddings, um das Anfrage-Embedding zu erhalten, wurden diese Anfragen dann in dasselbe Cluster eingeordnet. Die Durchschnittsberechnung behandelte Anfragen mit gleichen Ereignissen ähnlich wie ein Bag-of-Words-Modell, wodurch sie die selben Vektoren erhielten und in das gleiche Cluster gelangten. Das warf die Frage auf, ob die Cluster mit hohen Clusteranzahlen tatsächlich die semantische Ähnlichkeit widerspiegelten. Um dieses Problem zu beheben, wurde die Clusteranzahl künstlich reduziert, was die Anzahl der Anfragen in den Clustern erhöhte. Allerdings führte dies auch zu einer Verschlechterung der Clusterqualität. Dies führte zu Gruppen, in denen Anfragen häufig unterschiedliche Längen aufwiesen, was darauf hindeutet, dass die Länge eine untergeordnete Rolle spielte. Auch strukturell abweichende Anfragen wurden teilweise denselben Clustern zugeordnet. Beides spricht dafür, dass semantische Ähnlichkeiten und Bedeutungen erkannt wurden und der Ansatz seine Aufgabe grundsätzlich erfüllte. Der größte Nachteil dieses Ansatzes lag in der Entfernung der Punkte. Viele Cluster wiesen zwar eine sehr geringe Streuung auf, jedoch zeigte der DBI-Wert, dass die Streuung in einigen Fällen größer war als der Abstand zwischen den Mittelpunkten der Cluster. Dies zeigt, dass die Punkte generell sehr dicht beieinander lagen. Dadurch kann es vorkommen, dass Punkte falschen Clustern zugeordnet werden, da die Abgrenzung zwischen den Clustern unscharf bleibt. Mit diesem Ansatz wurden also Gruppen gebildet, die grundsätzlich zu einer semantischen Clusterung führten. Allerdings zeigte sich, dass die Punkte sehr dicht beieinander lagen, was zu einer schwachen Differenzierung zwischen den Clustern führte, sodass der Ansatz nur eine mäßige Clusterqualität aufwies.

Vergleich In Tabelle 2 werden die drei verschiedenen Ansätze miteinander verglichen. Dabei wurden die beiden besten Clusterungen des Hierarchischen- und Syntaxansatzes mit einer Clusterung des semantischen Ansatzes verglichen, die im Kapitel 5.4 erstellt wurden. Es ist zu beachten, dass der semantische Ansatz möglicherweise ein besseres Cluster erzeugen könnte, es jedoch unklar bleibt, ob dieses Cluster dann tatsächlich die semantische Aussagekraft der Anfragen angemessen widerspiegelt hätte. Der hierarchische Ansatz verwendete nur 6 Cluster, während der syntaktische Ansatz 54 und der semantische Ansatz 48 Cluster nutzten. Dabei bildete der hierarchische Ansatz auch die größten Cluster, was aufgrund der geringen Clusteranzahl zu erwarten war. Aufrund der ähnlichen Clusteranzahl lassen sich die anderen beiden Ansätze besser vergleichen. Beim syntaktischen Ansatz streuten die Clustergrößen stärker, wobei der Median bei nur 7 lag und 50 Prozent der Clustergrößen zwischen 4,25 und 11 lagen. Dies deutet darauf hin, dass die Cluster tendenziell kleiner sind als beim semantischen Ansatz, jedoch mit größeren Ausreißern. Beim semantischen Ansatz sind die Clustergrößen gleichmäßiger verteilt, wobei 50 Prozent der Clustergrößen zwischen 5 und 16 liegen. Dies zeigt, dass in diesem Ansatz häufiger größere Cluster gebildet werden, während die größten Cluster jedoch beim syntaktischen Ansatz auftreten. Weiterhin war der syntaktische Ansatz der einzige, bei dem eine starke Längenabhängigkeit zu beobachten war. Während in den anderen Ansätzen die Cluster größere Schwankungen in der Länge der Anfragen aufwiesen, hatten die Anfragen im syntaktischen Ansatz innerhalb eines Clusters oft ähnliche Längen oder nur kleine Abweichungen.
Der hierarchische und der semantische Ansatz erzielten ähnliche Silhouetten-Scores von 0,38 bzw. 0,36, was darauf hinweist, dass die Punkte in diesen Ansätzen oft in passende Cluster gruppiert wurden, jedoch auch nahegelegene Cluster existierten, in denen sie ebenfalls gut gepasst hätten. Der syntaktische Ansatz hebt sich jedoch mit einem Silhouetten-Score von 0,61 ab, was darauf hindeutet, dass die Punkte häufig eindeutig einem Cluster zugeordnet werden konnten. Obwohl der syntaktische und der hierarchische Ansatz ähnliche DBI-Werte aufwiesen, zeigt sich, dass die Clusterung des syntaktischen Ansatzes, durch den hohen Silhouette Score, am besten war. Der semantische Ansatz erzielte mit einem DBI von 1,09 den schlechtesten DBI-Wert, was jedoch durch die enge Verteilung der Punkte erklärt werden kann, wie bereits zuvor gezeigt wurde.
Die Punkteverteilung des strukturellen Ansatzes war ziemlich gleichmäßig, was darauf hinweist, dass die Gewichte und die Struktur in diesem Ansatz nicht hilfreich waren, da sie die Unterschiede zwischen den Anfragen nicht ausreichend herausstellen konnte. Bei den anderen beiden Ansätzen hingegen waren die Punkteverteilungen deutlich klumpiger, was zeigt, dass diese Ansätze die relevanten Merkmale besser voneinander trennen konnten.

	Anzahl	Silhouetten Score	Davies-Bouldin-Index
Hierarchie	6	0.38	0.76
Syntax	54	0.61	0.7
Semantik	48	0.36	1.09

Tabelle 2: Vergleich der Ansätze

Ausblick In dieser Arbeit wurden lediglich die Grundlagen für die Erstellung von Clusterungen für Anfragen über Ereignisströme gelegt. Es gibt zahlreiche Aspekte, die in der Zukunft weiter untersucht werden können, um die Ansätze zu optimieren.
Für den hierarchischen Ansatz könnte man die Clusterergebnisse zunächst von Experten sichten lassen, um zu prüfen, ob die Struktur tatsächlich die passenden Eigenschaften als Relation abbildet. So könnte man feststellen, ob es sinnvoll ist, auf Grundlage dieser Struktur Cluster zu bilden. Darüber hinaus könnten alternative Ähnlichkeitsmetriken eingesetzt werden, die andere Merkmale der Struktur stärker nutzen, um auf diese Weise andere Cluster aus den Anfragen zu erzeugen.
Das Problem des semantischen Ansatzes, dass die Anfragen nicht klar genug differenziert wurden, ließe sich möglicherweise durch ein größeres Datenset beheben. Mit einer größeren Datenbasis könnten die Modelle feiner zwischen unterschiedlichen Ereignissen und deren Bedeutungen unterscheiden, da ihnen mehr Trainingsbeispiele zur Verfügung stehen. Daher sollte bei einer zukünftigen Weiterverfolgung dieses Ansatzes versucht werden, ein umfangreicheres Datenset zu nutzen, um semantische Unterschiede besser abzubilden und die Ereignis-Embeddings klarer voneinander abzugrenzen. Zudem sollte auch die Methode zur Bildung der Anfrage-Embeddings überdacht werden. Der Durchschnitt der Ereignis-Embeddings liefert zwar bereits gewisse Ergebnisse, kann durch die starke Abstraktion jedoch auch zu ungenauen oder irreführenden Zuordnungen führen. Möglicherweise gibt es dort bessere Metriken, um den Anfragevektor zu erstellen.
Auch bei der Clusterung besteht Potenzial, in Zukunft alternative Methoden auszuprobieren. Der K-Means-Algorithmus bringt zwar bestimmte Vor- und Nachteile mit sich und wurde deshalb in dieser Arbeit eingesetzt, dennoch bieten sich weitere Ansätze an. So könnte die Clusterung beispielsweise nicht auf dem euklidischen Abstand, sondern auf der Kosinus-Ähnlichkeit basieren, also dem Winkel zwischen zwei Anfragen. Dadurch würde die Länge der Anfragen deutlich weniger Einfluss auf die Ähnlichkeitsberechnung nehmen. Was zu anderen Ergebnissen bei dem syntaktischen Ansatz führen würde. Alternativ könnte man anstelle des K-Means-Algorithmus, der auf Distanzminimierung basiert und Cluster durch die geringste Entfernung zu einem Zentroiden bildet, auch einen Algorithmus verwenden, der auf einem dichtebasierten Ansatz beruht. Dabei werden Cluster anhand der Punktedichte gebildet, was zu einer anderen Gruppierung der Anfragen führen und möglicherweise bessere Ergebnisse liefern könnte, insbesondere bei ungleichmäßig verteilten oder nicht-kugelförmigen Clustern, wie es teilweise der Fall war.
Abschließend könnte ein vordefinierter Datensatz mit bereits vergebenen Labels einen wertvollen Beitrag zur zukünftigen Clustererstellung leisten. Durch die Verwendung eines solchen Datensatzes, in dem Anfragen bereits als ähnlich gekennzeichnet sind, wäre es möglich, die Ansätze gezielt zu evaluieren und ihre Leistung zu verbessern. Darüber hinaus ließe sich der Datensatz als Trainingsgrundlage nutzen, um die Modelle gezielt so zu optimieren, dass sie ähnliche Anfragen präziser zu gemeinsamen Clustern zusammenführen. Ein solcher Ansatz würde nicht nur die Qualität der Clusterung steigern, sondern auch die Expertise von Fachleuten oder bereits bestehenden Labels in den Prozess integrieren und so zu einer präziseren und aussagekräftigeren Gruppierung führen.

6 Fazit

Der hierarchische Ansatz ist nur dann zu empfehlen, wenn die zugrunde liegende Struktur tatsächlich die Relation widerspiegelt, auf deren Basis die Clusterung erfolgen soll. In diesem Fall lassen sich mit den Parametern gezielt Feinabstimmungen vornehmen. Ist dies jedoch nicht gegeben, eignet sich der Ansatz aufgrund seiner festen Struktur und der insgesamt etwas schwächeren Clusterqualität nur bedingt. Ohne ein Verständnis für die vorliegende Struktur sind die Ergebnisse schwer interpretierbar und die Auswertung entsprechend eingeschränkt.
Der syntaktische Ansatz eignet sich am besten für kleine bis mittelgroße Anfragesets, da er durch die zahlreichen anpassbaren Parameter eine flexible Gruppierung nach der definierten Ähnlichkeit ermöglicht. Allerdings ist dieser Ansatz stark größenabhängig. Möchte man einen größenunabhängigen Ansatz verwenden, wäre es sinnvoll, diesen Ansatz zu erweitern, indem man die Clusterung vom euklidischen Abstand auf einen Kosinuswinkel-Abstand umstellt, um die Größenabhängigkeit zu eliminieren.
Der semantische Ansatz hat in dieser Arbeit noch nicht sein volles Potenzial entfaltet und ist in der aktuellen Form noch nicht empfehlenswert, da er noch nicht ausgereift ist. Allerdings könnte dieser Ansatz in der Zukunft sehr interessant werden, insbesondere wenn die Datensätze größer werden. Sollte es gelingen, größere Datenmengen an Anfragen zu sammeln und dabei die Laufzeit in Grenzen zu halten, könnte dieser Ansatz semantische Gemeinsamkeiten zwischen Anfragen und Ereignissen aufdecken, die bislang schwer erkennbar oder sogar unsichtbar waren.

Literatur

[1] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, Cham, 2018.

[2] K. Backhaus, B. Erichson, S. Gensler, R. Weiber, and T. Weiber. *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. Springer Gabler, Wiesbaden, 17., aktualisierte auflage edition, 2023.

[3] M. Baude. Projektimplementierung zur bachelorarbeit: Clusterung von anfragen über ereignisströme. zugriff: https://scm.cms.hu-berlin.de/baudemor/queryanalyzer, 2025. Zugriff am 01.05.2025.

[4] M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2):75–94, 2005.

[5] A. Beutelspacher. *Lineare Algebra: Eine Einführung in die Wissenschaft der Vektoren, Abbildungen und Matrizen*. Springer Spektrum Wiesbaden, Wiesbaden, 2014.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information, 2017.

[7] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB'03, page 73–78. AAAI Press, 2003.

[8] D. Davies and D. Bouldin. A cluster separation measure. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1:224 – 227, 05 1979.

[9] M. Eckert. Complex event processing with xchangeeq, Dezember 2008.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[11] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[12] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 1231–1239, New York, NY, USA, 2012. Association for Computing Machinery.

[13] S. Kantabutra and A. Couch. Parallel k-means clustering algorithm on nows. *NOCTEC Technical Journal*, 1, January 2000.

[14] D. Karani. Introduction to word embedding and word2vec. *Towards Data Science*, 2018.

[15] S. Kleest-Meißner. *Exploring the Complexity of Event Query Discovery.* PhD thesis, Humboldt-Universität zu Berlin, 2024.

[16] S. Kleest-Meißner, R. Sattler, M. L. Schmid, N. Schweikardt, and M. Weidlich. Discovering event queries from traces: Laying foundations for subsequence-queries with wildcards and gap-size constraints. In D. Olteanu and N. Vortmeier, editors, *25th International Conference on Database Theory (ICDT 2022)*, volume 220 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:21, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[17] S. Kleest-Meißner, R. Sattler, M. L. Schmid, N. Schweikardt, and M. Weidlich. Discovering multi-dimensional subsequence queries from traces – from theory to practice. In *BTW 2023*, pages 511–533. Gesellschaft für Informatik e.V., Bonn, 2023.

[18] I. Kolchinsky and A. Schuster. Join query optimization techniques for complex event processing applications, 2018.

[19] J. B. Kruskal and M. Wish. *Multidimensional Scaling.* Quantitative Applications in the Social Sciences. SAGE Publications, Inc., Thousand Oaks, CA, 1978. Accessed: 7 Mar 2025.

[20] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, Feb. 1966.

[21] M. Leys. The art of pooling embeddings, June 2022. Accessed: 2024-02-26.

[22] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, USA, 2008.

[23] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[24] G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, Nov. 1995.

[25] S. Mukherjee. Pooling in embedding, September 2024. Accessed: 2024-02-26.

[26] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[27] C. Reiss, J. Wilkes, and J. L. Hellerstein. Google cluster-usage traces: Format + schema. White Paper, 2011. pp. 1–14.

[28] D. M. Saputra and L. D. Oswari. Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. *Proceedings of the Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)*, 2020.

[29] R. Sattler, S. Kleest-Meißner, S. Lange, M. L. Schmid, N. Schweikardt, and M. Weidlich. Disces: Systematic discovery of event stream queries. *Proc. ACM Manag. Data*, 3(1), Feb. 2025.

[30] A. Schwartz. Einführung in die graphentheorie, August 2013. Universität Würzburg, Zugriff am 6. März 2024.

[31] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 801–809, 2011.

[32] W. S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17:401–419, 1952.

[33] A. Widder. Complex event processing und maschinelle lernverfahren: Entwicklung eines hybrid-modells zur erkennung von identitätsdiebstahl beim online-banking, 2011.

[34] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, USA, 1994. Association for Computational Linguistics.

[35] P. Yadav, D. Sarkar, D. Salwala, and E. Curry. Traffic prediction framework for openstreetmap using deep learning based complex event processing and open traffic cameras. *CoRR*, abs/2008.00928, 2020.

[36] S. Zhang, H. T. Vo, D. Dahlmeier, and B. He. Multi-query optimization for complex event processing in sap esp. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1213–1224, 2017.

[37] Q. Zhou, Y. Simmhan, and V. K. Prasanna. On using complex event processing for dynamic demand response optimization in microgrid. *CoRR*, abs/1311.6146, 2013.

Appendix

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den 7. Juli 2025 .....