FACHHOCHSCHULE
ERFURT UNIVERSITY
OF APPLIED SCIENCES
Angewandte
Informatik

Masterarbeit
in der Angewandten Informatik
Nr. AI-2021-MA-014

Generisches Clustern hoch-komplexer Produktdaten

Hannes Dröse

Abgabedatum: 25.04.2022

Prof. Dr. Ines-Kerstin Rossak
Dipl.-Inform. Robert Queck

Zusammenfassung

Die Kurzfassung wird zumeist als letzter Abschnitt geschrieben. Sie soll auf einer Seite den Hintergrund bzw. die Motivation sowie die zentralen Ergebnisse der Arbeit zusammenfassen. Der Zweck dieses Texts ist es, einem Recherchierenden oder Informationssuchenden zu einem bestimmten Thema zu signalisieren, ob er in diesem Werk für ihn relevante Informationen finden wird: Lohnt es sich, die Arbeit zu bestellen oder zu kaufen? Was ist neu?
Die Kurzfassung ist das sichtbare Aushängeschild der Arbeit und dient der Aufnahme in Referenzdatenbanken, in Online-Literatur-Shops u. Ä. Schauen Sie sich die Abstracts in der Fachliteratur (etwa in Fachbüchern, am Anfang von Zeitschriftenartikeln, in Konferenz- Proceedings, in ACM- oder IEEE-Literaturdatenbanken, im OPAC der Bibliothek usw.) an, um ein Gefühl dafür zu entwickeln, was hineingehört (siehe Abschnitt 2.4 Materialsammlung, S. 2).
Was definitiv nicht hineingehört, sind Gliederungsübersichten, chronologische Arbeitsberichte u. dgl.

Abstract

The abstract has to be provided in english as well.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Hintergrund

Diese Masterarbeit ist in Kooperation mit der adesso SE entstanden. adesso ist ein deutsches IT-Beratungs- und Dienstleistungsunternehmen, welches 1997 gegründet worden ist mit Hauptsitz in Dortmund. Seit der Gründung ist die Firma sehr kontinuierlich gewachsen. Mittlerweile sind über 5.800 Mitarbeiter an 44 Standorten in 10 verschiedenen europäischen Ländern hier beschäftigt.

Über die Jahre sind nicht nur die Mitarbeiterzahlen gestiegen, sondern auch die Menge an abgedeckten Branchen, in denen das Unternehmen tätig ist. In den letzten Jahren sind zunehmend Projekte im E-Commerce-Sektor umgesetzt worden. Mit Beginn von 2022 kümmert sich eine eigene Abteilung des Unternehmens explizit um das Thema E-Commerce und Retail.

In diesen Bereich fällt die Programmierung und Betreuung von Online-Shops sowie das umfangreiche Thema des Product-Information-Managements (PIM). Dabei geht es um das Verwalten und Aufbereiten von Produktdaten für verschiedene Anwendungen wie Warenhaltung, Marketing oder Bestellabwicklung. adesso erwägt in Zukunft ein eigenes Product-Information-Management-System (PIM-System) zu entwickeln. In Vorbereitung dessen steht die Evaluierung der Machbarkeit verschiedener Ansätze und Ideen rund um das Product-Information-Management an. Dies ist auch der Aufhänger für diese Masterarbeit.

1.2 Themenfindung

Die initiale Idee bestand darin zu überprüfen, ob es mögliche Anwendungen des Verfahrens der Clusteranalyse im Zusammenhang mit dem Product-Information-Management gibt. Beim Clustering handelt es sich um die Einteilung von Objekten in Gruppen (sog. Cluster), sodass sich die Objekte in der gleichen Gruppe ähnlicher sind als Objekte in den anderen Gruppen. [King15, Kap. 1.1 What Is a Cluster?]

In einem ersten Brainstorming sind verschiedene Anwendungen der Clusteranalyse diskutiert worden. Es folgte eine intensive Literaturrecherche zu den besprochenen Anwendungen, die im Folgenden kurz diskutiert werden.

Die erste Idee war das automatische Kategorisieren von Produkten. Zu dieser Anwendung gibt es kaum Literatur – lediglich einige hoch-experimentelle Ansätze. Z.B. nutzten Chen et al. [ChZhYi19] ein neuronales Netz zum Vorschlagen von Produktkategorien zu Produkten bestehend aus Titel und Beschreibung.

Eine weitere Idee war die Nutzung der Clusteranalyse zur Entwicklung eines Produktempfehlungsalgorithmus. Zu diesem Thema gibt es ganze Reihe an Arbeiten:

- Oh und Kim [OhKi19] implementierten einen Empfehlungsalgorithmus für Cloud Computing-Angebote. Die verschiedenen Anbieter wurden nach den Anforderung der auszuführenden Applikation geclustert, um so das am besten passende Angebot zu finden.
- Cui [Cui21] verbesserte die Klick- und Kaufraten in einem Online-Shop durch die Kombination von Assoziationsregeln mit Fuzzy Clustering. Dadurch konnten passende Warenkörbe schneller identifiziert und vorgeschlagen werden.
- Zuletzt sei die Arbeit von Kumar et al. erwähnt [KRRT01]. Sie näherten sich mit einem möglichst allgemeinen mathematischen Modell an das Thema. Auf dieser Basis identifizierten sie Faktoren, die ein Empfehlungsalgorithmus erfüllen sollte. Werden diese befolgt, so können bereits mit wenigen Daten über das Nutzerverhalten effektive Empfehlungen gegeben werden. Das Clustern der Produkte beschleunigt diesen Prozess, da Erkenntnisse zu einem Produkt auf die gesamte Gruppe übertragbar sind.

Die letzte Idee beschäftigte sich mit der Suche von Produkten. Ein sehr spannendes Paper hierzu verfassten Kou und Lou [KoLo12]. Sie verbesserten die Klickraten einer Websuchmaschine durch die Nutzung von Clustering. Ihr Algorithmus nutzte die am meisten geklickten Webseiten zu jedem Suchbegriff als initiale Schwerpunkte für die Cluster. Anschließend sind die übrigen Webseiten rund um diese Schwerpunkte geclustert worden. Dadurch konnten neu hinzugekommene Seiten direkt in die Suche integriert werden. Außerdem erhöhte sich die Relevanz der gefundenen Suchergebnisse.

Davon abgesehen gibt es für diesen Bereich kaum Arbeiten, in denen Suchfunktionen mit Clustering kombiniert worden sind. Das liegt sicherlich daran, dass klassische probabilistische Suchalgorithmen (der bekannteste ist der BM25 [RoZa09]) sehr gute Ergebnisse liefern. Ansätze wie von Kou und Lou sind in dem Zusammenhang eher als experimentell anzusehen.

1.3 Kernfrage und Ablauf

Parallel zu der beschriebenen Recherche ist ebenfalls ein Blick in typische PIM-System wie z.B. Akeneo-PIM geworfen worden. Ziel war es zu verstehen, wie die Produktdaten in solchen System typischerweise vorliegen. Dies ist nötig, um einzuschätzen, ob und wie das Clustering der Produktdaten in solchen Systemen möglich ist.

Zum einen weisen die Produktdaten einen hohen Grad an Struktur auf. Die Attribute sind fest definiert. Umfangreiche Constraints sorgen für die Einhaltung des definierten Schemas. Zum anderen kommen sehr verschiedenartige Datentypen vor wie z.B. Textfelder, Einfach- und Mehrfachauswahl, numerische Daten mit verschiedenen Einheiten. Zudem sind viele Attribute lokalisierbar und weisen je nach Sprache andere Werte auf. Die klassische Clusteranalyse arbeitet allerdings nur mit numerischen Daten [King15, Kap. 1.2 Capturing the Clusters]. Somit war abzusehen, dass das Clustern selbst kein einfacher Prozess sein wird.

Dadurch verschob sich der Fokus der Arbeit. Bevor mögliche Anwendungen der Clusteranalyse evaluiert werden, stellt sich zuerst die Frage, ob so vielfältige Produktdaten überhaupt geclustert werden können. Die Kernfrage der Arbeit lautet also:

Ist das effektive Clustern hoch-komplexer Produktdaten möglich – und wenn ja, wie?

Diese Fragestellung war der Ausgangspunkt weiterer Recherchen, Experimente und Versuche. Sie werden in den folgenden Kapiteln erläutert.

Kapitel 2 gibt einen Überblick über die wissenschaftlichen Grundlagen und Erkenntnisse des Themenfeldes.

Auf dieser Basis wird in *Kapitel 3* eine Konzeption dargelegt, wie das Clustern durchgeführt werden kann. Ebenso werden Versuche geplant, die zur Überprüfung des Konzeptes dienen.

Kapitel 4 beschreibt die Umsetzung des dargelegten Konzeptes. Thema dabei sind die Implementierung nötiger Software-Komponenten, Tools und Techniken, die zur Überprüfung der Fragestellung angewendet worden sind.

Die Auswertung und Evaluation der Versuche erfolgt in *Kapitel 5*.

Im finalen *Kapitel 6* wird das Fazit gezogen, ob die Kernfrage beantwortet werden kann. Ebenso wird ein Ausblick für weitere Versuche und Fragestellungen in der Zukunft gegeben.

2 Clusteranalyse

2.1 Begriff und Einordnung

King definiert die *Clusteranalyse* als die "[...] Generierung eines Klassifizierungsschemas, welches Individuen in eine feste Anzahl an Gruppen einteilt, so dass sich die Individuen innerhalb einer Gruppe auf eine Art und Weise ähnlich sind und unähnlich denen in anderen Gruppen" [King15, Kap. 1.1 What Is a Cluster?]. Diese Gruppen werden auch als Cluster bezeichnet.

Dieser Prozess des Clusterings ist eine Methode des *unüberwachten Lernens (unsupervised learning)* – einem Teilgebiet des maschinellen Lernens. Papp et al. schreiben dazu: "Machine Learning beschäftigt sich mit der Entwicklung von Methoden, die Modelle von Zusammenhängen aus Daten erlernen, anstatt sie *per Hand* zu implementieren" [PWMO19, Kap. 5 Statistik-Grundlagen]. Ferner geben sie an, dass die Unterschiede zur Statistik fließend sind [PWMO19, Kap. 5 Statistik-Grundlagen]. Unüberwachtes Lernen bedeutet dabei, dass die verwendeten Daten nicht im Vorhinein gekennzeichnet sind (Unlabeled Data). Stattdessen werden Ähnlichkeiten und Muster in den Daten selbst gesucht ohne eine vorgegebene Zielgröße. Häufig dienen diese Analysen als erste Schritte der *Data Exploration*, aus denen im Anschluss neue Erkenntnisse und Anwendungen abgeleitet werden. [PWMO19, Kap. 5.2.3 Unüberwachtes Lernen]

Allgemein wird die Clusteranalyse als eine Form des Data Minings gesehen. Laut Bissantz und Hagedorn "beschreibt [Data Mining] die Extraktion implizit vorhandenen, nicht trivialen und nützlichen Wissens aus großen, dynamischen, relativ komplex strukturierten Datenbeständen" [BiHa09]. Mit "Wissen" meinen sie dabei in den Daten implizit enthaltene Muster, welche für den Anwender interessant sind und mit einer hinreichenden Wahrscheinlichkeit tatsächlich in den Daten existieren. [BiHa09]

2.2 Notation

Bevor konkrete Algorithmen und Verfahren der Clusteranalyse vorgestellt werden, müssen einige grundlegende Sachverhalte und ihre mathematische Definition geklärt werden:

Die Objekte, welche es zu clustern gilt, werden in dieser Arbeit auch als *Datenpunkte* oder *Produkte* (da dies die Anwendung im Praxisteil ist) bezeichnet. Mathematisch werden sie i.d.R. mittels x definiert. Diese Produkte bestehen aus einer festen Menge an Attributen (z.B. Farbe, Gewicht etc.). Mathematisch ist so ein Datenpunkt als ein Vektor definiert und jedes Element im Vektor steht für die Ausprägung eines spezifischen Attributes. Mittels Superskript werden spezifische Attribute eines Datenpunktes angesprochen. x^i bezeichnet also das i-te Attribut von x.

$$x = (a|a \text{ is attribute}) \tag{2.1}$$

Die Produkte oder Datenpunkte sind Teil eines *Datensets* X. Hierbei handelt es sich um eine simple Menge dieser Datenpunkte $x \in X$. Mittels Subskript werden einzelne Punkte des Sets angesprochen (z.B. x_1, x_2 oder x_i). Wenn nicht anders angegeben, gilt $n = |X|$.

Als *Cluster* C wird nun eine Teilgruppe aus dem Datenset bezeichnet: $C \subset X$. Allerdings können Cluster auch "hierarchisch" strukturiert sein. D.h. ein Cluster kann sowohl Datenpunkte als auch andere Cluster enthalten.

$$C = \{e|e = C_i \vee e \in X\} \tag{2.2}$$

Die *Menge aller Cluster* wird mit $K = \{C_1, C_2, ...\}$ bezeichnet. Dabei wird die Anzahl der Cluster mit $k = |K|$ angegeben.

Schlussendlich gibt es noch das *Clustering-Ergebnis* Y. Dies ist ein Vektor der Länge n, welches jeden Datenpunkt im Set einem entsprechenden Cluster zuordnet.

$$Y = (l|l_i = \text{ label for } x_i \in X) \tag{2.3}$$

2.3 Distanz- und Ähnlichkeitsmaße

2.3.1 Definition

Die Forderung, dass Objekte in gleichen Clustern sich "ähnlich" sein sollen und unähnlich zu den Objekten in anderen Clustern, muss in irgendeiner Form quantifiziert werden. Dies erfolgt über die Berechnung der "Nähe" (engl. proximity) der Objekte zueinander. Dazu werden sog. Abstands- bzw. Distanzmaße verwendet. [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

Mathematisch wird dieses Distanzmaß mittels der Funktion $d(x_1, x_2)$ (engl. distance) definiert, welche die Distanz zwischen zwei Datenpunkten x_1 und x_2 als skalaren Wert zurückgibt. Zusätzlich geben Kaufmann und Rousseeuw folgende Eigenschaften für d an [KaRo09, Kap. 1.2.1 Interval-Scaled Variables]:

Tabelle 2.1: Eigenschaften der Abstandfunktion d

1.	$d(x_1, x_2) \geq 0$	Distanzen sind stets positiv
2.	$d(x_1, x_1) = 0$	zwei gleiche Objekte haben immer einen Abstand von 0
3.	$d(x_1, x_2) = d(x_2, x_1)$	die Distanzfunktion ist kommutativ bzw. symmetrisch
4.	$d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$	Distanzen geben stets den kürzesten Weg an

Anstatt des Abstandes kann alternativ auch die Ähnlichkeit zweier Objekte berechnet werden. Solche Ähnlichkeitsmaßes $s(x_1, x_2)$ (engl. similarity) sind häufig im Interval $[0; 1]$ angegeben, wobei die 1 maximale Ähnlichkeit angibt. Ist nun eine entsprechende Distanzfunktion z.B. durch Normalisierung ebenfalls im Interval $[0; 1]$ definiert, so können Distanzen und Ähnlichkeiten einfach ineinander überführt werden [KaRo09, Kap. 1.2.3 Similarities]:

$$d(x_1, x_2) = 1 - s(x_1, x_2) \tag{2.4}$$

Distanzen und Ähnlichkeiten sind dadurch beliebig austauschbar. Daher ist dieses Vorgehen stets empfehlenswert.

Es existieren eine Vielzahl an Distanz- und Ähnlichkeitsmaßen. Welche zur Anwendung kommen, hängt maßgeblich von den Attributen und vor allem den Attribut-Typen ab, aus denen die Datenpunkte bestehen. [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

2.3.2 Numerische Attribute

Numerische Attribute sind im Allgemeinen alle Arten von (rationale) Zahlen [KaRo09, Kap. 1.2 Types of Data and How to Handle Them] mit stetigen (engl. continuous) Werten [Huan98]. Es kann sich dabei sowohl um Intervall- als auch um Verhältnisskalen handeln (engl. interval and ratio data). [Bosl12, Kap. 1 Basic Concepts of Measurement]

Datenpunkte, die nur aus numerischen Attributen bestehen, sind die klassischste Form von Daten, mit denen Clusteranalyse durchgeführt wird. Die meisten Algorithmen und Verfahren sind damit speziell auf diesen Attribut-Typ ausgelegt. [King15, Kap. 1.2 Capturing the Clusters]

Minkowski-Familie

Der Abstand zwischen reinen numerischen Vektoren wird mittels Maßen aus der sog. Minkowski-Familie berechnet. Das älteste dieser Metriken ist der “euklidische Abstand” (auch pythagoreische Metrik genannt), welcher den Abstand zwischen zwei Punkten über eine gerade Linie zwischen diesen beiden bestimmt. Im 19. Jahrhundert verallgemeinerte Hermann Minkowski dieses Maß zu einer ganzen Familie. [Cha07]

Tabelle 2.2: Übersicht über die gängigen Maße aus der Minkowski-Familie [Cha07]

Name	p-Norm	Formel
allgemeine Form		$d(x_1, x_2) = \sqrt[p]{\sum_{i=1}^{n} \lvert x_1^i - x_2^i \rvert^p}$
Manhattan-Abstand	$p = 1$	$d(x_1, x_2) = \sum_{i=1}^{n} \lvert x_1^i - x_2^i \rvert$
euklidischer Abstand	$p = 2$	$d(x_1, x_2) = \sqrt{\sum_{i=1}^{n} \lvert x_1^i - x_2^i \rvert^2}$
Chebyshev-Abstand	$p = \infty$	$d(x_1, x_2) = \max \lvert x_1^i - x_2^i \rvert$

Die allgemeine Form ist über den Ausdruck p parametrisiert. Die konkreten Vertreter der Familie kommen nun durch ein spezifisch gewähltes p zustande. Daher wird bei diesen Maßen

auch von der sog. p-Norm gesprochen. Eine höhere p-Norm steht dabei im Allgemeinen für eine robustere bzw. eindeutigere Bestimmung des Abstandes. [King15, Kap. 12.3 Which Proximity Measure Should Be Used?]

Dies sind bei weitem nicht alle Arten von Maßen für numerische Attribute. Cha [Cha07] hat eine Vielzahl solcher Maße zusammengetragen, analysiert und miteinander verglichen. Dabei kam er zu folgenden Erkenntnissen:

Die meisten dieser Maße sind Abwandlungen eines Vertreters aus der Minkowski-Familie bzw. lassen sich auf einen solchen abbilden (z.B. basieren einige Maße auf der Pearson-Korrelation, welche ihrerseits wiederum auf dem euklidischen Abstand beruht). In seinen anschließenden Versuchen zeigte er, dass Maße, die auf einander basieren, auch stets ähnliche Abstände liefern. [Cha07]

Daraus lässt sich ableiten, dass die Vertreter der Minkowski-Familie ausreichend sind, um den Abstand numerischer Vektoren zu bestimmen.

Vorverarbeitung

Häufig macht eine Vorverarbeitung der numerischen Attribute Sinn, um verschiedene Arten von Problemen zu vermeiden. [KaRo09, Kap. 1.2.1 Interval-Scaled Variables]

Logarithmische Attribute Die Abstandsmaße der Minkowski-Familie gehen i.d.R. von linearen Verhältnissen der Zahlen zueinander aus (Punkte mit einem Abstand von 2 sind doppelt so weit entfernt, wie Punkte mit einem Abstand von 1). Es kann aber vorkommen, dass die vorliegenden Daten eigentlich einen logarithmischen Zusammenhang aufweisen (der Abstand von 10 zu 20 ist exakt gleichbedeutend mit dem Abstand von 100 zu 200). Damit die Abstandsmaße richtig arbeiten können, sollten solche Attribute vorher in eine lineare Abbildung überführt werden [KaRo09, Kap. 1.2.5 Nominal, Ordinal, and Ratio Variables]. Dies wird folgendermaßen bewerkstelligt:

$$x'^i = \log x^i \tag{2.5}$$

Standardisierung Weitere Probleme können entstehen, wenn die absoluten Werte in den verschiedenen Dimensionen der Vektoren stark voneinander abweichen. Angenommen das Datenset besteht aus zwei-dimensionalen Vektoren. Die Werte in der ersten Dimension liegen im Bereich $[0; 1]$ und in der zweiten im Bereich $[100; 200]$. Dadurch würden die Unterschiede zwischen den Werten in der zweiten Dimension viel stärker ins Gewicht fallen, da hier die Differenzen zwischen den Werten bis zu 100 Mal höher ausfallen. Um eine Gleichgewichtung zu erzeugen, sollten diese Attribute jeweils auf die gleiche Art standardisiert werden. [KaRo09, Kap. 1.2.1 Interval-Scaled Variables]

Milligan und Cooper [MiCo88] haben eine Vielzahl solcher Methoden der Standardisierung analysiert und experimentell miteinander verglichen.

$$x'^i = \frac{x^i - \text{avg } X^i}{\sigma(X^i)} \tag{2.6}$$

Die Formel 2.6 zeigt die Standardisierung mittels des klassischen z-Wertes. Die auftretenden Werte werden dadurch auf die 0 zentriert und mit einer Varianz von 1 drumherum verteilt. Dieses Maß ist besonders anfällig für Outliers (stark von der Hauptmenge abweichende Werte). In Milligan und Coopers Versuchen schnitt es daher eher mittelmäßig ab. [MiCo88]

$$x'^i = \frac{x^i - \text{avg } X^i}{\text{mad } (X^i)} \tag{2.7}$$

$$mad(X^i) = \frac{1}{n} \sum_j |x_j^i - \text{avg } X^i| \tag{2.8}$$

Kaufmann und Rousseeuw empfehlen eine modifizierte Form des z-Wertes nach Hartigan [Hart75]. Statt durch die Standardabweichung $\sigma(X^i)$ zu teilen, wird durch die mittlere absolute Abweichung (engl. *mean absolute deviation*) $mad(X^i)$ dividiert. Dieses Vorgehen soll den Einfluss von Outliern dämpfen. [KaRo09, Kap. 1.2.1 Interval-Scaled Variables]

$$x'^i = \frac{x^i - \min X^i}{\max X^i - \min X^i} \tag{2.9}$$

Eine weitere Methode ist die Skalierung aller Werte auf das Interval $[0; 1]$. Obwohl Outliers bei dieser Methode ebenfalls die restlichen Werte in einen kleineren Bereich komprimieren, war dieser Ansatz in Milligan und Coopers Versuchen äußerst robust und lieferte über alle Versuche akzeptable Ergebnisse. [MiCo88]

Kaufmann und Rousseeuw geben zu bedenken, dass die unterschiedlichen Gewichtungen der Werte echte Zusammenhänge der realen Welt widerspiegeln könnten. Durch Standardisierung gehen diese entsprechend verloren. Weiterhin ist sogar denkbar, bewusst einige Attribute stärker zu gewichten als anderen (indem sie mit entsprechenden Gewichten multipliziert werden). Dies sind allerdings eher speziellere Fälle, in denen im Vorfeld bereits Domänenwissen zum Datenset vorliegen muss. [KaRo09, Kap. 1.2.1 Interval-Scaled Variables]

2.3.3 Kategorische Attribute

Kategorische Attribute bilden die zweite große Gruppe von Attribut-Typen. Sie zeichnen sich dadurch aus, dass als Wertausprägungen Labels verwendet werden, die verschiedene Kategorien repräsentieren. Diese Labels können in Form von Zeichenketten (z.B. “red”, “green”, “blue”), Bool-Werten ($true/false$, Ja/Nein, 0/1) oder auch diskreten Ziffern auftreten. Es ist eine weitere Unterscheidung in die zwei Teilgruppen ordinale und nominale Attribute möglich. [Bosl12, Kap. 5 Categorical Data]

Ordinale Attribute

Lassen sich die vorkommenden Labels eines Attributes in eine sinnvolle Reihenfolge bringen (z.B. “XS”, “S”, “M”, “L”, “XL”), so werden sie als *ordinal* bezeichnet. Wichtig dabei ist, dass nur eine sinnvolle Reihenfolge besteht. Eine Aussage über die Intervalle zwischen den Labels (“XS” zu “S” ist genauso weit wie “M” zu “L”) oder über deren quantitativen

Verhältnisse (3·“XS” = 2·“S”) macht aber keinen Sinn. [Bosl12, Kap. 1 Basic Concepts of Measurement]

Für die Berechnung des Abstandes gibt es grundsätzlich zwei Möglichkeiten: Zum einen können ordinale Attribute genauso wie nominale verarbeitet werden (siehe nächster Abschnitt). Zum anderen kann die Reihenfolge der Labels verwendet werden, um die Labels in numerische Werte umzuwandeln. Anschließend kann der Abstand mittels der Maße für numerische Attribute berechnet werden. [KaRo09, Kap. 1.2.5 Nominal, Ordinal, and Ratio Variables]

Die Umwandlung erfolgt dabei so, dass die Labels entsprechend ihrer natürlichen Reihenfolge beginnend mit 1 nummeriert werden. Der finale numerische Wert berechnet sich dann wie folgt:

$$x'^i = \frac{x^i - 1}{|\{x \in X^i\}| - 1} \tag{2.10}$$

$|\{x \in X^i\}|$ ist dabei die Menge an auftretenden Labels. Dadurch liegen alle Werte erneut im Interval $[0; 1]$, welches in $|\{x \in X^i\}| - 1$ gleichmäßige Schritte unterteilt ist. [KaRo09, Kap. 1.2.5 Nominal, Ordinal, and Ratio Variables]

Nominale Attribute

Die Verarbeitung von nominalen Attributen geschieht über die vorherige Umwandlung in sog. binäre Attribute. Diese Attribute sind dadurch gekennzeichnet, dass sie nur zwei verschiedene Werte annehmen können (i.d.R. 0/1 bzw. $false/true$) [KaRo09, Kap. 1.2.5 Nominal, Ordinal, and Ratio Variables]. Hier gilt es zwei Fälle zu unterscheiden:

Kommen in den Labels eines Attributs maximal zwei verschiedene Werte vor (z.B. Ja/Nein, schwarz/weiß etc.), so werden diese in ein einzelnes binäres Attribut umgewandelt. Solche Attribute werden auch als *symmetrische* binäre Attribute bezeichnet. [KaRo09, Kap. 1.2.4 Binary Variables]

Bei mehr als zwei Wertausprägungen gibt es hingegen wiederum zwei Möglichkeiten:

- Zum einen können die vorkommenden Labels auf genau zwei verschiedene Wertausprägungen komprimiert werden, sodass wieder ein einzelnes binäres Attribut verwendet werden kann. Dieser Vorgang geht aber mit einem Informationsverlust einher. [KaRo09, Kap. 1.2.5 Nominal, Ordinal, and Ratio Variables]
- Zum anderen kann für jedes vorkommende Label ein eigenes binäres Attribut angelegt werden. Das Attribut, welches das tatsächlich vorkommende Label eines Datenpunktes repräsentiert, wird entsprechend auf $true$ gesetzt und alle anderen auf $false$. Diese Umwandlung resultiert in sog. *asymmetrische* binäre Attribute. [KaRo09, Kap. 1.2.4 Binary Variables]

Kennzeichnend für die symmetrischen binäre Attribute ist, dass beide Wertausprägung eine konkrete Eigenschaft des Objektes darstellen (z.B. 0 = schwarz und 1 = weiß). Asymmetrische zeigen hingegen nur das Vorhanden- oder Nicht-Vorhandensein einer bestimmten

Eigenschaft an und müssen stets im Kontext der gesamten Gruppe betrachtet werden. Für beide Arten gibt es jeweils eigene Distanzmaße. [KaRo09, Kap. 1.2.4 Binary Variables]

Simple Matching Das sog. Simple Matching ist für die Bewertung der Ähnlichkeit von Objekten mit ausschließlich symmetrischen binären Attributen geeignet [KaRo09, Kap. 1.2.4 Binary Variables]. Es berechnet sich wie folgt:

$$s(x_1, x_2) = \frac{|x_1 \cap x_2| + |\bar{x_1} \cap \bar{x_2}|}{|x_1 \cup \bar{x_1} \cup x_2 \cup \bar{x_2}|} \tag{2.11}$$

Verglichen werden also die Menge an Attributen mit der gleichen Wertausprägung in beiden Objekten (beide $true$: $|x_1 \cap x_2|$ und beide $false$: $|\bar{x_1} \cap \bar{x_2}|$) im Verhältnis zur Anzahl aller Attribute. Es handelt sich damit um ein Ähnlichkeitsmaß im Interval $[0; 1]$, wobei die 1 für perfekte Übereinstimmung steht. [KaRo09, Kap. 1.2.4 Binary Variables]

Es gibt von diesem Verfahren eine Reihe von Abwandlungen, welche die Ausprägungen bestimmter Wertekombinationen gesondert gewichten. Alle diese Varianten basieren im Kern auf dem Simple Matching und liefern stets ähnliche Ergebnisse. [KaRo09, Kap. 1.2.4 Binary Variables]

Jaccard-Koeffizient Für die Bewertung der Ähnlichkeit von Objekten mit asymmetrischen binären Attributen eignet sich hingegen der Jaccard-Koeffizient besser [KaRo09, Kap. 1.2.4 Binary Variables]. Er berechnet sich wie folgt:

$$s(x_1, x_2) = \frac{|x_1 \cap x_2|}{|x_1 \cup x_2|} \tag{2.12}$$

Im Gegensatz zum Simple Matching werden hier alle Attribute, in denen beide Objekte den Wert $false$ aufweisen, ignoriert. Die Übereinstimmung wird nur mit den Attributen berechnet, wo wenigstens eines von beiden Objekten den Wert $true$ aufweist. [KaRo09, Kap. 1.2.4 Binary Variables]

Hintergrund: Angenommen ein kategorisches Attribut weist im Datenset bis zu drei verschiedene Labels auf ("red", "green" und "blue"). x_1 hat das Label "red" und x_2 hat "blue". Nach der Umwandlung in binäre Werte wäre $x_1 = (1, 0, 0)$ und $x_2 = (0, 0, 1)$. Das Simple Matching würde nun eine Ähnlichkeit von $\frac{1}{3}$ berechnen, da beiden Objekten gemeinsam ist, dass sie *nicht "green"* sind. Bei asymmetrischen binären Attributen sind aber zwangsläufig mit steigender Anzahl an vorkommenden Labels die meisten Werte 0 bzw. $false$. Dies treibt das Ergebnis vom Simple Matching künstlich in die Höhe, macht aber inhaltlich in diesem Kontext keinen Sinn. [KaRo09, Kap. 1.2.4 Binary Variables]

2.3.4 String-Attribute

Manche Autoren (z.B. [RaRa11]) definieren eine weitere Gruppe sog. String-Attribute. Streng genommen sind dies ebenfalls nominale bzw. kategorische Attribute. Sie zeichnen sich aber durch eine hohe Variation an Werten aus, sodass häufig keine einzige Wertausprägung mehr als einmal vorkommt. Ein klassisches Beispiel wäre der Produkttitel. Es wird i.d.R.

keine zwei Produkte mit dem gleichen Titel geben. Dies führt die Betrachtung dieser Attribute als “kategorisch” ad-absurdum, wenn jeder Datenpunkt seine eigene individuelle Kategorie definiert. Sollen diese Werte dennoch auf ihre Ähnlichkeit geprüft werden, so gibt es eine Reihe verschiedener Vorgehensweisen.

String-Metrics

Eine Methode besteht aus dem Vergleich übereinstimmender Buchstaben zwischen zwei String-Werten. Dazu existieren eine Reihe sog. “String-Metriken” von verschiedenen Autoren. [CRF03]

Hamming [Hamm50] entwickelte 1950 ein Verfahren zur Fehlererkennung binärer Codes. Es bildet die Grundlage des sog. *Hamming-Abstands*, welches sich später in abgewandelter Form im Bereich der String-Metrics als Distanzmaß etablierte [RaRa11]. Der Hamming-Abstand ist die Anzahl an Buchstaben zwischen zwei Zeichenketten gleicher Länge, welche **nicht** identisch sind. Er kann nur zum Vergleich von Strings mit exakt identischer Anzahl an Buchstaben verwendet werden und liefert Abstände von 0 bis n (Anzahl an Buchstaben im jeweiligen String).

Ein Verfahren, welches auch für Strings unterschiedlicher Länge geeignet ist, ist der *Levensthein-Abstand*. Zur Berechnung wird gezählt, wie viele Buchstaben eingefügt, entfernt oder umgewandelt werden müssen, um einen String in den anderen umzuwandeln. Dabei wird stets die kleinstmögliche Zahl an Ersetzungen bevorzugt. Die Ergebnisse liegen damit ebenfalls zwischen 0 (geringster Abstand, da kein einziger Schritt zur Umwandlung nötig ist) und n, was die Anzahl an Buchstaben des längeren Strings ist. Der Nachteil dieses Verfahrens ist die recht aufwendige Berechnung, da stets mehrere Arten der Buchstabentransformationen durchgerechnet werden müssen, um die kürzeste Kette an Transformationen zu finden. [RaRa11]

Mit deutlich weniger Rechenaufwand kommt die *Jaro-Metrik* [Jaro95] aus. Sie berechnet sich aus der Menge an Buchstaben, welche in beiden Wörtern in der gleichen Reihenfolge vorkommen, zu der Menge aller Buchstaben je Wort (Dies ist eine verkürzte Darstellung. Siehe [Jaro95] oder [CRF03] für die genaue Berechnung). Das Ergebnis gibt die Ähnlichkeit beider Strings zwischen 0 (komplett unähnlich) und 1 (maximale Übereinstimmung). [CRF03]

Eine Variante des eben erklärten Maßes ist die *Jaro-Winkler-Metrik*, welcher eine etwas höhere Ähnlichkeit feststellt, wenn die ersten Buchstaben (Prefix) identisch sind. [CRF03]

In den Versuchen von Cohen [CRF03] lieferte der Jaro-Winkler-Abstand die besten Ergebnisse bei der Verwendung für das Clustering.

Tokenization

Strings-Metrics sind eher für den Vergleich kürzerer Zeichenketten geeignet. Für ganze Textzeilen und -blöcke werden eher Verfahren aus dem Document Retrieval angewandt. [StKaKu00]

Wesentlich im Document Retrieval ist die Zerlegung eines Textes in die einzelnen Wörter, aus denen sich der Text zusammensetzt. Da in vielen Sprachen das gleiche Wort je nach Fall

verschiedene Endungen (engl. stemming) aufweisen kann, werden diese Endungen häufig entfernt (z.B. mit Verfahren wie Porters Stemming Algorithmus). Ebenso weisen alle Sprachen kleine Füllwörter (z.B. Artikel, Bindewörter etc.) auf, die sehr häufig in eigentlich jedem Text vorkommen. Auch diese werden i.d.R. entfernt (engl. stop-word removal), da sie keine Relevanz für die Unterscheidung der Texte haben. Das Ergebnis ist eine Menge aus den Grundwörtern eines Textes, welche auch "Tokens" genannt werden. [StKaKu00; und CRF03]

Nach dieser "Tokenization" können zwei Texte z.B. auf ihre Ähnlichkeit überprüft werden, indem man die auftretenden Token beider Texte miteinander vergleicht. Für die Berechnung können bereits erklärte Metriken wie der Jaccard-Koeffizient eingesetzt werden. Die Tokens repräsentieren dann "Kategorien" oder Themenfelder der Texte. [CRF03]

Eine andere Möglichkeit besteht darin, die Tokens in numerische Vektoren umzuwandeln. Jedes Element im Vektor repräsentiert eines der Token und der Wert des Elementes entspricht der Anzahl, wie häufig dieses Token im ursprünglichen Text vorkommt. Dieser Vorgang wird auch als Transformation der Texte in das "Vector-Space-Model" bezeichnet. [StKaKu00]

Diese numerischen Vektoren können nun wie numerische Attribute verarbeitet werden. Allerdings existieren auch gesonderte Metriken speziell für den Bereich des Document Retrieval. Bspw. wird statt des euklidischen Abstandes häufig der "cosine"-Abstand zur Bestimmung der Distanz verwendet. [StKaKu00]

$$cosine(x_1, x_2) = \frac{|x_1 \cdot x_2|}{|x_1| \cdot |x_2|} \tag{2.13}$$

2.3.5 Gemischte Attribute

In einem Datenset können nicht nur Attribute des gleichen Types (z.B. ausschließlich numerisch) vorkommen. Gibt es Attribute mit unterschiedlichen Typen, so spricht man von gemischten Daten oder gemischten Attributen. Der Umgang mit diesen ist sehr vielfältig. [KaRo09, Kap. 1.2.6 Mixed Variables]

Umwandlung aller Attribute in den selben Typ

Eine der einfachsten Varianten besteht darin, alle Attribute in den gleichen Typ umzuwandeln. [KaRo09, Kap. 1.2.6 Mixed Variables]

Umwandlung in numerische Werte Für ordinale Attribute ist diese Umwandlung bereits beschrieben worden. Nominale Attribute können in eine künstliche Reihenfolge gebracht werden, wodurch sie ihren ordinalen Brüdern gleichen und entsprechend umgewandelt werden können. Diese Transformation verzerrt u.U. die Daten, da nun z.B. "rote" und "grüne" Produkte als ähnlicher betrachtet werden als "rote" und "blaue" (aufgrund der künstlichen Reihenfolge). Nichtsdestotrotz kann dieses Verfahren gute Ergebnisse produzieren. [KaRo09, Kap. 1.2.6 Mixed Variables]

Die Transformation von String-Attributen in numerische Vektoren ist ebenfalls bereits dargelegt worden.

Umwandlung in kategorische Werte Auch der umgekehrte Weg ist denkbar. Ordinale Attribute können als nominal betrachtet und verarbeitet werden. Ebenso können numerische Werte durch Diskretisierung in feste Gruppen oder Bänder eingeteilt werden (z.B. in Alterspannen "unter 18", "18-35" usw.). [KaRo09, Kap. 1.2.6 Mixed Variables]

Diese Umwandlung geht aber mit einem massiven Verlust an Informationen einher. Bspw. brachte die Diskretisierung numerischer Attribute in den Versuchen von Milligan und Cooper die schlechtesten Clustering-Ergebnisse. [MiCo88]

Subspace-Clustering

Ein ganz anderer Ansatz besteht darin, das Clustering mehrmals durchzuführen, jeweils mit jedem Attribut-Typen separat. Anschließend werden die Ergebnisse verglichen und entsprechende Erkenntnisse abgeleitet. Allerdings ist hier der Interpretationsspielraum sehr groß, sodass i.d.R. andere Arten des Vorgehens empfehlenswerter sind. [KaRo09, Kap. 1.2.6 Mixed Variables]

Dieser Ansatz wird auch als *Subspace-Clustering* bezeichnet und kann sehr weit ausgeführt werden. So werden teilweise mehrmalige Clusterings mit verschiedenen Kombinationen an Attributen (beliebigen Types) durchgeführt und bewertet. Solche Ansätze sind sehr aufwendig sowohl in Sachen Laufzeit als auch in der Interpretation der eigentlichen Ergebnisse. [JiCh17]

Kombinierte Distanzfunktion

Schließlich besteht die Möglichkeit, verschiedene Distanzmaße zu kombinieren. Die verschiedenen Attribute also mit einer Metrik ihres Types zu verrechnen und diese Ergebnisse anschließend zu addieren. [KaRo09, Kap. 1.2.6 Mixed Variables]

Ein geläufiges Beispiel ist das kombinierte Distanzmaß des K-Prototypes-Algorithmus [Huan98], welches wie folgt definiert ist:

$$d(x_1, x_2) = d_{num}(x_1^{num}, x_2^{num}) + w \cdot d_{cat}(x_1^{cat}, x_2^{cat}) \tag{2.14}$$

Dieser Algorithmus kann numerische und kategorische Attribute gemeinsam verarbeiten. Die spezifischen Distanzmaße für jeden Attribut-Typ sind dabei frei wählbar (im Original: euklidisch für die numerischen und Simple Matching für die kategorischen). Der Faktor w soll Unterschiede in der Gewichtung der Distanzmaße ausgleichen. So hat der euklidische Abstand umso höhere Werte, je mehr numerische Attribute vorkommen. Das Simple Matching hingegen liegt immer im Interval $[0; 1]$. [Huan98]

2.4 Clustering-Verfahren

2.4.1 Partitionierendes Clustering

Überblick

Partitionierende Verfahren betrachten die Clusterzuteilung als ein Minimierungsproblem. Auf verschiedene Arten starten die jeweiligen Verfahren mit einer initialen Gruppierung der Datenpunkte. Anschließend werden die Clusterzuteilungen wiederholt verändert bis ein lokales Optimum erreicht ist [King15, Kap. 4.1 Partition Clustering – Introduction]. Wichtig zu erwähnen ist dabei, dass jeder Datenpunkt nur genau einem Cluster zugeordnet werden kann [KaRo09, Kap. 1.3.1 Partitioning Methods]. Die konkreten Verfahren unterscheiden sich dabei in der Art der Initialisierung der Cluster, dem genauen Ablauf der Minimierung sowie in der Definition, wann ein Clustering "besseres" ist [King15, Kap. 4.1 Partition Clustering – Introduction]. Ebenso gibt es spezielle Verfahren für das Clustern von Daten, welche nicht ausschließlich numerisch sind. [Huan98]

Kennzeichnend für diese Verfahren ist außerdem, dass die Anzahl der erwarteten Cluster im Vorfeld bekannt sein muss. Dies ist zum Teil problematisch, wenn kaum Informationen zum Datenset vorliegen. Eine mögliche Lösung ist das mehrmalige Clustern mit verschiedenen Werten für die Clusteranzahl. Anschließend kann mittels verschiedener Metriken (siehe Abschnitt Cluster-Validität) eine adäquate Clusteranzahl aus den Ergebnissen ausgewählt werden. Ebenso könnte zu erst ein hierarchisches Clustering zur Bestimmung der Clusterzahl durchgeführt werden. [King15, Kap. 4.1 Partition Clustering – Introduction]

Diese Verfahren neigen dazu in lokalen Minima während des Clusterings hängen zu bleiben. Deswegen ist es empfehlenswert, das jeweilige Verfahren mehrmals mit unterschiedlichen Start-Partitionierungen durchzuführen und anschließend das "beste" Ergebnis aus diesen Durchläufen auszuwählen. Trotz der mehrfachen Durchläufe sind partitionierende Verfahren allgemein recht effizient in ihrer Laufzeit. Die Komplexität beträgt $\mathcal{O}(n \cdot k \cdot l)$. n ist die Anzahl an Datenpunkten, k die Menge an gesuchten Clustern und l die mehrmalige Wiederholung des Verfahrens. k und l sind typischerweise recht kleine Werte, die auch mit steigender Anzahl von Datenpunkten im Datenset nicht zunehmen. Daher handelt es sich unterm Strich um eine lineare Laufzeit von $\mathcal{O}(n)$. [Huan98]

K-Means

K-Means ist das bekannteste partitionierende Clustering-Verfahren und gilt häufig als Inbegriff der Clusteranalyse selbst. [Huan98]

Jedes Cluster wird in diesem Verfahren durch einen Mittelpunkt c repräsentiert. Dieser Mittelpunkt oder Centroid ist kein tatsächlicher Punkt des Datensets [StKaKu00], sondern wird aus den Punkten des Clusters berechnet. Konkret wird für jedes einzelne Attribut der Durchschnitt (engl. mean) aus den Werten der Cluster-Mitglieder für das jeweilige Attribute gebildet [King15, Kap. 4.5 K-Means Algorithm]:

$$c(C) = \frac{1}{|C|} \sum_{x_i \in C} x_i \tag{2.15}$$

Durch das Verändern der Clusterzuordnungen ändern sich entsprechend auch die Mittelpunkte der Cluster. Ziel dieser Verschiebung ist, dass die Summe der Abstände der Datenpunkte zu ihrem jeweiligen Mittelpunkt minimal wird. Der Abstand wird mit einem Distanzmaß für numerische Attribute (meist der euklidische Abstand) berechnet. [King15, Kap. 4.5 K-Means Algorithm]

Dieses Verfahren funktioniert also ausschließlich mit numerischen Attributen, da nur hier aus den Werten ein Durchschnitt berechnet werden kann. [Huan98]

Steinbach et al. [StKaKu00] beschreiben den typischen Ablauf des Verfahrens folgendermaßen:

1. Wahl von k Punkten aus dem Datenset als initiale Mittelpunkte
2. Zuordnung aller Datenpunkte des Datensets zum jeweils nächstgelegen Mittelpunkt (mit dem geringsten Abstand)
3. Neuberechnung aller Mittelpunkte aus den Durchschnitten der zugeordneten Datenpunkte (wodurch sich diese verschieben)
4. Wiederholung der Schritte 2 und 3; solange bis sich die Clusterzuordnungen nicht mehr verändern oder ein Höchstlimit an Iterationen überschritten ist

Die Wahl der initialen Punkte kann z.B. durch die Wahl der k ersten oder letzten Punkte im Datenset geschehen. Ebenso möglich ist die Wahl jedes $\frac{n}{k}$-ten Punktes. Am geläufigsten ist wohl die komplett zufällige Wahl von Punkten aus dem Set. [King15, Kap. 4.3 The Initial Partition]

Zu diesem Basis-Verfahren gibt es eine große Vielfalt an Varianten und Abwandlungen. Bspw. muss die Neuberechnung der Mittelpunkte nicht erst am Ende eines Laufes durch das gesamte Datenset geschehen. Stattdessen können die Mittelpunkte auch unmittelbar, wenn ein neuer Punkt zum Cluster zugeordnet oder entfernt wird, aktualisiert werden. King [King15, Kap. 4.5 K-Means Algorithm] bezeichnet dieses alternative Vorgehen als “MacQueens-Methode”. Viele weitere Varianten sind in der Literatur beschrieben worden. Im folgenden wird nur eine kleine relevante Auswahl dessen präsentiert.

K-Medoids

Der K-Medioids ist eine Variante des K-Means-Algorithmus, bei dem die Centroids der Cluster tatsächliche Datenpunkte des Datensets sein müssen. Anstatt den Durchschnitt der jeweiligen Attribute aller Punkte im Cluster zu berechnen, werden die Abstände aller Punkte eines Clusters zueinander betrachtet. Der Punkt, welcher in der Summe den kleinsten Abstand zu allen anderen ausweist, muss zwangsläufig “mittig” im Cluster liegen. Fortan gilt dieser Punkt als Centroid des Clusters. Dieses Verfahren ist rechenaufwendiger als der normale K-Means. Allerdings konvergieren die Cluster häufig in weniger Durchläufen zu einem stabilen Optimum. Deshalb kann dieses Verfahren, auf den gesamten Ablauf betrachtet, dennoch Performance-Vorteile zum klassischen K-Means bringen. [ArVa16]

K-Modes

Der K-Modes verarbeitet ausschließlich kategorische Attribute – statt numerischen. Das Grundprinzip des K-Means bleibt bestehen. Als Distanzmaß muss eine entsprechend geeignete Metrik verwendet werden (im Original das Simple Matching). Außerdem erfolgt die Berechnung der Cluster-Mittelpunkte nicht über den Durchschnitt, da zwischen kategorischen Attributen kein Durchschnitt berechnet werden kann. Daher bilden sich die Centroids aus dem Modus (engl. mode), also dem am häufigsten auftretenden Label im jeweiligen Attribut. [Huan98]

Dieser Algorithmus verwendet außerdem den "MacQueens-Ansatz". D.h. dass sich der Modus für jedes Attribut direkt aktualisiert, wenn ein Objekt dem Cluster neu zugeordnet oder entfernt wird. Auch hier wird dieser Prozess mehrmals durchlaufen, solange bis sich die Clusterzuteilungen nach einem kompletten Durchlauf durch das Datenset nicht mehr verändert haben. [Huan98]

K-Prototypes

Der K-Prototypes kombiniert den K-Means- und K-Modes-Algorithmus zum Clustern gemischter Datensets. Die Centroids berechnen sich entsprechend aus dem Durchschnitt der Werte numerischer und aus dem Modus der Werte kategorischer Attribute. Die Distanz wird mittels einer kombinierten Distanzfunktion (siehe der entsprechende Abschnitt) berechnet. [Huan98]

2.4.2 Hierarchisches Clustering

Überblick

Hierarchische Verfahren produzieren eine ineinander verschachtelte Partitionierung der Datenpunkte in Cluster. Dabei werden zwei grundlegende Arten von Verfahren unterschieden [StKaKu00; und King15, Kap. 3.1 Hierarchical Clustering – Introduction]:

- *Top-down- oder divisive Verfahren* beginnen mit einem Cluster, welches alle Datenpunkte enthält. Anschließend wird dieses Cluster in zwei kleinere Cluster aufgesplittet, danach werden die beiden resultierenden Cluster wiederum jeweils in zwei kleinere aufgeteilt usw. Dieses Verfahren wird solange fortgesetzt, bis nur noch Cluster übrig bleiben, die genau einen Datenpunkt enthalten.
- *Bottom-up- oder agglomerative Verfahren* starten exakt andersrum mit jeweils einem Cluster für jeden Datenpunkt. Anschließend werden immer wieder zwei Cluster zu einem größeren kombiniert, solange bis im letzten Schritt ein Cluster entsteht, welches wiederum alle Datenpunkte enthält.

Das bedeutet, dass jeder Datenpunkt mehreren verschiedenen Clustern zugeteilt ist, je nachdem welche Stufe in der Hierarchie betrachtet wird. Alternativ können die Ergebnisse auch als Baumstruktur in einem sog. Dendrogram dargestellt werden. [StKaKu00]

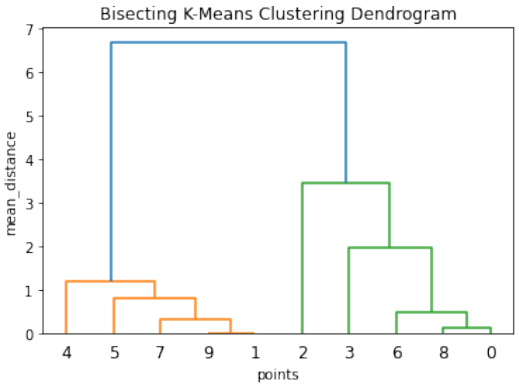

Abbildung 2.1: Bsp. eines Dendrograms

Ein großer Vorteil von diesem Verfahren ist, dass die Anzahl an gesuchten Clustern nicht vorher bekannt sein muss. Stattdessen wird eine Clusterzuteilung für alle möglichen Clusterzahlen ($1 \leq k \leq n$) im Zuge der Berechnung ermittelt [KaRo09, Kap. 1.3.2 Hierarchical Methods]. Dadurch können die verschiedenen Hierarchie-Ebenen mit geeigneten Metriken (siehe Abschnitt Cluster-Validität) bewertet und anschließend z.B. eine geeignete Clusteranzahl für die Durchführung eines partitionierenden Verfahrens ausgewählt werden. [King15, Kap. 4.1 Partition Clustering – Introduction]

Weiterhin bringt die hierarchische Struktur vielfältige Möglichkeiten der Analyse. So kann nicht nur festgestellt werden, welche Objekte in verschiedene Cluster sortiert werden, sondern auch welche der Cluster sich ähnlicher sind (aufgrund ihrer Position in der Hierarchie). Das Ermitteln von Outliern wird ebenfalls möglich, da diese meistens recht früh in der Hierarchie in ihr eigenes Cluster platziert werden. [DoBi22]

Ein großer Nachteil dieser Verfahren ist, dass die Clusterzuteilungen im Laufe der Berechnung nicht mehr nachträglich korrigiert werden. Nachdem also zwei Datenpunkte in einem Top-Down-Verfahren in verschiedene Cluster gesplittet worden sind, werden sie in den nachfolgenden Hierarchie-Ebenen niemals wieder im selben Cluster landen (für Bottom-Up-Verfahren gilt das gleiche entsprechend umgekehrt). Dies führt u.U. zu einer ungeeigneten Clusterzuteilung je nach betrachteter Hierarchie-Ebene. [KaRo09, Kap. 1.3.2 Hierarchical Methods]

Agglomeratives Clustering (Bottom-Up)

Wie bereits erklärt beginnt das agglomerative Clustering mit jedem Datenpunkt in einem eigenen Cluster. Anschließend werden stets die beiden “am nächsten gelegen” bzw. “ähnlichsten” Cluster zu einem größeren zusammengefasst. Dieser Prozess wird solange wiederholt, bis sich zum Schluss alle Datenpunkte in einem großen Cluster befinden. [StKaKu00]

Von zentraler Bedeutung dabei ist die Frage, wie zwei Cluster miteinander auf ihre Ähnlichkeit untersucht werden. Für die vorkommenden Datenpunkte wird dazu ein geeignetes Distanz- oder Ähnlichkeitsmaß gewählt. Die Berechnung der Ähnlichkeit zwischen zwei Clustern (welche ja potenziell aus mehreren Punkten bestehen können) geschieht dann mit Hilfe sog. Fusionierungsverfahren (engl. linkage functions). Ein bekannter Vertreter ist z.B. der “Single-Link”, welcher den Abstand zwischen zwei Clustern über den Abstand der

zwei am nächsten gelegenen Punkte der Cluster definiert [DoBi22]. Nachfolgend sind einige typische Vertreter aufgelistet:

Single-Link $d(C_1, C_2) = \min_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
Abstand wird durch die beiden dichtesten Punkte definiert; neigt zur Bildung von langen Ketten (chaining effect); anfällig gegen Outlier

Complete-Link $d(C_1, C_2) = \max_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
Abstand wird durch die beiden am entferntesten liegenden Punkte definiert; neigt zur Bildung von vielen sehr kleinen Clustern; anfällig gegen Outlier

Average-Link $d(C_1, C_2) = \frac{1}{|C_1|} \frac{1}{|C_2|} \sum_{x_i \in C_1} \sum_{x_j \in C_2} d(x_i, x_j)$
durchschnittlicher Abstand aller Punkte der beiden Cluster zueinander; ähnlich dem Mittelpunkt von k-Means, aber viel aufwendiger, da immerzu jeder Punkt eines Clusters mit jedem anderen Punkt der anderen Clustern verglichen werden muss

Centroid-Link $d(C_1, C_2) = d([\frac{1}{|C_1|} \sum_{x_i \in C_1} x_i], [\frac{1}{|C_2|} \sum_{x_j \in C_2} x_j])$
Abstand der Mittelpunkte der Cluster zueinenader (Berechnung des Mittelpunktes wie bei K-Means); ähnliches Verhalten wie Average-Link, aber effizienter zu berechnen, da der Mittelpunkt nur einmal pro Cluster berechnet werden muss

Ward-Methode $d(C_1, C_2) = \frac{|C_1||C_2|}{|C_1|+|C_2|} d([\frac{1}{|C_1|} \sum_{x_i \in C_1} x_i], [\frac{1}{|C_2|} \sum_{x_j \in C_2} x_j])$
berechnet die Zunahme der Varianz durch den jeweiligen Merge; neigt zur Bildung von Kreis- oder Kugelförmigen Clustern; recht effizient; anfällig gegen Outlier

Da für den Centroid-Link und die Ward-Methode die Mittelpunkte der Cluster gebildet werden müssen, sind diese Verfahren primär nur für numerische Datensets geeignet [DoBi22]. Die Verwendung dieser Verfahren für gemischte Attribute (z.B. durch Verwendung des Modus für kategorische) ist in der Literatur nicht beschrieben worden.

Ein großer Nachteil dieser Verfahren ist die Lauzeit. Da im ersten Schritt jeder Datenpunkt mit jedem anderen verglichen werden muss (es sind ja zu Beginn n Cluster), beträgt die Laufzeit immer mindestens $\mathcal{O}(n^2)$. Durch geschickte Implementierung, kann diese Laufzeit auch für einige Verfahren (z.B. für Single- [Sibs73] oder Complete-Link [Defa77]) tatsächlich erreicht werden. Häufig liegen die Laufzeiten aber bei $\mathcal{O}(n^2 \log n)$ oder sogar $\mathcal{O}(n^3)$. [DoBi22]

Divisives Clustering (Top-Down)

Überblick Divisives Clustering startet mit allen Datenpunkten in einem großen Cluster. Anschließend werden immerwieder Zweier-Splits der Cluster durchgeführt, solange bis final jeder Datenpunkt in genau einem eigenen Cluster platziert ist. [KaRo09, Kap. 6 Divisive Analysis (Program DIANA)]

Lange Zeit wurde dieser Ansatz des hierarchischen Clusterings ignoriert, da das Verfahren auf den ersten Blick sehr rechenintensiv erscheint. Im aller ersten Zustand, wenn alle Punkte in einem Cluster sind, gibt es theoretisch $2^n - 1$ verschiedene Möglichkeiten dieses große Cluster in zwei kleinere zu teilen. Eine solche exponentielle Laufzeit könnte niemals für Datensets auch nur mittlerer Größe verwendet werden. Allerdings ist es nicht nötig alle dieses Splits zu betrachten und durchzurechnen. Mittlerweile gibt es einige Ansätze für diese Art von Clustering-Verfahren, welche dieses Problem geschickt umgehen. [KaRo09, Kap. 6 Divisive Analysis (Program DIANA)]

Außerdem zeigt die Praxis, dass Top-down-Clustering i.d.R. bessere und intuitivere Cluster generiert. Der Grund ist vermutlich, dass wir Menschen beim Clustering per Hand automatisch eine Art Top-down-Denkweise an den Tag legen. [King15, Kap. 3.3 Agglomerative versus Divisive Clustering]

DIANA DIANA steht für "divisive analysis" und wurde von Kaufman und Rousseeuw [KaRo09, Kap. 6 Divisive Analysis (Program DIANA)] nach einer Idee von Macnaughton-Smith et al. [MWDM64] entwickelt.

Sie beschreiben den Ablauf über die Analogie einer politischen Partei, welche sich durch Meinungsverschiedenheiten aufsplittet. Zuerst tritt das Mitglied mit den größten Differenzen zum Rest der Partei aus derselbigen aus und gründet eine eigene. Anschließend entscheiden sich nach und nach die restlichen Mitglieder, ob sie ebenfalls "die Fronten wechseln" oder in der ursprünglichen Partei verbleiben wollen [KaRo09, Kap. 6 Divisive Analysis (Program DIANA)]. Aufgrund dieser Analogie wird das Verfahren auch als "Splitterpartei-Methode" bezeichnet [RaRa11].

Konkret lässt sich das Verfahren von Kaufman und Rousseeuw, wie folgt beschreiben:

1. Start mit allen Punkten in einem großen Cluster
2. Berechnung des durchschnittlichen Abstandes jedes Datenpunktes zu den restlichen Punkten im Cluster
3. der Punkt mit der größten mittleren Abweichung verlässt das große Cluster und bildet ein neues
4. alle restlichen Punkte aktualisieren ihren durschnittlichen Abstand
5. die Punkte, deren durchschnittlicher Abstand zum Cluster größer ist als ihr Abstand zum neu herausgelösten (also der Splittergruppe), verlassen das ursprüngliche Cluster ebenfalls und werden dem neuen Cluster zugeordnet
6. Wiederholgung der Schritte ab 4., solange bis keine Punkte mehr das Cluster wechseln
7. aus den entstandenen Clustern wird das größte (mit dem größten Durchmesser) ausgewählt und die Schritte ab 2. wiederholt, solange bis jeder Datenpunkt in einem eigenen Cluster gelandet ist

Der *durchschnittliche Abstand md* eines Datenpunktes x_i zu den anderen Punkten x_j im Cluster C berechnet sich aus dem Durchschnitt der einzelnen Abstände zu den jeweils anderen Punkten. Diese Abstände werden mit einem geeigneten Distanzmaß berechnet. Damit ist dieses Verfahren für jede Art von Datentypen geeignet, zu denen auch ein Distanzmaß konzipiert werden kann. Mathematisch berechnet sich md für den Datenpunkt $x_i \in C$, wie folgt:

$$md(x_i, C) = \frac{1}{|C| - 1} \sum_{x_j \in C, x_i \neq x_j} d(x_i, x_j) \tag{2.16}$$

Im Ablauf des Verfahrens ist von einem *Clusterdurchschnitt* die Rede. Es dient zur Bestimmung, welches Cluster als nächstes geteilt werden soll. Der Clusterdurchschnitt cd ist der Abstand der beiden Datenpunkte im Cluster, die am weitesten voneinander entfernt sind:

$$cd(C) = \max_{x_i \in C, x_j \in C} d(x_i, x_j) \tag{2.17}$$

Die genaue Laufzeit des Verfahrens ist nirgendwo beschrieben worden. Am maßgeblichsten müsste der Split des ersten großen Clusters für die Laufzeit sein. Hier werden initial alle Datenpunkte miteinander verrechnet, sodass mindestens eine Laufzeit von $\mathcal{O}(n^2)$ zustande kommt. Die nachfolgenden Splits sind zunehmend weniger aufwendig in der Berechnung, da die Cluster mit jedem Durchlauf kleiner werden. Dadurch sinkt die Menge an Datenpunkten, die miteinander verrechnet werden müssen, immer weiter ab.

Rajalingam und Ranjini [RaRa11] verglichen die Performance einiger Bottom-up-Verfahren mit einem Top-down-Verfahren, welches dem beschriebenen DIANA entweder identisch oder sehr ähnlich ist. In ihren Versuchen war das Top-down-Verfahren fast doppelt so schnell in der Berechnung der Cluster.

Bisecting K-Means Der Bisecting K-Means wurde von Steinbach et al. [StKaKu00] zum Clustern von Dokumenten entwickelt. Ihr ursprünglicher Plan war der Vergleich von agglomerativen Verfahren mit dem klassichen K-Means-Algorithmus. In diesem Prozess entwickelten sie die Idee für das divise Verfahren – Bisecting K-Means.

Initial befinden sich wieder alle Datenpunkte in einem großen Cluster. Für den Split in zwei kleinere Cluster wird nun der klassische K-Means-Algorithmus mit $k = 2$ ausgeführt. Aus den resultierenden kleineren Clustern gilt es nun einen Kandidaten für den nächsten Split auszuwählen. Das kann das Cluster mit dem größeren Durchschnitt oder mit der höheren Anzahl an Datenpunkten sein (in ihren Versuchen ergab sich zwischen beiden Vorgehen kein Unterschied). Der Split wird wiederum durch K-Means mit $k = 2$ ausgeführt usw. Das Verfahren endet, wie gewohnt, wenn alle Punkte in ihrem eigenen Cluster gelandet sind. [StKaKu00]

Das Verfahren besticht durch seine Laufzeit $\mathcal{O}(n)$, welche durch den K-Means für den allerersten Split bestimmt wird. Alle folgenden Splits sind durch die kleinere Menge an Datenpunkten je Cluster zu vernachlässigen. [StKaKu00]

2.5 Cluster-Validität

2.5.1 Überblick

Die Clusteranalyse selbst ist ein Verfahren des unüberwachten Lernens und findet interne Muster in Daten ohne Referenz zu externen Zuweisungen (Labels). Dennoch finden verschiedene Clusteringverfahren unterschiedliche Gruppenzuteilungen. Auch die verschiedenen Parameter, die für ein jeweiliges Clusteringverfahren gesetzt werden, beeinflussen das Ergebnis. Daher bedarf es Methoden zur Evaluation und Vergleich der Clusterings miteinander. [RAAQ11]

Externe Indizes (engl. External Indices) sind Metriken, welche die berechnete Gruppenzuteilung mit einer extern vorgegeben Zuteilung vergleicht. Das heißt, es gibt eine erwartete Art der Gruppierung, welche nicht Teil des Datensets selbst ist. Diese Indizes messen nun den

Grad der Übereinstimmung zwischen berechnetem und gewünschten Clustering-Ergebnis. [RAAQ11]

Interne Indizes (engl. Internal Indices) messen die Qualität des Clusterings ohne externe Informationen. Die Cluster sollten möglichst "kompakt" gruppiert und die verschiedenen Gruppen "gut von einander getrennt" sein. Diese Indizes versuchen diese Anforderungen zu quantifizieren. [RAAQ11; und King15, Kap. 8.1. Cluster Validity – Introduction]

Die **Stabilität** ist ebenfalls ein wichtiger Faktor. Hierbei wird das Datenset mehrmals mit verschiedenen Modifikationen geclustert. Solche Modifikationen können die Änderung einzelner Werte oder das Weglassen ganzer Spalten sein. Ein "stabiles" Clustering erzeugt auch mit diesen Veränderungen ähnliche Ergebnisse. Für die konkrete Bewertung der Stabilität werden externe oder interne Indizes verwendet und ihre Ergebnisse über die verschiedenen Clusterings miteinander verglichen. [King15, Kap. 8.1. Cluster Validity – Introduction]

Manche Autoren (z.B. [RAAQ11] und auch [King15]) führen formal noch sog. **Relative Inidzes** an. Hiermit wird die Performance verschiedener Clustering-Verfahren und verschiedener Meta-Parameter der Clusterings über das gleiche Datenset verglichen. In der Praxis erfolgt dies aber stets über den Vergleich der externen oder internen Indizes der Clustering-Ergebnisse miteinander.

2.5.2 External Indices

Überblick

Zur Bewertung der Übereinstimmung zweier Clustering-Ergebnisse können grundsätzlich Metriken aus dem Bereich der Klassifikation verwendet werden [HuAr85]. Beispiele dafür wären Maße wie die Entropie oder ihr Gegensatz die Reinheit (engl. purity), welche analysieren, wie viele falsche zu richtigen Zuordnungen innerhalb einer Klasse aufgetreten sind [RAAQ11]. Ebenso das F-Maß (engl. F-Measure), welches aus dem Bereich des Document Retrievals bekannt ist. Es simuliert durchgeführte "Suchen" und vergleicht die gefundenen mit den erwarteten Suchergebnissen [RAAQ11; und StKaKu00].

Das Problem mit diesen Maßen ist, dass die Benennung der Labels im Ergebnis von entscheidender Bedeutung in der Bewertung ist. Nehmen wir an, wir haben ein Datenset X mit vier Datenpunkten. Die berechnete Clusterzuordnung ist $Y = (0, 0, 1, 1)$ und die erwartete Zuordnung ist $Y' = (1, 1, 0, 0)$. Metriken der Klassifikation würden eine Übereinstimmung von null Prozent feststellen, da keines der Labels in Y und Y' den gleichen Klassen zugeordnet worden ist. Im Kontext der Clusteranalyse weisen Y und Y' aber eine perfekte Übereinstimmung auf, denn es geht alleine um die Zuordnung der Datenpunkte in die gleichen Gruppen. Wie diese Gruppen benannt werden (in diesem Beispiel 0 und 1), spielt dabei keine Rolle. Daher sind für die Clusteranalyse eigene Metriken entwickelt worden, welche unabhängig von der Benennung der Labels die Übereinstimmung zwischen den Zuordnungen berechnen können. [Rand71; und HuAr85]

Rand-Index

Der Rand-Index war einer der ersten Metriken, die speziell für die Clusteranalyse entwickelt worden sind [King15, Kap. 8.4 Indices of Cluster Validity]. William M. Rand veröffentlichte dieses Maß 1971 [Rand71] und es ist immer noch eines der populärsten und weitverbreitetsten (bzw. vor allem die Verbesserungen dieses Indexes siehe nächster Abschnitt). [HuAr85; und Stei04]

Um unabhängig von der Benennung der Labels zu werden, vergleicht man nicht wie in der Klassifikation die gefundenen Labels eins zu eins miteinander. Stattdessen werden alle möglichen Paarungen der Datenpunkte in X betrachtet und die Zuordnung der Paare in Y und Y' miteinander verglichen. [Rand71]

Tabelle 2.3: Kontingenz der Datenpunkt-Paare in Y und Y' [Stei04]

$Y \setminus Y'$	Paar im gleichen Cluster	Paar in anderem Cluster
Paar im gleichen Cluster	a	b
Paar in anderem Cluster	c	d

Die Tabelle zeigt die möglichen Fälle: Ein Paar aus Datenpunkten kann entweder dem gleichen Cluster oder zwei unterschiedlichen Clustern zugeordnet worden sein. Haben sowohl Y als auch Y' das Paar jeweils in das gleiche Cluster (mit dem gleichen Label wie auch immer dieses benannt ist) zugordnet, so wird dieses Paar zu a gezählt. Haben beide jeweils ein unterschiedliches Cluster zugeordnet, so wird das Paar in d gezählt usw. [Stei04]

$$rand = \frac{a+d}{a+b+c+d} = \frac{a+d}{\binom{n}{2}} \tag{2.18}$$

Der Rand-Index teilt nun die Menge an Paaren, welche die gleiche Zuordnung erhalten haben (a und d) durch die Anzahl aller möglichen Paarungen. Das Ergebnis ist ein Wert zwischen 0 (keine Übereinstimmung) und 1 (perfekte Übereinstimmung). [Rand71]

Der Rand-Index weist eine Reihe von Problemen auf, welche teilweise von Rand selbst erkannt oder später ermittelt worden sind:

- Je höher die Anzahl an Clustern ist, desto höher (näher an der 1) liegt der Rand-Index standardmäßig. Das kommt daher, dass bei einer hohen Anzahl an Clustern die meisten Paare unterschiedlichen Clustern zugewiesen sein müssen. Das treibt den Index künstlich in die Höhe. [Stei04; teilweise zitiert nach Rand71]
- Zum Vergleich verschiedener Clustering-Verfahren werden häufig Monte-Carlo-Simulationen mit großen Mengen an zufällig generierten Datensätzen und Clusterings verwendet. Werden zwei Zufalls-Zuordnung miteinander verglichen, so gibt der Rand-Index keine Werte nahe der 0, was aber wünschenswert wäre. Das liegt daran, dass die Anzahl an zufälligen Übereinstimmungen nicht adäquat herausgerechnet wird. [HuAr85; und King15, Kap. 8.4 Indices of Cluster Validity]

Adjusted Rand-Index

Aus den genannten Problemen haben eine Vielzahl von Autoren versucht, eine bessere Variante des Rand-Indexes zu finden. Vor allem das "Herausrechnen" von angeblichen hohen Übereinstimmungen bei zufälligen Zuteilungen (engl. correction for chance) ist eines der Hauptanliegen gewesen [HuAr85]. Von allen vorgestellten Lösungen scheint die Variante von Hubert und Arabie [HuAr85] diejenige mit den wünschenwertesten Eigenschaften zu sein. (siehe die Versuche z.B. von Steinley [Stei04])

$$arand = \frac{rand - rand_{expected}}{rand_{max} - rand_{expected}} \tag{2.19}$$

Im Allgemeinen geht es darum, den Rand-Index zu berechnen und anschließend den "erwarteten Rand-Index" für zufällige Zuordnungen davon abzuziehen. Verschiedene Autoren haben nun unterschiedliche Methoden hergeleitet, um diesen "erwarteten Rand-Index" zu berechnen. [HuAr85]

$$arand = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \tag{2.20}$$

Hubert und Arabie berechnen den "erwarteten Rand-Index" aus der Menge aller möglichen Permutationen der Paarungen und ihrer Übereinstimmung $\frac{(a+b)(a+c)+(c+d)(b+d)}{\binom{n}{2}}$. Die gegebene Formel zeigt eine breits vereinfachte Umstellung von Steinley. [HuAr85; und vereinfachte Formel aus Stei04]

Dieser "Adjusted Rand-Index" liefert Werte zwischen -1 und 1. Negative Ergebnisse oder Werte um die 0 zeigen, dass die Übereinstimmung der beiden Clusterings eher auf den Zufall zurückzuführen und damit nicht signifikant ist. Höhere Zahlen nahe der 1 stehen nach wie vor für eine perfekte Übereinstimmung, welche zusätzlich statistische Signifikanz aufweist. Auch der Bias zu hohen Werten bei einer hohen Anzahl an Clustern wird dadurch ausgeglichen. [HuAr85; und Stei04]

2.5.3 Internal Indices

In der Literatur werden eine Vielzahl von Indizes beschrieben, welche die Qualität der gefundenen Cluster messen können. Einen Überblick dazu geben Rendón et al. [RAAQ11]. An dieser Stellen sollen nur ein paar Vertreter vorgestellt werden, welche für diese Arbeit von besonderer Relevanz sind.

Silhouettenkoeffizient

Der Silhouettenkoeffizient wurde von Peter J. Rousseeuw entwickelt [Rous87] und ist ein häufig verwendetes Maß für die Qualität von Clusterings.

$$s(x) = \frac{b(x) - a(x)}{\max a(x), b(x)} \tag{2.21}$$

$$a(x) = d(x, C_x) \tag{2.22}$$

$$b(x) = \min_{C_x \neq C_i} d(x, C_i) \tag{2.23}$$

Für jeden Datenpunkt x im Datenset wird die Silhouetten-Weite $s(x)$ berechnet. Diese ergibt sich aus der Differenz zwischen dem durchschnittlichen Abstand zu allen Datenpunkten im selben Cluster ($a(x)$) und dem durchschnittlichen Abstand zu allen Datenpunkten im direkt benachbarten Cluster ($b(x)$). Anschließend wird der Wert zwischen -1 und 1 normiert. [Rous87]

- Eine negative Silhouetten-Weite gibt an, dass der Datenpunkt näher am benachbarten als an seinem eigenen Cluster liegt und somit falsch zugeordnet worden ist.
- Werte um die 0 zeigen, dass der Datenpunkt fast mittig zwischen beiden Clustern platziert ist.
- Befindet sich die Weite nahe der 1, so liegt der Datenpunkt mittig in seinem eigenen Cluster und das benachbarte Cluster ist ordentlich weit entfernt.

$$sil = \frac{1}{n} \sum_{i=1}^{n} s(x_i) \tag{2.24}$$

Der Koeffizient ergibt sich schließlich aus dem Durchschnitt aller Silhouetten-Weiten aller Datenpunkte. Werte nahe der 1 deuten auf kompakte und wohl-separierte Cluster hin. [Rous87]

In den Versuchen von Rendón et al [RAAQ11] erwies sich dieser Index als einer der besten Indikatoren für ein gutes Clustering-Ergebnis. Er ist die direkte mathematische Definition für die Anforderung, dass Datenpunkte im gleichen Cluster möglichst ähnlich und zu den Punkten der anderen Cluster möglichst unähnlich sein sollen. Ein Nachteil dieses Indizes ist die quadratische Laufzeit, da jeder Punkt mit jedem anderen verglichen werden muss.

Davies-Bouldin Index

Der Davies-Bouldin Index wurde von seinen Namesgebern David L. Davies und Donald W. Bouldin [DaBo79] entwickelt. Ziel war es, eine Metrik zu konstruieren, welche die durchschnittliche Ähnlichkeit zwischen benachbarten Clustern berechnet.

$$dbi = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \frac{d(c_i, C_i) + d(c_j, C_j)}{d(c_i, c_j)} \tag{2.25}$$

Die Ähnlichkeit zwischen den Clustern K (K steht hier für die Menge an gefundenen Clustern C_i) berechnet sich aus dem durchschnittlichen Abstand der Punkte eines Clusters zu ihrem Cluster-Mittelpunkt ($d(c_i, C_i)$ und $d(c_j, C_j)$) geteilt durch den Abstand der beiden Cluster-Mittelpunkte zueinander ($d(c_i, c_j)$). [DaBo79]

Je kleiner der Wert des Indexes, desto enger liegen die Datenpunkte um ihren Cluster-Mittelpunkt im Verhältnis zum benachbarten Cluster. Möglichst niedrige Werte nahe der 0 sind also als optimal anzusehen. [DaBo79]

Der große Vorteil von diesem Verfahren ist die geringere Laufzeit, da die Punkte der Cluster nur mit ihrem Mittelpunkt verrechnet werden. Nachteilig ist, dass die Be- und Verrechnung der Mittelpunkte primär nur für numerische Vektoren definiert ist. In den Versuchen von Rendón et al. [RAAQ11] schnitt dieser Index genauso gut ab wie der Silhouettenkoeffizient.

3 Konzeption

3.1 Überblick

Zur Beantwortung der Kernfrage wird ein Konzept für das Clustern komplexerer Datenstrukturen erarbeitet und anschließend evaluiert. Diese Herleitung soll aber nicht nur theoretischer Natur sein. Daher wird die Überprüfung des Konzeptes an einem möglichst realen Beispiel mit realitätsnahen Daten erfolgen. Die folgende Grafik zeigt einen Überblick zum Praxisteil dieser Arbeit:

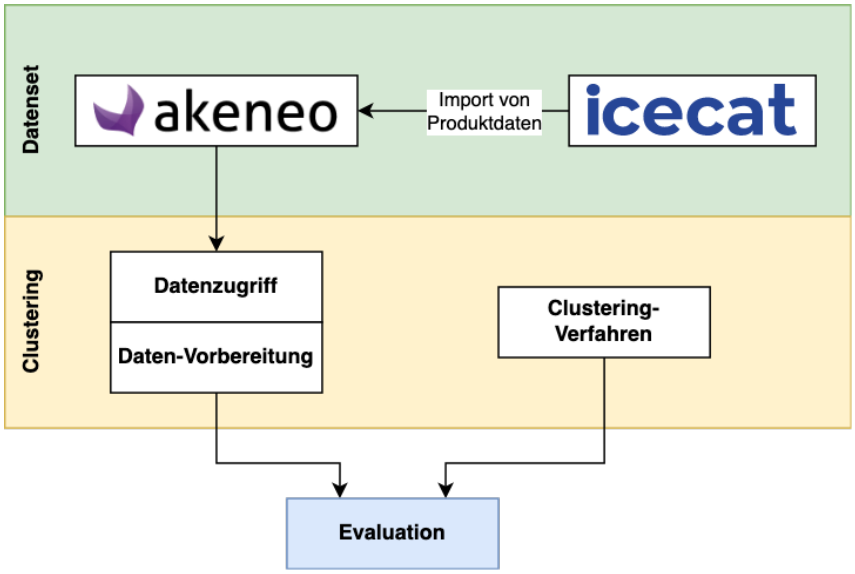

Abbildung 3.1: Grober Architektur-Entwurf für den Praxisteil

Um ein authentisches *Datenset* in den typischen Datenstrukturen zu erhalten, wird eine Instanz des Product-Information-Management-Systems Akeneo-PIM aufgesetzt und mit Produktdaten aus dem umfangreichen Online-Katalog Icecat gefüllt. Im Abschnitt Datenquellen & -sets werden beide Komponenten genauer erläutert. Außerdem werden genaue Anforderung für die Auswahl der Produkte, welche importiert werden, definiert.

Parallel wird ein Konzept für ein *Clustering-Verfahren* erarbeitet, welches die vielfältigen Datenstrukturen verarbeiten kann. Dies setzt i.d.R. eine entsprechende Vorverarbeitung der Daten voraus. Das entsprechende Konzept wird im Abschnitt Clustering-Verfahren hergeleitet.

In der *Evaluation* kommt schließlich das erstellte Datenset mit dem erarbeiteten Clustering-Verfahren zusammen. Es wird überprüft, inwiefern das Verfahren "sinnvolle" Cluster in den Produktdaten findet. Eine genaue Definition von "sinnvollem" Clustering sowieso die konkrete Berechnung entsprechender Metriken usw. wird im Abschnitt Evaluation dargelegt.

3.2 Datenquellen & -sets

3.2.1 Akeneo-PIM

Als realistische Datenquelle ist das Product-Information-Management-System Akeneo-PIM https://www.akeneo.com/de/akeneo-pim-community-edition/ ausgewählt worden.

Überblick

Der Ursprung von Product-Information-Management-Systemen liegt in gedruckten Produktkatalogen. Speziell im letzten Jahrhundert waren diese die zentrale Form, wie Produktdaten verteilt und verwaltet worden sind. Mit beginnender Digitalisierung und dem Aufkommen des E-Commerce in den 1990er-Jahren wurden zunehmend computer-basierte Systeme zur Warenhaltung und Bestellabwicklung, sog. ERP-Systeme, verwendet. Anfangs wurden in diesen Systemen auch die Produktdaten gespeichert. Dieses Vorgehen stieß aber zunehmend an Grenzen, da die schiere Vielfalt an Daten (Produkte, Bestellungen, Transaktionen, Warenbestand usw.) schwer zu verwalten war. Mitte der 2000er kamen spezielle Systeme zum Einsatz, die lediglich die Produktdaten enthielten. Somit konnte der "Produktkatalog" zentral verwaltet, angereichert, übersetzt und aufbereitet werden. Weitere Systeme (z.B. besagte ERP-Systeme) konnten nun auf diesen zentralen Katalog zugreifen und müssen selbst nur minimale Informationen zu den Produkten vorhalten. PIM-Systeme sitzen daher heute häufig im Zentrum der Unternehmens-Architektur und dienen als zentrale Datenquelle für die angeknüpften Systeme. [Pimc22]

"Akeneo-PIM" der gleichnamigen Firma ist ein solches PIM-System, welches heute breite Anwendung findet. Es ist in einer betreuten Enterprise Edition erhältlich sowie als Open-Source-Variante "Akeneo-PIM Community Edition" (verfügbar unter https://github.com/akeneo/pim-community-dev), welche selbst gehostet werden muss. Die Community-Variante ist frei erhältlich und daher sehr populär. [Aken22a]

Abbildung 3.2: Schematische Darstellung der Funktionen von Akeneo-PIM [Aken22b]

Ein weiterer Grund für die weite Verbreitung ist, dass Akeneo-PIM über diverse Schnittstellen an externe Systeme angebunden werden kann. Ebenso unterstützt es eine Vielzahl an Import- und Export-Möglichkeiten in unterschiedliche Datenformaten. Es gibt außerdem

den “Akeneo App Store” (https://marketplace.akeneo.com/), wo viele Anbieter Plugins veröffentlichen, um die Funktionen und Anbindungen von Akeneo-PIM zu erweitern. [Aken22b]

Diese Gründe führten zur Verwendung von Akeneo-PIM für diese Arbeit.

Systemübersicht

Akeneo-PIM ist eine PHP-Applikation, welche auf dem Symfony-Framework basiert. Für die Datenspeicherung wird die Verwendung von einer MySQL-Datenbank empfohlen. Zum schnelleren Suchen und Abrufen kommt Elasticsearch zum Einsatz. [Aken22c] Hinzu kommen weitere Technologien zur Generierung des Webfrontends. Um diesen verschiedenen Tools Herr zu werden, stellt Akeneo Konfigurationen für Containerization-Tools wie Docker-Compose zur Verfügung. [Aken22d]

Das System kann über ein Webfrontend genutzt werden. Hier lassen sich verschiedene Nutzer und Applikationen hinterlegen, welche verschiedene Berechtigungen zum Abrufen und Bearbeiten der Daten bekommen können.

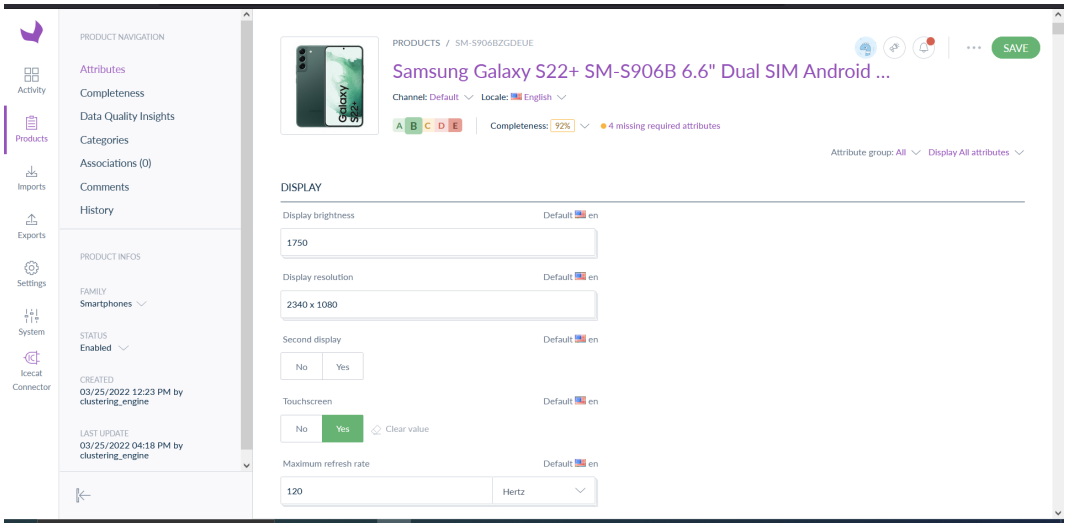

Abbildung 3.3: Akeneo-PIM Webfrontend bei der Bearbeitung eines Produktes

Eine Restful-API steht ebenfalls zur Verfügung, um Daten mittels HTTP-Requests abzurufen und zu verändern. In der Regel werden die Daten im JSON-Format transferiert und zurückgegeben. [Aken22e]

Datenstrukturen

Innerhalb von Akeneo-PIM kommen eine Vielzahl von verschiedenen Datenstrukturen zum Einsatz. Im Folgenden werden nur die für diese Arbeit wichtigsten Strukturen kurz erklärt.

Product & Product Values Die Produkte sind das zentrale Element in Akeneo-PIM. Aus ihnen leiten sich eine Reihe weiterer Strukturen ab. [Aken22f]

Jedes Produkt wird genau einer "Family" zugeordnet. Sie bestimmt, welche Attribute dieses Produkt aufweisen sollte.

Ein Produkt kann einer oder mehreren "Categories" zugeordnet sein. Der Sinn dabei ist der Aufbau z.B. von Navigationsstrukturen. Hierbei ist die Zuordnung in mehrere verschiedene Kategorien möglich.

Die "ProductValues" sind die tatsächlichen Wertausprägungen eines Produktes. Sie werden in Form einer Hash-Map angegeben. Der Key der Einträge ist stets der "code" eines bestimmten "Attributes" (eine weitere Datenstruktur). Dieses Attribut wird separat definiert und legt verschiedene Constraints für den "ProductValue" fest. Dabei bietet Akeneo volle Flexibilität. Jedes Produkt kann jedes beliebige Attribut mit einem entsprechenden Wert aufweisen.

```
{
  "identifier": "1111111195",
  "family": "clothing",
  "categories": ["tshirts"],
  "values": {
    "ean": [
      {"locale": null, "scope": null, "data": "1234567890207"}
    ],
    "size": [
      {"locale": null, "scope": null, "data": "s"}
    ],
    // ...
  },
  "enabled": true,
  "created": "2017-10-05T11:25:48+02:00",
  "updated": "2017-10-05T11:25:48+02:00",
  // ...
}
```

Category Mit einer "Category" können Produkte in verschiedene Kategorien sortiert werden. Wichtig ist, dass "Categories" beliebig verschachtelt und hierarchisch sortiert werden können. Es gibt kein Limit für die Tiefe der Verschachtelung. [Aken22g]

```
{
  "code": "tvs_projectors",
  "parent": "master",
  "labels": {
    "en_US": "TVs and projectors",
    // ...
  }
}
```

Hinweis: Mit Ausnahme der "Products" haben die meisten Entitäten einen "code", welcher als eindeutiger Identifier fungiert.

Family Eine "Family" definiert die Attribute, welches ein zugehöriges "Product" aufweisen sollte. Dies ist allerdings keine strikte Zuteilung. Produkte können auch Attribute aufweisen, welche gar nicht in ihrer "Family" vorgesehen sind. Ebenso können in der Family erforderliche

(engl. required) Attribute hinterlegt werden. Produkte müssen diese aber nicht erfüllen. Das Produkt weist dann lediglich einen niedrigeren "Completeness Score" auf. Dadurch kann in Akeneo auch mit unvollständigen Produkten gearbeitet werden, bevor man diese weiter anreichert. [Aken22g]

```
{
  "code": "camcorders",
  "attributes":  ["description", "name", "optical_zoom"],
  "attribute_requirements": {
    "ecommerce": ["description", "name", "optical_zoom"],
    "mobile":    ["name"],
    // ...
  },
  "labels": {
    "en_US": "Camcorders",
    // ...
  }
}
```

Die Erforderlichkeit von Attributen kann für jeden Channel einzeln definiert werden. Für eine Erklärung zu Channels siehe Abschnitt Channel, Currency, Locale.

Attribute & Attribute Group Die verschiedenen Wertausprägungen eines Produktes müssen einem entsprechenden "Attribute" zugeordnet sein. Ein solches Attribut muss zuerst definiert werden, bevor Produkte einen Wert darin aufweisen können. [Aken22g]

```
{
  "code": "auto_exposure",
  "type": "pim_catalog_boolean",
  "group": "technical",
  "localizable": false, // erlaubt andere Werte in anderen Sprachen
  "scopable": false,    // erlaubt andere Werte auf anderen Channels
  "unique": false,      // verhindert die mehrfache Verwendung eines Wertes
  // ... diverse weitere Constraints
}
```

Jedes Attribut kann einer "AttributeGroup" zugeordnet werden. Dies erlaubt das bessere Strukturieren der Attribute. [Aken22g]

Jedes Attribut ist einem `type` zugeordnet, welches den Datentyp anzeigt. Z.B. der Typ `pim_catalog_boolean` zeigt einen booleschen Wert an, `pim_catalog_text` eine Textzeile usw. Zusätzlich können in einem Attribut auch weitere Constraints für die jeweiligen Typen definiert werden. Bspw. kann mit dem Constraint `decimals_allowed` festgelegt werden, ob ein numerisches Attribut Nachkommastellen erlaubt oder nicht [Aken22g]. Auf diese Constraints wird allerdings nicht weiter eingegangen, da sie für diese Arbeit keine Rolle spielen. Die folgende Tabelle zeigt ein Auflistung aller möglicher Typen.

Tabelle 3.1: Attribut-Typen in Akeneo-PIM [Aken22h]

Akeneo-Typ	Erklärung
pim_catalog_identifier	eindeutige Id
pim_catalog_text	Textzeile
pim_catalog_textarea	mehrzeiliger Textblock
pim_catalog_simpleselect	einfache Auswahl aus einer Liste an Optionen

Akeneo-Typ	Erklärung
pim_catalog_multiselect	mehrfache Auswahl aus einer Liste an Optionen
pim_catalog_boolean	boolescher Wert
pim_catalog_date	Datum
pim_catalog_number	einfache rationale Zahl
pim_catalog_metric	rationale Zahl mit Recheneinheit
pim_catalog_price_collection	Preise je Währung
pim_catalog_image	Bild
pim_catalog_file	sonstige Datei (z.B. Datenblatt als PDF)
pim_reference_data_simpleselect	einfache Auswahl einer "ReferenceEntity"
pim_reference_data_multiselect	mehrfache Auswahl von "ReferenceEntities"

Hinweis: "ReferenceEntities" sind in dieser Arbeit ebenfalls nicht verwendet worden und können daher ignoriert werden.

Channel, Currency, Locale Um maximale Flexibilität bei der Verwaltung der Daten zu gewährleisten, können verschiedene "Channels" oder "Scopes" definiert werden. Für jeden "Channel" können die erforderlichen Währungen ("Currencies") und Lokalisierungen ("Locales") hinterlegt werden. [Aken22i]

```
{
  "code": "ecommerce",
  "currencies": ["USD", "EUR"],
  "locales": ["de_DE", "en_US", "fr_FR"],
  "category_tree": "master",
  "labels": {
    "en_US": "E-Commerce",
    // ...
  }
}
```

"Attributes" können als `localizable` definiert werden. Also treten verschiedene Werte je nach gewählter Sprache und Region auf (z.B. der Produktname). Außerdem können sie als `scopable` festgelegt werden. In dem Fall können zu jedem "Channel" zusätzlich nochmal unterschiedliche Werte hinterlegt werden. Bspw. gibt es die Produktbeschreibung in den verschiedenen Sprachen und dazu für jedes Produkt eine kürzere Beschreibung im Channel "mobile" und eine längere im Channel "print". [Aken22g]

Measurement Family Für den Attribut-Typ "pim_catalog_metric" werden eine Zahl und eine Einheit hinterlegt. Jede Einheit wird einer bestimmten "Measurement Family" zugeordnet. Dort sind alle verfügbaren Einheiten sowie deren Umrechnung ineinander hinterlegt. [Aken22g]

3.2.2 Icecat

Akeneo-PIM ist nur eine Software, in der Produkte strukturiert gespeichert und aufbereitet werden können. Tatsächliche Produktdaten müssen aber erst importiert werden. Eine Möglichkeit wäre die Kreation eines komplett künstlichen Sets an Produkten. Das Ziel

dieser Arbeit ist es aber, eine möglichst realitätsnahe Evaluation durchzuführen. Daher wird für den Import von realistischen Produktdaten auf Icecat (https://icecat.biz/de) zurückgegriffen.

Icecat ist ein sehr umfangreicher offener Online-Produktkatalog. Über 33.000 Hersteller veröffentlichen hier die Daten ihrer Produkte. Zum heutigen Stand (April 2022) sind über 11 Mio. Produkte bei Icecat gelistet. Viele dieser Daten sind nur für Premium-Nutzer zugänglich. Allerdings gibt es eine ganze Reihe von Unternehmen (u.a. große Player wie Samsung), die als "Icecat-Sposoren" ihre Produktdaten frei zugänglich zur Verfügung stellen. [Icec22a]

Icecat bietet Herstellern damit die Möglichkeit, sehr schnell ihre neuen Produkte zu verbreiten, sodass Shops und Händler direkt die Daten automatisiert in ihr Sortiment aufnehmen können. Zusätzlich werden die Produktdaten nach der UNSPSC-Taxonomy (eine international geläufige Taxonomie zur Definition von Produkteigenschaften) von Icecat standardisiert. [Icec22b]

Aus diesem umfangreichen Fundus wird ein Datenset zur Beantwortung der Kernfrage erstellt. Icecat ist zudem besonders attraktiv, da es einen Importer für Akeneo gibt (https://marketplace.akeneo.com/extension/akeneo-icecat-connector), welcher den Import von Produktdaten größtenteils automatisieren soll.

3.2.3 Datenset

Anforderungen

Für die Auswahl von Produkten werden zunächst einige Anforderungen definiert:

Die Menge an verschiedenen Attributen könnte sich sehr unterschiedlich auf das Clustering auswirken. Daher sollte es sowohl Produkte mit sehr **wenigen** und mit sehr **vielen** Attributen geben.

Daraus folgt, dass Produkte aus **verschiedenen Kategorien** gewählt werden sollten. Als zusätzliche Herausforderung könnten diese Kategorien am besten **miteinander verwandt** sein, sodass die Fähigkeit einer eindeutigen Trennung der Kategorien überprüft werden kann.

Innerhalb einer Produktkategorien sollten **verschiedene Variationen** des gleichen Produktes sowie **verwandte Varianten** (etwa mehrere verschiedene Generationen des Produktes) auftreten.

Zuletzt sollten **Duplikate** enthalten sein. In realen Datensets kommt häufig dasselbe Produkt mehrmals vor, etwa durch geringfügige Unterschiede der Daten beim Import aus verschiedenen Quellen. Ein Clustering-Verfahren sollte in der Lage sein, solche Duplikate zu erkennen.

Umsetzung

Aus den Anforderungen ist folgendes Vorgehen abgeleitet worden:

Es werden Smartphones der Firma Samsung aus der Galaxy S-Reihe importiert. Hier werden Exemplare der letzten drei Generationen ausgewählt. In jeder Generation gibt es unterschiedliche Varianten (die Standard-Ausführung, “Plus”, “Ultra” und “Fan-Edition”), welche verschiedene Bildschirmgrößen und etwas unterschiedliche Komponenten verbaut haben. [DE22]

Smartphones weisen sehr viele verschiedene Attribute auf und eine einfache Suche auf Icecat zeigte bereits mehrere Duplikate der Samsung-Geräte auf der Plattform.

Als nächstes werden noch Produkte aus der Kategorie “Smartphone-Hüllen” ergänzt. Es werden Hüllen von verschiedenen Herstellern für die besagten Samsung Galaxy Smartphones ausgewählt. Solche Hüllen haben eine viel geringere Menge an Attributen, sind aber trotzdem eng mit den Smartphones im Set verwandt.

Es ist damit sogar denkbar, einige Clustering mit dem Ziel durchzuführen, die Smartphones mit ihren zugehörigen Hüllen in die gleiche Gruppe einteilen zu können.

3.3 Clustering

3.3.1 Clustering-Verfahren

Anforderungen

Um aus den vielfältigen Ansätzen des Clusterings ein geeignetes Verfahren auswählen zu können, werden zunächst einige genauere Anforderungen für die Fragestellung dieser Arbeit definiert:

Ein **hierarchisches Verfahren** ist für diese Arbeit wünschenswert. Es ist bezeichnend, dass Akeneo beliebige Verschachtelungen der “Categories” erlaubt. In der Praxis ist die Einteilung von Produkten in Kategorien stark vom gewünschten Detail-Grad abhängig. Ein hierarchisches Verfahren bildet dieses Verhalten exakter ab. Außerdem müssen partitionierende Verfahren die gesuchte Anzahl an Clustern vorab übergeben bekommen. Damit sind zuerst andere Analysen notwendig, bevor der richtige Wert ermittelt ist. Eine hierarchische Einteilung gibt dem Anwender die Wahl, wie weit aufgefächert die Ergebnisse sein sollen.

Ebenso ist die **Laufzeit** ein wichtiger Faktor, da in der Praxis tendenziell sehr große Mengen an Produkten auftreten können.

Von Vorteil ist weiterhin, wenn das Verfahren mit relativ vielen **verschiedenen Datentypen** umgehen kann. Die meisten vorherigen Transformationen verringern tendenziell den Informationsgehalt der Daten (bspw. die Diskretierung numerischer Attribute in eine geringere Anzahl an Bändern). [KaRo09, Kap. 1.2.6 Mixed Variables]

Ansatz

Agglomerative Verfahren weisen immer mindestens eine quadratische Laufzeit auf [Sibs73]. Diversive Ansätze sind dagegen meistens effizienter (siehe Versuche von [RaRa11]) und liefern tendenziell für uns Menschen besser nachvollziehbare Einteilungen. [King15, Kap. 3.3 Agglomerative versus Divisive Clustering]

Das Verfahren Bisecting K-Means wird in dem Zusammenhang als sehr potent angesehen [StKaKu00]. Allerdings muss dieser Ansatz modifiziert werden, um auch mit gemischten Attributen arbeiten zu können. Da zum K-Means eine Variante für gemischte Attribute verfügbar ist (K-Prototypes [Huan98]), wird dieser Ansatz mit dem "Bisecting"-Prinzip kombiniert. Das eingesetzte Verfahren ist also ein **Bisecting K-Prototypes**. Aus der Literatur ist diese Kombination noch nicht bekannt und damit neuartig. Mit diesem Ansatz können sehr viele Arten von Attributen verarbeitet werden. Mehr dazu im folgenden Abschnitt.

3.3.2 Distanzfunktion

Problemstellung

Grundsätzlich kann K-Prototypes mit gemischten Attributen (numerisch und kategorisch) arbeiten. Die Produktdaten in Akeneo-PIM können allerdings noch weitere Ausprägungen annehmen. Die folgende Tabelle zeigt die grobe Einteilung der Attribut-Typen in Typ-Klassen:

Tabelle 3.2: Einteilung der Akeneo-Typen in übergeordnete Datenklassen

Klasse	Akeneo-Typ
numerisch	pim_catalog_date, pim_catalog_number, pim_catalog_metric, pim_catalog_price_collection
kategorisch	pim_catalog_boolean, pim_catalog_simpleselect, pim_reference_data_simpleselect
multi-kategorisch	pim_catalog_multiselect, pim_reference_data_multiselect
string	pim_catalog_text, pim_catalog_textarea
Datei	pim_catalog_image, pim_catalog_file
sonstige	pim_catalog_identifier

Für die numerischen und kategorischen Attribute muss überprüft werden, ob eine Art "Vorverarbeitung" vor dem Clustering sinnvoll ist. Ansonsten können diese direkt verwendet werden.

Die Klasse "multi-kategorisch" beschreibt die Möglichkeit aus einer gegeben Liste an Optionen mehrere auswählen zu können. Bspw. könnte eine Liste von Materialien gegeben sein und das jeweilige Produkt markiert alle vorkommenden (z.B. Silikon, PET & Glas). Eine solche Datenklasse ist in der Literatur nicht beschrieben. Der Umgang mit diesen Attributen muss also gesondert erarbeitet werden.

Die "string"-Klasse ist streng genommen klassisch "kategorisch". Allerdings handelt es sich hier um freie Textfelder (z.B. der Produkttitel), welche bei jedem Produkt völlig frei gefüllt

werden kann. Somit ist die Betrachtung als “Kategorien” nicht zielführend. Wenn bspw. jedes Produkt einen individuellen Titel hat, dann gibt es genau so viele “Kategorien” wie Produkte. Dies ist keine sinnvolle Form der Datenverarbeitung. Auch für diese Klasse muss eine gesonderte Art des Umgangs gefunden werden.

Die Klasse “Datei” wird im Rahmen dieser Arbeit ignoriert. Die Analyse von Bildern oder Textdokumenten ist ein weites Feld mit vielen verschiedenen Ansätzen. Die tatsächliche Relevanz bspw. eines Produktbildes für das Clustering ist aber fraglich. In der Theorie müssten alle Eigenschaften eines Produktes in den anderen Attributen ebenfalls abgebildet sein. Im Bild liegen diese Information aber äußerst unstrukturiert vor, wenn überhaupt.

Der Akeneo-Typ “pim_catalog_identifier” ist die Id eines Produktes. Sie hat keine Relevanz für das Clustering. Ebenso können in Akeneo Attribute als `unique` gekennzeichnet werden. Dies ist sinnvoll für das Speichern mehrerer verschiedener Identifier (also in Akeneo wird das besagte Identifier-Attribut genutzt, aber die Id im ERP-System ist eine andere, die hier ebenfalls hinterlegt ist). Diese Identifier geben aber keine Ähnlichkeitsinformationen, sondern sind rein logistische Werte. Sie werden im Clustering stets ignoriert werden.

Ein weiteres Problem ist, dass die Definition der Attribut-Anforderungen in den Akeneo-Families keinen bindenden Charakter haben. Das heißt, jedes Produkt kann Werte für jedes beliebige Attribut aufweisen oder auch nicht. Der Umgang mit `null`-Values wird also ebenfalls eine zentrale Rolle einnehmen.

Ansatz

numerische Attribute Viele der numerischen Attribute enthalten ihre Daten auf eine implizite Art und Weise (z.B. ein Datum). Die folgende Tabelle zeigt für die verschiedenen Typen in Akeneo, ob und wie diese Daten extrahiert werden.

Tabelle 3.3: Vorverarbeitung der numerischen Attribute in Akeneo

Akeneo-Typ	Vorverarbeitung
pim_catalog_date	Umwandlung in Unix-Timestamp, Normalisierung
pim_catalog_number	Normalisierung
pim_catalog_metric	Umrechnung in Standard-Unit des Attributs, Normalisierung
pim_catalog_price_collection	Filter nach einer Währung z.B. USD, Normalisierung

Die Verarbeitungsschritte sind recht selbsterklärend. Alle Attribute werden auf das Interval zwischen $[0, 1]$ normalisiert, um eine stärkere Gewichtung von Attributen mit tendenziell höheren Zahlen (z.B. Unix-Timestamps) zu vermeiden. Für die Normalisierung werden nur die tatsächlich vorkommenden Werte genutzt. Werte, welche in den Constraints der Akeneo-Attribute definiert sind (z.B. `date_min` und `date_max`), werden nicht betrachtet.

$$x'^i = \frac{x^i - \min X^i}{\max X^i - \min X^i} \tag{3.1}$$

kategorische Attribute Es könnte bei ordinalen Attributen sinnvoll sein, diese u.U. als numerische Attribute anzusehen. Allerdings geben die Daten in Akeneo keinen direkten Rückschluss her, ob es sich z.B. bei einem Single-Select eigentlich um ein ordinales Attribut handelt. Somit müssten hier alle Attribute (knapp hundert) händisch analysiert werden, was den Rahmen dieser Arbeit weit gesprengt hätte.

Der K-Prototypes verlangt keine spezielle Vorverarbeitung kategorischer Attribute abseits von der Umwandlung der Labels in symmetrische bzw. asymmetrische binäre Werte. Der Akeneo-Typ "Bool" ist theoretisch symmetrisch. Praktisch kann aber in Akeneo jedes Attribut auch `null` sein. Somit können tatsächlich drei Wertausprägungen ($true$, $false$ oder `null`) vorkommen. Daher werden diese Attribute genauso wie alle anderen (z.B. Single-Selects) in asymmetrische binäre Attribute umgewandelt.

Zu beachten ist aber, dass kategorische Attribute auf ihre Ähnlichkeit und numerische auf ihre Distanz überprüft werden. Da die numerischen im Interval $[0; 1]$ liegen, können die Ergebnisse der Ähnlichkeitsmaße (z.B. Jaccard-Koeffizient) einfach invertiert werden.

$$d(x_1^{cat}, x_2^{cat}) = 1 - \frac{|x_1^{cat} \cap x_2^{cat}|}{|x_1^{cat} \cup x_2^{cat}|} \tag{3.2}$$

multi-kategorische Attribute Der naheliegendste Ansatz besteht darin, die verschiedenen auftretenden Kombinationen an gewählten Optionen in eigene Kategorien zu fassen und sie wie normale kategorische Attribute zu behandeln. Dies geht allerdings mit einem Verlust an Informationen einher: Angenommen ein Produkt besteht aus den Materialien $\{\text{silicone}, \text{pet}\}$ und ein anderes aus $\{\text{silicone}, \text{glass}\}$. Beide würden nun unterschiedlichen Kategorien zugeordnet und ihre Ähnlichkeit ist 0, obwohl sie zumindest eines der Materialien gemeinsam haben. Besser wäre, wenn dieses Attribut z.B. mittels Jaccard-Koeffizient analysiert werden würde. Hier würde eine Ähnlichkeit von $\frac{1}{3}$ herauskommen.

Dieser Ansatz ist neuartig und so noch nicht beschrieben worden. Daher wird er im Rahmen dieser Arbeit mit dem naheliegenderem Ansatz verglichen werden.

String-Attribute Für diese Attribute gibt sehr vielfältige Ansätze:

Die einfache Verwendung als kategorische Attribute fällt, wie bereits in der Problemstellung dargelegt, weg.

Eine weitere Möglichkeit wäre der Vergleich mittels String-Metrics (wie Jaro-Winkler oder Levensthein-Distance). Dieser Ansatz birgt ein Hauptproblem: Der K-Prototypes-Algorithmus berechnet für jedes Cluster einen Mittelpunkt. Die String-Metrics sind aber nur für einen paarweisen Vergleich von Strings geeignet. Ein "Mittelpunkt" kann hieraus nicht abgeleitet werden. Die Verwendung des Modus (wie bei kategorischen Attributen) scheidet ebenfalls aus. Wenn jeder Produkttitel individuell ist, dann ist der Modus immer 1. Dieser Ansatz funktioniert für das gewählte Clustering-Verfahren also nicht ohne weiteres.

Schließlich besteht der Ansatz, mit Tokenization zu arbeiten. In der Literatur wird dieser Ansatz aber lediglich für Datensets beschrieben, deren Datenpunkte aus ausschließlich einem String-Wert bestehen. Datensets mit mehreren String-Attributen oder in gemischter Form sind hingegen noch nicht zusammen verarbeitet worden.

Zur Lösung kam die Idee folgende auf: Zuvor wurde ein Ansatz zur Evaluation multi-kategorischer Attribute hergeleitet. Dieser Ansatz könnte für Strings ebenfalls verwendet werden. Zerlegt man einen solchen String in Tokens, so erhält man eine Art multi-kategorischen Wert. Bsp.: aus "Samsung Galaxy S20 128GB" wird $\{\text{samsung}, \text{galaxi}, \text{s20}, \text{128gb}\}$. Solche Tokens lassen sich wieder mittels Jaccard-Koeffizienten vergleichen.

Auch dieser Ansatz wird mit der Verwendung der Strings als einfaches kategorisches Attribute verglichen werden.

Mathematische Formulierung

Aus den genannten Ansätzen lässt sich insgesamt folgende Formell für die finale Distanz-Funktion ableiten:

$$d(x_1, x_2) = \frac{d_{num}(x_1, x_2) + |x_1^{num_{null}}| + |x_2^{num_{null}}| + n_{cat} \cdot d_{cat}(x_1, x_2) + d_{mul}(x_1, x_2)}{|attributes \text{ in } x_1 \cup x_2|} \tag{3.3}$$

$$d_{num}(x_1, x_2) = \sum_{i \in num} |x_1^i - x_2^i| \tag{3.4}$$

$$d_{cat}(x_1, x_2) = 1 - \frac{|x_1^{cat} \cap x_2^{cat}|}{|x_1^{cat} \cup x_2^{cat}|} \tag{3.5}$$

$$d_{mul}(x_1, x_2) = \sum_{i \in mul} \frac{|x_1^i \cap x_2^i|}{|x_1^i \cup x_2^i|} \tag{3.6}$$

Die numerischen Attribute werden mittels Manhatten-Distanz verrechnet. Da die Attribute vorher normalisiert worden sind, kann so maximal eine Distanz von 1 je numerischem Attribut entstehen. Die kategorischen Attribute werden mit dem beschriebenen inversen Jaccard-Koeffizienten berechnet (da asynchron). Der Koeffizient wird außerdem mit der Anzahl an kategorischen Attributen n_{cat} multipliziert, um ihn mit den anderen Metriken gleich zu gewichten. Die multi-kategorischen Attribute werden jeweils einzeln mittels inversem Jaccard-Koeffizienten verglichen.

Wie bereits ausführlich dargelegt, können in den Produktdaten von Akeneo immerzu `null`-Values vorkommen. Die kategorischen und multi-kategorischen Distanzmaße können danke des Jaccard-Koeffizienten problemlos damit umgehen. Die Minkowski-Metriken sind allerdings nur für tatsächlich vorhandene numerische Werte in beiden Produkten definiert. Daher wird d_{num} ausschließlich in diesem Fall aufgerufen. Für den Fall, dass `null`-Values in den numerischen Attributen von einem der beiden Produkte vorkommen, wird für jedes dieser Attribute 1 addiert. Die Attribute sind sich ja maximal unähnlich, da das eine Produkt es definiert und das andere nicht. In der Formel ist das mit $|x_1^{num_{null}}|$ und $|x_2^{num_{null}}|$ hinterlegt.

Der Divisor $|attributes \text{ in } x_1 \cup x_2|$ sorgt für eine Normalisierung der Distanz in das Interval $[0; 1]$. Dadurch werden Unterschiede zwischen Produkten, welche alleine aus der verschiedenen Anzahl an definierten Attributen entstehen, ausgeglichen.

Eine vereinfachte (serialisierte) Form dieser Funktion sieht folgendermaßen aus:

$$d(x_1, x_2) = \frac{\sum d'(x_1^i, x_2^i)}{|attributes \text{ in } x_1 \cup x_2|} \tag{3.7}$$

$$d'(x_1^i, x_2^i) = \begin{cases} 1 & , x_1^i \text{ is } null \vee x_2^i \text{ is } null \\ |x_1^i - x_2^i| & , i \text{ is } numerical \\ 0 & , i \text{ is } categorical \wedge x_1^i = x_2^i \\ 1 & , i \text{ is } categorical \wedge x_1^i \neq x_2^i \\ 1 - \frac{|x_1^i \cap x_2^i|}{|x_1^i \cup x_2^i|} & , i \text{ is } multi - categorical \end{cases} \tag{3.8}$$

3.4 Evaluation

3.4.1 Kriterien

Das hergeleitete Clustering-Verfahren in seinen Varianten sollte auf seine Validität überprüft werden. Speziell wird getestet, ob mit diesem Verfahren "sinnvolle" Cluster gefunden werden. Die verschiedenen Aspekte dieser Sinnhaftigkeit werden nun anhand von drei Hauptkriterien definiert: Stabilität, Qualität und Erkennungsfähigkeit.

Stabilität

Der K-Prototypes-Algorithmus nutzt wie alle Verfahren dieser Klasse ein Initialisierungsverfahren, welches auf dem Zufall beruht. Da es sich um ein klassisches Minimierungsverfahren mit eventuellen lokalen Minima handelt, können mehrmalige Durchläufe über das gleiche Datenset verschiedene Clusterzuteilungen finden. Ein Verfahren, welches keinerlei Determinismus aufweist und je nach Uhrzeit komplett andere Ergebnisse liefert, ist allerdings in der Praxis nicht zu gebrauchen. Zudem sollte eine wohldefinierte Distanzfunktion in der Lage sein, die Datenpunkte eindeutig genug voneinander zu trennen, sodass trotz verschiedener Startpunkte die Zuteilung in Cluster stets ähnlich abläuft.

Zur Prüfung der Stabilität wird also wie folgt vorgegangen: Das Clustering wird stets mehrmals hintereinander ausgeführt. Anschließend wird die Ähnlichkeit der gefundenen Cluster mittels Adjusted-Rand-Index berechnet. Da ein hierarchisches Verfahren verwendet wird, wird jede Hierarchie-Ebene der verschiedenen Durchläufe betrachtet und anschließend der Durchschnitt aus allen Ähnlichkeitsmessungen über alle Hierarchie-Stufen und alle Durchläufe berechnet. Die Ähnlichkeit der Cluster sollte dabei nahe 100% sein.

Qualität

Ziel des Clusterings ist es, ordentlich voneinander getrennte Gruppen zu finden. Dies ist zum einen wichtig, um mit den Ergebnissen überhaupt weiterarbeiten und belastbare Aussagen zu den Gruppierungen finden zu können. Zum anderen sind wohl-separierte Cluster aber auch ein Garant für eine gewisse Resilienz vor neuen Datenpunkten, welche in Zukunft zum Datenset hinzugefügt werden. Schließlich ist es nicht sinnvoll, wenn sich jedes mal bei einem

neu hinzugefügten Produkt die Cluster stark verschieben. Sind sie ordentlich voneinander getrennt, so ist dies unwahrscheinlicher.

Daher wird ein gefundenes Clustering-Ergebnis mittels eines internen Index auf seine Qualität überprüft. Zum Einsatz können entweder der Silhouetten-Koeffizient oder Davies-Bouldin-Index kommen. Um bei der Evaluation keine zusätzlichen Fehlerquellen zu erzeugen, wird für die Berechnung der Metriken auf externe Bibliotheken zurückgegriffen werden. Es wird dabei der Index zum Einsatz kommen, der besser nutzbar ist.

Da das Clustering ein hierarchisches Verfahren verwendet, wird der interne Index für jede sinnvolle Hierarchie-Stufe ($2 \leq k \leq n$) berechnet und anschließend der Durchschnitt gebildet.

Erkennungsfähigkeit

Stabile und wohl-separierte Cluster zu finden, sind sehr sinnvolle Kriterien. Allerdings bewerten sie eher das Clustering an sich ohne einen wirklichen Bezug zur Realität. Dies soll mit diesem dritten Kriterium umgesetzt werden.

Generationen und Modelle In dem erstellten Datenset kommen objektiv erkennbare natürliche "Gruppen" vor. Die verschiedenen Generationen der Smartphones sowie die verschiedenen Modelle innerhalb einer Serie sind für uns Menschen offensichtliche Unterscheidungsmerkmale dieser Produkte. Daher sollte das Clustering-Verfahren ebenfalls in der Lage sein, diese Unterteilung zu finden.

Konkret werden die Generationen und Modelle in den "Akeneo-Categories" hinterlegt werden. Nach einem Cluster-Durchlauf werden anschließend die Hierarchie-Ebenen für die jeweiligen Kategorien abgefragt (z.B. wenn es drei Generationen im Datenset gibt, dann sollte das Clustering bei $k = 3$ jedes Produkt der gleichen Generation in die gleiche Gruppe positioniert haben). Der Vergleich der Übereinstimmung mit "Akeneo-Categories" und Clustering-Ergebnis erfolgt über den Adjusted-Rand-Index.

Duplikate Außerdem befinden sich unter den Smartphones einige Duplikate. Diese werden ebenfalls mit einer entsprechenden "Category" in Akeneo markiert. Anschließend wird geprüft, ob die Duplikate über alle Hierarchie-Ebenen stets dem gleichen Cluster zugeordnet worden sind (was man von einem sinnvollen Clustering-Verfahren erwarten würde). Je nach Implementierung könnte hier der Jaccard-Koeffizient zum Einsatz kommen. Der Vergleich erfolgt nun über die Menge an zugeteilten Clustern beider Produkte. Diese sollte bei beiden sehr ähnlich sein und der Koeffizient entsprechend eine Ähnlichkeit nahe der 1 feststellen.

3.4.2 Aspekte

Für die Evaluation spielen eine Reihe von Teilaspekten bei der Auswertung eine besondere Rolle:

Verarbeitung multi-kategorischer Attribute

Im vorherigen Abschnitt ist ein alternativer Ansatz zur Verarbeitung multi-kategorischer Attribute vorgestellt worden, welcher mit weniger Informationsverlust einhergeht. Daher sollte diese Art der Verarbeitung bessere Cluster produzieren, als wenn die multi-kategorischen Attribute in normale kategorische Attribute umgewandelt werden.

Zusätzlich kann diese Art der Verarbeitung auch auf String-Attribute angewendet werden, welches ebenfalls zu besseren Ergebnissen führen sollte.

Auswahl der Attribute

Produkte in einem PIM-System weisen gerne mehrere hundert Attribute auf. Dadurch stellt sich die Frage, wie sich die Menge an Attributen auf das Clustering auswirkt. Dieser Einfluss wird zum einen mit den verschiedenen Produktfamilien (Smartphones und Hüllen) überprüft. Zum anderen könnte aber auch das gezielte Weglassen von vorhandenen Attributen ebenfalls die Ergebnisse verändern. Dabei sind verschiedene Ebenen denkbar:

Manche Typen von Attributen könnten wichtiger sein als andere. Z.B. stecken in String-Attribute potenziell mehr Informationen als in einem einfachen kategorischen Attribut. Andererseits könnte die "Unstrukturiertheit" der Strings diese Informationen verwaschen. Daher werden die **verschiedenen Typen** in unterschiedlichen Kombinationen geclustert und verglichen.

Über die Families in Akeneo können Attribute als **optional oder erforderlich** gekennzeichnet werden. Attribute mit vielen `null`-Werten könnten für das Clustering womöglich keine Rolle spielen. Daher werden die Cluster mit und ohne Verwendung optionaler Attribute überprüft.

Als letztes wird überprüft, ob eine **menschliche "Vorauswahl"** der Attribute sinnvollere Ergebnisse liefern kann. Um den Rahmen dieser Arbeit nicht komplett zu sprengen, wird der Autor die Auswahl nach persönlichem Ermessen treffen. Es ist aber denkbar, in Zukunft z.B. externe Statistiken heranzuziehen, um vor allem Attribute zu verwenden, welche für Kunden von besonderer Relevanz sind.

Gewichtung der Attribute

In der gesamten bisherigen Herleitung ist stets die "Gleichgewichtung" der Attribute angestrebt worden (z.B. durch Normalisierung der numerischen Attribute etc.). Es ist aber denkbar, dass der komplette Ausschluss von Attributen mit zu viel Verlust an Informationen einhergeht. Daher wird gesondert überprüft, ob sich bessere Cluster bilden, wenn bestimmte Attribute (z.B. die menschliche "Vorauswahl" im vorherigen Absatz) im Verhältnis zu den anderen übergewichtet werden.

Die Übergewichtung lässt sich bewerkstelligen, indem die entsprechenden Attribute im Datenset während der Vorverarbeitung dupliziert werden.

3.4.3 Vorgehen

Die genannten Aspekte werden mit Hilfe der definierten Kriterien und Metriken evaluiert. Das Vorgehen ist dabei wie folgt:

Als erstes werden nur die “Smartphone-Hüllen” geclustert und in den genannten Aspekten überprüft. Da hier die Anzahl an Attributen überschaubar ist, können Fehler in der Implementierung leichter identifiziert und verbessert werden. Auch lässt sich der Einfluss einzelner Attribute auf das Endergebnis leichter bewerten.

Diese initialen Ergebnisse werden nun mit dem Datenset, welches nur aus den “Samsung Smartphones” besteht, gegengeprüft. Hier kommen deutlich mehr Attribute vor, sodass interessant zu sehen sein wird, ob sich die Ergebnisse aus “einfacheren” Produkten auch auf komplexere übertragen lassen.

Schließlich werden alle Produkte (Smartphones und Hüllen) gleichzeitig geclustert mithilfe der zuvor gewonnen Erkenntnisse (bspw. welche Attribut-Typen oder Gewichtungen die besten Ergebnisse geliefert haben etc.). Ziel hierbei ist es, dass die Ergebnisse des Clusterings der Teilsets und des Gesamtsets möglichst identisch sind. Ist das der Fall, so ist das gefundene Clustering-Verfahren tatsächlich “generisch”, da nicht einmal eine vorherige Trennung der Produktfamilien nötig ist.

Eine letzte Teilfrage wird in diesem Zusammenhang ebenfalls evaluiert: Wird das Clustering nur mit Attributen durchgeführt, welche in beiden Familien vorkommen, so könnten u.U. die “passenden” Smartphones und Hüllen in gemeinsame Gruppen sortiert werden. Dieser Versuch stellt eher eine Spekulation dar. Die Überprüfung ist in diesem Rahmen aber leicht durchführbar und bildet damit den Abschluss der Evaluation.

4 Implementierung

4.1 Überblick

Nach der Konzeption folgte die Umsetzung und Implementierung des Praxisteils. Für eine bessere Übersicht ist in der folgenden Grafik die Architektur in detaillierterer Form inklusive implementierter Klassen und sonstigen Software-Komponenten dargestellt.

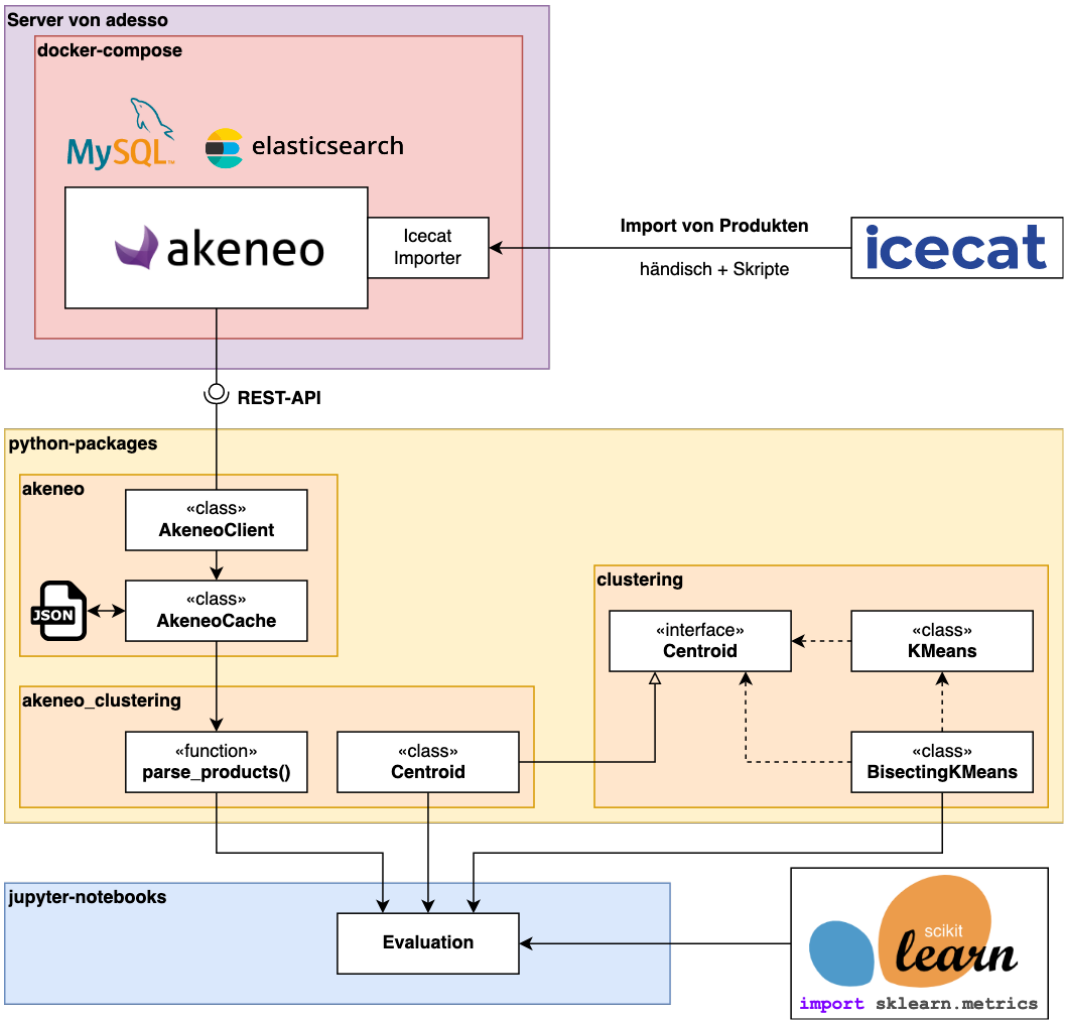

Abbildung 4.1: Detaillierte Architektur der praktischen Umsetzung

adesso stellte für diesen Teil einen Server bereit, welcher für das Hosting von Akeneo-PIM genutzt wurde. Anschließend erfolgte die Installation von Akeneo-PIM Community Edition Version 5.0 auf diesem Server. Dazu wurde die offizielle Installationsanleitung von Akeneo unter Verwendung der Containerization-Lösung Docker (https://www.docker.com/)

befolgt [Aken22d]. Zusätzlich wurde das Akeneo-Plugin "Akeneo Icecat Connector" Version 2.0.0 [Aken22j] in die Instanz integriert. Das kostenpflichtige Plugin wurde von adesso gesponsert.

Nun musste Akeneo-PIM mit Produktdaten gefüllt werden, welche aus dem Online-Katalog Icecat stammen. Eine genaue Übersicht zur Auswahl der Produkte sowie der weiteren Aufbereitung innerhalb von Akeneo siehe Abschnitt Datenset.

Anschließend erfolgte die Implementierung diverser Klassen und Packages, welche das hergeleitete Konzept umsetzen oder wichtige Hilfsfunktionalitäten für die spätere Evaluation liefern. Alle diese Komponenten sind in der Programmiersprache Python (https://www.python.org/) implementiert worden. Python ist sehr weit verbreitet für Aufgaben der Datenanalyse bzw. Data Science allgemein. Das liegt daran, dass es eine Vielzahl nützlicher Bibliotheken für die Analyse und Aufbereitung von Daten gibt. Ebenso sind Implementierungen vieler Algorithmen und Verfahren in Python verfügbar [PWMO19, Kap. 2.4.2 Programmierung]. Zu den meisten umgesetzten Elementen ist ebenfalls grundlegendes Unit-Testing durchgeführt worden, um die korrekte Funktionalität zu prüfen.

Das Package `akeneo` liefert zwei Hauptkomponenten. Der `AkeneoClient` kommuniziert mit der REST-API von Akeneo zum Abrufen der hinterlegten Daten. Ebenso konnten damit verschiedene Aufgaben, wie das Zuordnen der Produkte zu ihren passenden Kategorien teilweise automatisiert werden. Der `AkeneoCache` nutzt den `AkeneoClient`, um alle Endpunkte, welche Daten zurückgeben, abzufragen. Die `JSON`-Responses dieser Anfragen werden anschließend in `JSON`-Dateien gespeichert. Die REST-Anfragen dauern mit unter recht lange (mehrere Sekunden), allerdings änderte sich das Datenset nach der Erstellung nicht mehr. Durch diese Zwischenspeicherung wurde die Evaluation erheblich beschleunigt, da die Daten direkt aus den Dateien gelesen werden konnten. Der `AkeneoCache` ist ebenso für das Laden der Daten zuständig. Dazu werden die `JSON`-Objekte in Python-Datenstrukturen überführt. Dazu kam die Bibliothek `dacite` (https://github.com/konradhalas/dacite) zum Einsatz.

Zunächst wurde überprüft, ob externe Clustering-Bibliotheken in der Lage sind, das erarbeitete Konzept umzusetzen. Da keine Bibliothek den Anforderungen genügte, wurde im Package `clustering` eine generische Version des `KMeans` und `BisectingKMeans` implementiert. Sie nutzen ein allgemeines Interface `Centroid`, welches für eine konkrete Clustering-Anwendung vorher implementiert werden muss. Genauere Details dazu in Abschnitt Clustering.

Das Package `akeneo_clustering` bringt nun die beiden anderen Packages zusammen. Die Funktion `parse_products()` liest die Daten aus dem Cache aus und bereitet die Produkte für das Clustering auf. Dabei werden zum Beispiel die beschriebenen Vorverarbeitungen für numerische und String-Attribute durchgeführt. Die `Centroid`-Klasse implementiert das Interface für das `clustering`-Package. Hier ist die erarbeitete Distanzfunktion hinterlegt, ebenso der genaue Prozess, wie Mittelpunkte zu den jeweiligen Clustern berechnet werden. Weitere Details im Abschnitt Clustering

Abschließend fand die Evaluation mithilfe von Jupyter-Notebooks statt. Das sind Source-Code-Dateien, die in einzelne Code-Zellen unterteilt sind. Mithilfe eines Webfrontends, welches Jupyter mitliefert, können die Notebooks interaktiv verändert und beliebig einzelne oder mehrere Code-Zellen ausgeführt werden. Die Ausgaben der Zellen erscheinen direkt darunter. Dadurch eignet sich Jupyter für Analysen, in denen wiederholt Versuche und

leichte Abwandlungen dieser durchgeführt werden. Die direkte Visualisierung beschleunigt den Evaluationsprozess [PWMO19, Kap. 2.4.2 Programmierung]. Um die Berechnung der Metriken möglichst fehlerfrei durchführen zu können, ist auf die umfangreiche Sammlung implementierter Metriken von scikit-learn zurückgegriffen worden. Näheres im Abschnitt Evaluation.

4.2 Datenset

4.2.1 Attribute

Wie bereits beschrieben, müssen die Attribute in Akeneo zu erst existieren, bevor ein Produkt einen Wert darin aufweisen kann. Icecat definiert ebenfalls eine sehr umfangreiche Liste an Icecat-Attributen. Diese können über den Icecat-Importer ausgewählt und in Akeneo importiert werden. Dabei werden die Icecat-Typen entsprechend auf äquivalente Akeneo-Attribut-Typen abgebildet. Dieser Prozess läuft grundsätzlich automatisch über den Importer. Allerdings müssen dazu vorher per Hand in der Weboberfläche des Importers alle Attribute ausgewählt werden, welche es zu importieren gilt.

Icecat stellt für jede Produktkategorie eine eigene Taxonomie zur Verfügung, welche beschreibt, was für Attribute in den Produkten der jeweiligen Kategorie vorkommen können und ob sie erforderlich oder optional sind. Entsprechend sind die Taxonomien für "mobile_phone_cases" und "smartphones" heruntergeladen worden. Die Excel-Dateien finden sich im angehängt Git-Repository im Ordner `cluster-analysis` > `data` > `icecat-taxonomy__final`. Mittels dieser Dateien sind per Hand im Importer alle benötigten Attribute ausgewählt und importiert worden.

Anschließend sind nach dem Vorbild der Icecat-Taxonomien in Akeneo 2 "Families" (eine für die Hüllen und eine für die Smartphones) angelegt worden. Über die "Family" kann in Akeneo festgelegt werden, welche Attribute bei Produkten dieser Art vorkommen können sowie ob sie erforderlich oder optional sind. Dieser Prozess ließ sich weitestgehend per Skript lösen. Im Repository im Ordner `cluster-analysis` > `reports` > `dataset` sind die ausgeführten Schritte, samt Code in PDF-Dateien hinterlegt.

Es folgten ein paare kleinere weitere Schritte, wie die Strukturierung der Attribute in Gruppen zur besseren Übersicht. Diese sind aber nicht weiter von Belang.

4.2.2 Produkte

Als nächstes sind Produkte für den Import ausgewählt worden. In der kostenlosen Version von Icecat stehen nur Produkte von einigen wenigen Herstellern (sog. Sponsoren) zur Verfügung. Zu diesen Sponsoren zählen auch Samsung und einige Hersteller von Smartphone-Hüllen für Samsung-Smartphones. Über die Suchfunktion von Icecat sind nun Samsung-Smartphones der S-Reihe aus den Generationen S20, S21 und S22 gesucht und ausgewählt worden – ebenso zu den Modellen passende Hüllen. Im angehängten Git-Repository gibt es den Ordner `cluster-analysis` > `data` > `dataset__final`. Dieser enthält mehrere `CSV`-Dateien, welche verschiedene Daten enthalten, die später in Akeneo importiert wurden. Die

Datei `products.csv` enthält alle ausgewählten Produkten mit ihrem jeweiligen Link zu Icecat. Außerdem ist zu jedem Produkt hinterlegt, zu welcher "Family" (Hülle oder Smartphone) es gehört. Außerdem sind die Produkte verschiedenen "Categories" zugeordnet worden. Über diese Categories ist hinterlegt, zu welchem Smartphone-Modell das Produkt gehört. Ebenso ist mit entsprechenden Categories vermerkt, ob zwei Produkte Duplikate voneinander sind. Die folgende Tabelle gibt eine Übersicht über die 122 ausgewählten Produkte:

Tabelle 4.1: Importierte Produkte nach Smartphone-Modell

Modell	Hüllen	Smartphones	davon Duplikate [1]
S20	18	5	2
S20+	14	5	0
S20 Ultra	11	4	2
S20 FE	2	3	0
S21	11	7	4
S21+	10	4	2
S21 Ultra	5	6	2
S21 FE	5	4	0
S22	2	2	0
S22+	1	1	0
S22 Ultra	1	1	0
gesamt:	*80*	*42*	*12*

Schließlich ist aus der Icecat-URL die SKU und EAN der Produkte ausgelesen und per Skript in Akeneo jeweils als neues Produkt eingefügt worden. Danach wurde im Icecat-Importer der `EnrichProducts`-Job ausgeführt, welcher die restlichen noch leeren Attribute der Produkte mit den Daten aus Icecat füllt.

4.2.3 Korrekturen

Nach dem Import der Attribute und Produkte wurden die Daten analysiert und auf Fehler geprüft. Dabei traten eine ganze Reihe von Problemen zutage:

- Alle 31 Attribute, welche in Icecat als Multi-Select definiert sind, wurden in Akeneo als Single-Select importiert. Unter den Labels des Attributs "Material" finden sich z.B. die Werte "Silicone, Thermoplastic polyurethane (TPU)" sowie "Thermoplastic polyurethane (TPU), Silicone". Also die gleiche Information als zwei verschiedene Labels.
- 22 numerische Attribute warfen beim Import Fehler, weshalb diese schließlich als Textzeilen importiert werden mussten.
- 10 Single-Selects sind automatisch als Textzeilen importiert worden.

Diese Fehler wurden anschließend per Skript gefixt. Dazu wurde jedes fehlerhafte Attribut ein zweites Mal in korrigierter Version in Akeneo angelegt. Anschließend wurde jedes Produkt durchgegangen und die Werte aus den fehlerhaften Attributen korrigiert und in den neuen Attributen gespeichert. Die Werte der Multi-Selects enthielten die verschiedenen

[1] Es kommen nur Duplikate von Smartphones vor.

Optionen als Komma-getrennte Liste, welche entsprechend aufgeteilt worden ist. Die fehlerhaften numerischen Attribute und Single-Selects konnten ohne Problem direkt auf den richtigen Typ übertragen werden.

Zum Schluss sind alle fehlerhaften Attribute in eine Attribut-Gruppe names "Faulty" sortiert worden. Dadurch können sie in der Evaluation direkt herausgefiltert werden, sodass nur die korrekten Attribute bei der Verarbeitung übrig bleiben.

Etwas später wurde noch offenkundig, dass alle numerischen Attribute mit einer "MeasurementUnit" ebenfalls fehlerhaft sind. Die Einheit ist falsch importiert worden, sodass bspw. die Smartphones zwischen 5 und 6 Kilogramm wiegen. Da dieser Fehler aber bei allen Attributen "konsistent" aufgetreten ist und die numerischen Attribute im Zuge der Vorverarbeitung sowieso normalisiert werden, kann dieser Fehler ignoriert werden.

Tabelle 4.2: Anzahl an Attributen je Attribut-Klasse und Produktfamilie

Attribut-Klasse	Hüllen		Smartphones		davon in beiden
	erforderlich	*optional*	*erforderlich*	*optional*	
numerisch	1	22	10	106	10
kategorisch	2	16	18	157	1
multi-kategorisch	1	4	3	26	3
strings	6	5	7	27	10
andere	5	0	4	2	4

Die Tabelle zeigt die jeweilige Menge an Attributen für die Hüllen und die Smartphones nach der Durchführung der Korrekturen.

4.3 Clustering

4.3.1 Überblick zu externen Libraries

Für Python existieren eine ganze Reihe an Bibliotheken für das maschinelle Lernen. Diverse Verfahren, welche in der Literatur beschrieben wurden, sind darin praktisch implementiert. Daher ist zuerst geprüft worden, ob eine der existierenden Lösungen bereits für die Clustering-Aufgabe verwendet werden kann.

Die Bibliothek "PySpark" von Apache [Foun22] ist eine der wenigen, die eine Implementierung für den Bisecting-K-Means-Algorithmus zur Verfügung stellt. Das originale Verfahren von Steinbach et al. [StKaKu00] ist nur für numerische Attribute ausgelegt gewesen. Die Implementierung von Apache ist es entsprechend ebenfalls. Auch die Verwendung von `null`-Values im Datenset funktioniert mit dieser Bibliothek nicht, sodass künstliche Werte eingefügt werden müssten, welche vor allem die Berechnung des Durchschnitts für den Centroid verzerren.

Für den K-Means und den K-Prototypes existieren ebenfalls Implementierungen, welche vielleicht als Grundlage für eine eigenen Implementierung in der "Bisecting-Variante" hätten dienen können.

Eine der größten Bibliotheken für das Machine Learning – scikit-learn – bietet nur den klassischen K-Means an und ist ebenfalls nicht in der Lage `null`-Values zu verarbeiten [scik22]. Für das Clustering der Produktdaten ist diese Bibliothek also ebenfalls ungeeignet. Allerdings ermöglicht sie zusätzlich die Berechnung diverser Metriken wie den Adjusted-Rand-Index oder den Silhouetten-Koeffizienten. Diese sind für die Evaluation verwendet worden.

Die einzige gepflegte Implementierung des K-Prototypes stammt von Nico de Vos [Vos22]. Wie alle Verfahren, kann auch diese Variante nicht mit `null`-Values umgehen. Da die Distanzfunktionen per Parameter übergeben werden, hätte die erarbeitete Berechnung der multi-kategorischen Attribute mit dieser Bibliothek wahrscheinlich umgesetzt werden können. Auf die Ermittlung des Mittelpunktes kann aber im Gegensatz dazu kein Einfluss genommen werden. Schlussendlich wurde auch diese Lösung verworfen.

Das Clustering wurden daraufhin selbst auf eine möglichst generische Art und Weise implementiert. Dabei dienten vor allem scikit-learn und die Implementierung von Nico de Vos als Vorbild.

4.3.2 Clustering-Library

`Centroid`-Interface

Das Ziel bei der Implementierung war es, eine möglichst flexible Lösung zu bauen, sodass in der Auswertung beliebige Varianten des Clusterings ausprobiert werden können. Der grundlegende Ablauf des K-Means ändert sich eigentlich nicht. Unterschiede treten vor allem in der Distanzfunktion auf und wie aus den Datenpunkten ein Mittelpunkt errechnet wird. Dadurch entstand die Idee, das Verfahren zu umzusetzen, dass diese Aspekte beliebig ausgetauscht werden können, der Rahmen-Ablauf aber bestehen bleibt.

Es entstand das `Centroid`-Interface, was eine komplett abstrakte Definition eines einzelnen Centroids darstellt. Das Interface verlangt die Implementierung von zwei Methoden:

- `calc_distance(dp: Datapoint) -> float` berechnet den Abstand zwischen dem Centroid und einem Datenpunkt. Auch dieser Datenpunkt ist lediglich als TypeVariable definiert. Es obliegt also der Centroid-Implementierung, welches Format die Datenpunkte haben müssen.
- `on_add_point(dp: Datapoint)` wird aufgerufen, wenn ein neuer Datenpunkt zum Cluster dieses Centroid hinzugefügt wird. Die Methode sollte den internen State des Centroid aktualisieren, sodass entsprechend die `calc_distance`-Methode nach dem Hinzufügen eines Datenpunktes entsprechend andere Ergebnisse liefert, da sich der Centroid verschoben hat.

Die tatsächliche Implementierung dieses Interfaces liegt auf der Seite des Anwenders. So könnte eine Variante für ausschließlich numerische Attribute implementiert werden, welche den euklidischen Abstand für die Distanzberechnung benutzt. Eine andere Implementierung könnte sowohl numerische als auch kategorische Attribute verarbeiten, unterstützt aber keine `null`-Values usw.

Generisches K-Means

Die umgesetzte Version des K-Means nutzt also das `Centroid`-Interface für das Clustering. Dadurch ist es streng genommen kein reiner "K-Means" mehr, sondern abhängig von der Centroid-Klasse eventuell ein K-Modes, ein K-Prototypes oder etwas komplett anderes. Aus Gründen Einfachheit wird stets vom (generischen) K-Means die Rede sein.

Die Umsetzung ähnelt der Version von scikit-learn. `KMeans` ist eine Klasse, welche direkt bei der Instanziierung das Datenset, die implementierte Centroid-Klasse und eventuelle weitere Parameter übergeben bekommt. Mit der Instanziierung wird direkt das Clustering ausgeführt. Die Ergebnisse können anschließend über Attribute der Instanz abgerufen werden.

Es können sehr ähnliche Parameter wie in der scikit-learn-Implementierung für das Clustering gesetzt werden. So kann mit dem Parameter `n_init` festgelegt werden, wie häufig sich der K-Means selbst initialisiert, um anschließend den besten "Lauf" auszuwählen. Über `random_state` lässt sich ein Seed für die zufällige Wahl der initialen Startpunkte festlegen usw. [scik22]

Im angehängten Git-Repository liegt im Ordner `cluster-analysis` > `reports` > `clustering` eine Datei namens `0-algorithm-check.html`. Sie enthält den Export eines Jupyter Notebooks, in welchem die Implementierung von scikit-learn, Nico de Vos und des Autors dieser Arbeit miteinander verglichen werden. Die Verfahren werden mit ähnlichen Parametern aufgerufen und liefern sehr ähnliche Ergebnisse über das gleiche Datenset.

Generisches Bisecting K-Means

Mithilfe des generischen K-Means wurde schließlich die generische Bisecting K-Means-Version implementiert. Auch der `BisectingKMeans` ist eine Klasse, welche direkt mit der Initialisierung das hierarchische Clustering mithilfe des K-Means ausführt. Der Abruf der Ergebnisse erfolgt ebenfalls über Attribute der Klasse. Das folgende Listing zeigt eine mögliche Darstellungsform des Clustering-Ergebnisses:

```
# ...
BisectingKMeans(dataset, Centroid).labels
# Ausgabe:
[
    {0, 1, 2},  # Datenpunkt 1
    {0, 1},     # Datenpunkt 2
    {0},        # Datenpunkt 3
    {0, 3, 4},  # Datenpunkt 4
    {0, 3},     # Datenpunkt 5
]
```

In dem Beispiel wurde ein Clustering über ein Datenset mit fünf Punkten durchgeführt. Jede Zeile listet die Clusterzuordnungen für jeweils einen Datenpunkt. Datenpunkt 1 gehört bspw. den Clustern 0, 1 und 2 an. Da es sich um ein Top-down-Clustering handelt, starten alle Datenpunkte entsprechend initial im Cluster 0. Der erste Split trennte die ersten beiden Punkte vom Rest. Als nächstes wurden Punkt 1 und 2 ebenfalls voneinander getrennt usw.

Es gibt in der Klasse auch eine Methode names `labels_flat(k: int)`. Hiermit können die Clusterzuordnungen für ein spezifisches k abgerufen werden. Entsprechend wäre das Ergebnis für das gegebene Beispiel mit $k = 3$:

```
# ...
BisectingKMeans(dataset, Centroid).labels_flat(3)
# Ausgabe:
[ 2, 1, 0, 0, 0 ]
```

Aus einer `BisectingKMeans`-Instanz kann also die Cluster-Zuordnung für jede beliebige Hierarchie-Ebene abgerufen werden.

4.3.3 Weitere Implementierungen

`Centroid`-Klasse

Mit der Implementierung des `Centroid`-Interfaces konnte nun das hergeleitete Konzept aus Kapitel 3 umgesetzt werden. Die Klasse verarbeitet Datenpunkte in Form einer Hash-Map, welche die "codes" der Attribute aus Akeneo einem numerischen, kategorischen oder multi-kategorischen Wert zuordnet. Das folgende Listing zeigt die Typ-Definition eines solchen Datenpunktes in Python.

```
Datapoint = dict[str, float|str|set[str]]
```

Damit weisen die Datenpunkte eine sehr ähnliche Struktur zu den "ProductValues" in Akeneo auf und bedürfen nur noch geringfügiger Vorverarbeitung.

Die `calc_distance`-Funktion konnte fast eins-zu-eins zu der serialisierten Formel in Kapitel 3 implementiert werden. `on_add_point` aktualisiert den Centroid in den numerischen Werten über den Durchschnitt und in den kategorischen und multi-kategorischen Werten über den Modus. `null`-Values (welche dadurch zustande kommen, dass ein Produkte einige Attribute nicht in seiner Hash-Map aufweist) werden wie beschrieben behandelt.

Datenvorverarbeitung

Im Rahmen der Vorverarbeitung müssen die "Products" aus Akeneo-PIM in die Form des definierten `Datapoint` gebracht werden. Dies geschieht mit Hilfe der `parse_products()`-Funktion. Sie extrahiert die "ProductValues" und bereitet sie entsprechend auf. Je nach Attribut-Typ werden hier verschiedene Schritte durchgeführt:

- Numerische Attribute werden entsprechend der Tabelle in Kapitel 3.3.2 verarbeitet und normalisiert.
- Kategorische Attribute bleiben wie sie sind, denn die serialisierte Version der Distanzfunktion bedarf keiner vorherigen Umwandlung in binäre Attribute.
- Multi-kategorische Attribute sind ebenfalls bereits in der richtigen Form
- Textzeilen werden entsprechend mithilfe der Python-Bibliothek nltk (`https://www.nltk.org/`), welche auf das Natural Language Processing spezialisiert ist, tokenisiert.
- Alle anderen Arten von Attributen werden entfernt.

In dem `akeneo_clustering`-Package existieren noch weitere Hilfsfunktionen, welche hier ber nicht weiter von Belang sind.

Damit sind alle Vorbereitungen für die Evaluation getroffen.

4.4 Evaluation

Die Evaluation erfolgt, wie in der Konzeption festgelegt, zuerst nur mit den Hüllen, dann nur mit den Smartphones und schließlich mit allen Produkten zusammen.

Die Versuche laufen stets nach einem ähnlichen Muster ab: Zuerst werden dazu die jeweiligen Produkte aus Akeneo geladen und vorverarbeitet (wie beschrieben mittels der `parse_products()`-Funktion und weiteren). Dann findet eine kurze Analyse der tatsächlich vorkommenden Attribute statt, welche davon viele `null`-Values enthalten etc. Anschließend werden immer verschiedene Kombinationen an Attributen ausgewählt und das Clustering wird mit dieser Auswahl durchgeführt. Zu den Clustering-Ergebnisse werden die Metriken für die Stabilität, Qualität und Erkennungsfähigkeit berechnet und ausgegeben. Daran können sich entsprechend weiterführende Analysen anschließen.

Die genaue Berechnung der Metriken funktioniert dabei, wie folgt:

- Die *Stabilität* wird berechnet, indem das Clustering mit den gewählten Attributen zehn Mal durchgeführt wird, jedes mal mit einem anderen Seed für die Zufallsinitialisierung. Dann werden die Clusterzuordnungen des ersten Durchlaufes nacheinander mit den neun folgenden Durchläufen verglichen. Der Vergleich erfolgt so, dass mittels der `labels_flat`-Methode des `BisectingKMeans` nacheinander alle Hierarchie-Ebenen von 2 bis $n-1$ abgerufen werden. Für die Clusterzuordnung von zwei verschiedenen Initialisierungen wird nun der Adjusted-Rand-Index (Implementierung aus scikit-learn) berechnet. Man erhält also $9 \cdot n-2$ verschiedene Ergebnisse des Adjusted-Rand-Indexes. Aus all diesen Werten wird der Durchschnitt berechnet. Die Stabilität ist also der durchschnittliche Adjusted-Rand-Index über alle sinnvollen Hierarchie-Ebenen von einem Clustering zu neun weiteren Durchläufen mit exakt den gleichen Parametern.
- Die *Qualität* wird über den Silhouetten-Koeffizienten ermittelt. Die Implementierung in scikit-learn erlaubt die Verwendung dieses Koeffizienten mit eigenen Distanzfunktionen, weswegen diese Metrik dem Davies-Bouldin-Index vorgezogen worden ist. Ähnlich wie bei der Stabilität wird ebenfalls die `labels_flat`-Methode benutzt, um alle Clusterzuordnungen über die verschiedenen sinnvollen Hierarchie-Stufen ($2 \leq k \leq n-1$) zu erhalten. Für jede Stufe wird nun der Silhouetten-Koeffizient berechnet. Anschließend wird der Durchschnitt aller Koeffizienten über alle Ebenen gebildet. D.h. die Qualität ist also der durchschnittliche Silhouetten-Koeffizient eines Clustering-Durchlaufes über alle sinnvollen Hierarchie-Ebenen.
- Die *Erkennungsfähigkeit* wird über den Vergleich der "Akeneo-Categories", wo die Smartphone-Generationen und -Modelle hinterlegt sind, durchgeführt. Über die `labels_flat`-Methode wird die Zuordnung für die jeweilige Stufe abgefragt (Generationen: $k = 3$, Modelle: $k = 11$) und anschließend der Adjusted-Rand-Index zwischen den berechneten und den erwarteten Zuteilungen ermittelt. Hier werden für jeden Versuch also in der Regel zwei Werte für die jeweiligen ks gegeben sein.
- Speziell in den Smartphones kommen zusätzlich noch die Duplikate vor, welche eine Sonderform für die Erkennungsfähigkeit darstellen. Hier ist die Erwartung, dass die Produkte, welche Duplikate von einander sind, über alle Hierarchie-Ebenen (minus der letzten) stets in die gleichen Cluster zugeteilt sein müssten. Wie bereits erklärt, kann aus der `BisectingKMeans`-Klasse zu jedem Datenpunkt die Zugehörigkeit zu allen Clustern im Verlaufe des Prozesses abgerufen werden. Die Menge y_1 sind also

alle Cluster, denen der Punkt x_1 angehört und y_2 entsprechend alle Cluster, denen x_2 angehört. Mit Hilfe eines abgewandelten Jaccard-Koeffizienten kann nun berechnet werden, ob beide Punkte häufig denselben Clustern zugeordnet worden sind:

$$\frac{|y_1 \cap y_2| - 1}{|y_1 \cup y_2| - 1} \tag{4.1}$$

Da die Duplikate im letzten Split ebenfalls in verschiedenen Clustern landen, wird im Zähler und im Nenner entsprechend die 1 für diesen finalen Split abgezogen. Sind beide Datenpunkte, wie erwartet, stets in den gleichen Clustern gewesen, so liegt der Wert des Koeffizienten bei 1. Wurden sie früher getrennt (also andere Produkte wurde als ähnlicher klassifiziert als die Duplikate), liegt der Wert entsprechend niedriger.

Für alle vorkommenden Duplikate wird dieser modifizierte Jaccard-Koeffizient berechnet und anschließend der Durchschnitt aus diesen Koeffizienten gebildet. Der Spezialwert "Duplikate" in der Erkennungsfähigkeit ist als die durchschnittliche Übereinstimmung der Clusterzuteilung aller Duplikate.

Alle Metriken liegen damit im Bereich $[-1; 1]$ (außer die Duplikat-Erkennung, die liegt zwischen $[0; 1]$) und Werte nahe der 1 stellen das beste Ergebnis dar. Werte um die 0 oder sogar tiefer sind entsprechend negativ für den jeweiligen Versuch zu werten.

Außerdem ist wichtig zu erwähnen, dass die Werte der Stabilität nahe der 1 liegen sollten. Andernfalls ist das Clustering nicht deterministisch und die Distanzfunktion trennt die Produkte nicht ausreichend voneinander. Die Qualität sollte auf jeden Fall positiv sein, allerdings ist es ganz normal, dass die Qualität auf bestimmten Hierarchie-Ebenen höher ist und auf anderen niedriger, wodurch im Durchschnitt eher mittelmäßige Werte in dieser Metrik entstehen. Die Erkennungsfähigkeit sollte so hoch wie möglich liegen, da dies ja der intuitiven Einteilung von Menschen entspricht. Liegen die Werte hier unter 0.5 ist die Verwendung in der Praxis wohl eher ausgeschlossen.

5 Auswertung

5.1 Datenset “Smartphone-Hüllen”

5.1.1 Verarbeitung multi-kategorischer Attribute

Tabelle 5.1: Clustering der Hüllen mit multi-kategorischem Attribut

Name	Stabilität	Qualität	Erkennung: Generation	Modell
multi	0.71	0.44	0.02	0.04
multi als single	0.61	0.44	0.00	0.04

Tabelle 5.2: Clustering der Hüllen mit String-Attributen

Name	Stabilität	Qualität	Erkennung: Generation	Modell
strings als multi	0.91	0.24	0.01	0.18
strings als single	0.79	0.15	0.01	0.08

5.1.2 Attribut-Auswahl

Vergleich nach Datentypen

Vergleich nach Erforderlichkeit

Vergleich nach menschlicher Auswahl

5.1.3 Attribut-Gewichtung

5.2 Datenset “Smartphones”

5.2.1 Verarbeitung multi-kategorischer Attribute

Tabelle 5.3: Clustering der Smartphones mit multi-kategorischen Attributen

Name	Stabilität	Qualität	Erkennung: Generation	Modell	Duplikate	Ø
multi	90.5%	38.9%	39.8%	45.2%	89.2%	60.7%
multi als single	85.4%	38.5%	11.7%	30.0%	87.8%	50.7%

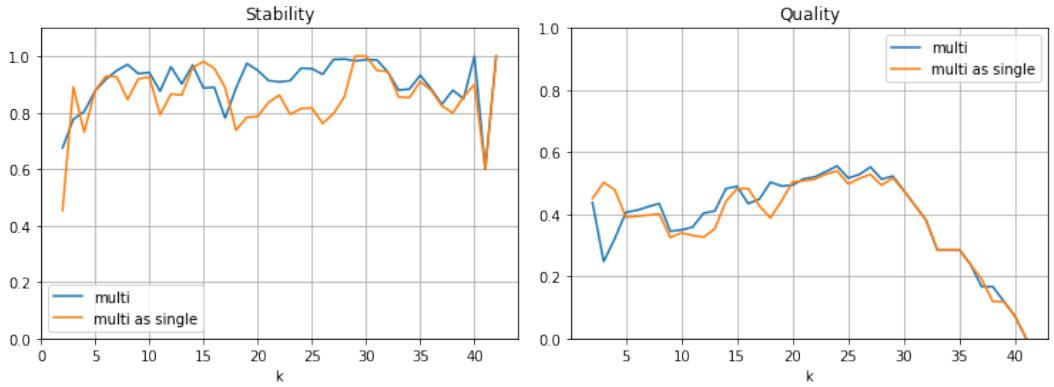

Abbildung 5.1: Stabilität und Qualität der Smartphones mit multi-kategorischen Attributen

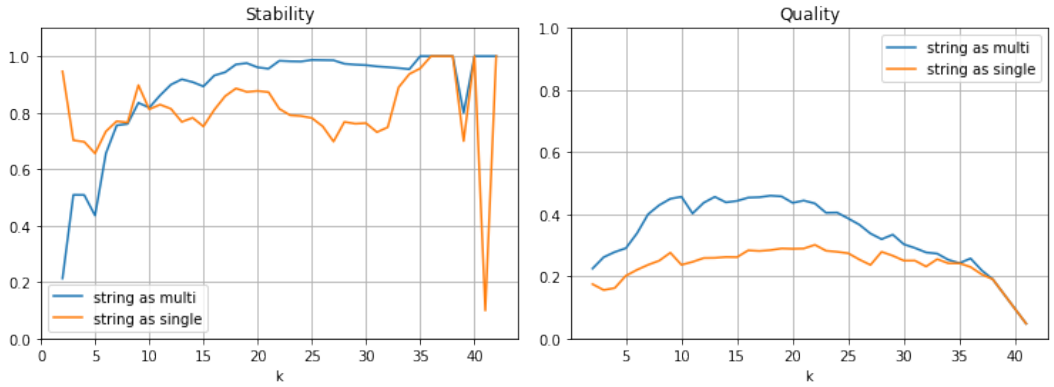

Abbildung 5.2: Stabilität und Qualität der Smartphones mit String-Attributen

Tabelle 5.4: Clustering der Smartphones mit String-Attributen

Name	Stabilität	Qualität	Erkennung: Generation	Modell	Duplikate	Ø
strings als multi	88.2%	34.0%	64.3%	87.2%	100.0%	74.7%
strings als single	80.7%	23.7%	13.8%	58.6%	96.7%	54.7%

5.2.2 Attribut-Auswahl

Vergleich nach Datentypen

Vergleich nach Erforderlichkeit

Vergleich nach menschlicher Auswahl

5.2.3 Attribut-Gewichtung

5.3 Kombiniertes Datenset

5.3.1 Verwendung aller Attribute

5.3.2 Verwendung gemeinsamer Attribute

6 Fazit und Ausblick

Quellenverzeichnis

[Aken22a] Akeneo: *About Akeneo*. URL `https://www.akeneo.com/de/about-us/`. - abgerufen am 2022-04-04

[Aken22e] Akeneo: *The REST API basics – Overview*. URL `https://api.akeneo.com/documentation/overview.html`. - abgerufen am 2022-04-04

[Aken22g] Akeneo: *Concepts & resources – Catalog structure*. URL `https://api.akeneo.com/concepts/catalog-structure.html`. - abgerufen am 2022-04-04

[Aken22h] Akeneo: *REST API endpoints – Create a new attribute*. URL `https://api.akeneo.com/api-reference-17.html#post_attributes`. - abgerufen am 2022-04-04

[Aken22c] Akeneo: *Recommended configuration*. URL `https://docs.akeneo.com/5.0/technical_architecture/technical_information/recommended_configuration.html`. - abgerufen am 2022-04-04

[Aken22j] Akeneo: *Akeneo Icecat Connector | Akeneo App Store*. URL `https://marketplace.akeneo.com/extension/akeneo-icecat-connector`. - abgerufen am 2022-04-14

[Aken22d] Akeneo: *Install Akeneo PIM for development with Docker*. URL `https://docs.akeneo.com/5.0/install_pim/docker/installation_docker.html`. - abgerufen am 2022-04-04

[Aken22b] Akeneo: *Akeneo PIM in a nutshell*. URL `https://docs.akeneo.com/5.0/index.html`. - abgerufen am 2022-04-04

[Aken22f] Akeneo: *Concepts & resources – Products*. URL `https://api.akeneo.com/concepts/products.html`. - abgerufen am 2022-04-04

[Aken22i] Akeneo: *Concepts & resources – Target market settings*. URL `https://api.akeneo.com/concepts/target-market-settings.html`. - abgerufen am 2022-04-05

[ArVa16] Arora, Preeti ; Varshney, Shipra ; u. a.: Analysis of k-means and k-medoids algorithm for big data. In: *Procedia Computer Science* Bd. 78, Elsevier (2016), S. 507–512

[BiHa09] Bissantz, Nicolas ; Hagedorn, Jürgen: Data Mining (Datenmustererkennung). In: *Wirtschaftsinformatik* Bd. 51, Springer (2009), Nr. 1, S. 139–144

[Bosl12] Boslaugh, Sarah: *Statistics in a nutshell: A desktop quick reference* : O'Reilly Media, 2012

[Cha07] Cha, Sung-Hyuk: Comprehensive survey on distance/similarity measures between probability density functions. In: *City* Bd. 1 (2007), Nr. 2, S. 1

[ChZhYi19] Chen, Hongshen ; Zhao, Jiashu ; Yin, Dawei: Fine-grained product categorization in e-commerce. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, S. 2349–2352

[CRF03] Cohen, William W ; Ravikumar, Pradeep ; Fienberg, Stephen E ; u. a.: A Comparison of String Distance Metrics for Name-Matching Tasks. In: *IIWeb*. Bd. 3 : Citeseer, 2003, S. 73–78

[Cui21] Cui, Yimin: Intelligent recommendation system based on mathematical modeling in personalized data mining. In: *Mathematical Problems in Engineering* Bd. 2021, Hindawi (2021)

[DaBo79] Davies, David L ; Bouldin, Donald W: A cluster separation measure. In: *IEEE transactions on pattern analysis and machine intelligence*, IEEE (1979), Nr. 2, S. 224–227

[DE22] DE, Samsung: *Galaxy S Serie*. URL `https://www.samsung.com/de/smartphones/galaxy-s/`. - abgerufen am 2022-04-05

[Defa77] Defays, Daniel: An efficient algorithm for a complete link method. In: *The Computer Journal* Bd. 20, Oxford University Press (1977), Nr. 4, S. 364–366

[DoBi22] Dogan, Alican ; Birant, Derya: K-centroid link: a novel hierarchical clustering linkage method. In: *Applied Intelligence* Bd. 52, Springer (2022), Nr. 5, S. 5537–5560

[Foun22] Foundation, The Apache Software: *BisectingKMeans – PySpark 3.2.1 documentation*. URL `https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.clustering.BisectingKMeans.html`. - abgerufen am 2022-04-14

[Hamm50] Hamming, Richard W: Error detecting and error correcting codes. In: *The Bell system technical journal* Bd. 29, Nokia Bell Labs (1950), Nr. 2, S. 147–160

[Hart75] Hartigan, John A: *Clustering algorithms* : John Wiley & Sons, Inc., 1975

[Huan98] Huang, Zhexue: Extensions to the k-means algorithm for clustering large data sets with categorical values. In: *Data mining and knowledge discovery* Bd. 2, Springer (1998), Nr. 3, S. 283–304

[HuAr85] Hubert, Lawrence ; Arabie, Phipps: Comparing partitions. In: *Journal of classification* Bd. 2, Springer (1985), Nr. 1, S. 193–218

[Icec22a] Icecat: *About Icecat*. URL `https://icecat.com/about-us/`. - abgerufen am 2022-04-05

[Icec22b] Icecat: *Icecat: open feed with product information, data-sheets for ecommerce.* URL `https://icecat.biz/en/menu/manufacturers/index.html`. - abgerufen am 2022-04-13

[Jaro95] Jaro, Matthew A: Probabilistic linkage of large public health data files. In: *Statistics in medicine* Bd. 14, Wiley Online Library (1995), Nr. 5-7, S. 491–498

[JiCh17] Jia, Hong ; Cheung, Yiu-Ming: Subspace clustering of categorical and numerical data with an unknown number of clusters. In: *IEEE transactions on neural networks and learning systems* Bd. 29, IEEE (2017), Nr. 8, S. 3308–3325

[KaRo09] Kaufman, Leonard ; Rousseeuw, Peter J: *Finding groups in data: an introduction to cluster analysis*. Bd. 344 : John Wiley & Sons, 2009

[King15] King, Ronald S: *Cluster analysis and data mining: An introduction* : Stylus Publishing, LLC, 2015

[KoLo12] Kou, Gang ; Lou, Chunwei: Multiple factor hierarchical clustering algorithm for large scale web page and search engine clickstream data. In: *Annals of Operations Research* Bd. 197, Springer (2012), Nr. 1, S. 123–134

[KRRT01] Kumar, Ravi ; Raghavan, Prabhakar ; Rajagopalan, Sridhar ; Tomkins, Andrew: Recommendation systems: A probabilistic analysis. In: *Journal of Computer and System Sciences* Bd. 63, Elsevier (2001), Nr. 1, S. 42–61

[MiCo88] Milligan, Glenn W ; Cooper, Martha C: A study of standardization of variables in cluster analysis. In: *Journal of classification* Bd. 5, Springer (1988), Nr. 2, S. 181–204

[MWDM64] Macnaughton-Smith, P ; Williams, WT ; Dale, MB ; Mockett, LG: Dissimilarity analysis: a new technique of hierarchical sub-division. In: *Nature* Bd. 202, Nature Publishing Group (1964), Nr. 4936, S. 1034–1035

[OhKi19] Oh, Yoori ; Kim, Yoonhee: A resource recommendation method based on dynamic cluster analysis of application characteristics. In: *Cluster Computing* Bd. 22, Springer (2019), Nr. 1, S. 175–184

[Pimc22] Pimcore: *What, why and how of product information management.* URL `https://pimcore.com/en/what-is-pim`. - abgerufen am 2022-04-04

[PWMO19] Papp, Stefan ; Weidinger, Wolfgang ; Meir-Huber, Mario ; Ortner, Bernhard ; Langs, Georg ; Wazir, Rania: *Handbuch Data Science: Mit Datenanalyse und Machine Learning Wert aus Daten generieren* : Carl Hanser Verlag GmbH Co KG, 2019

[RAAQ11] Rendón, Eréndira ; Abundez, Itzel ; Arizmendi, Alejandra ; Quiroz, Elvia M: Internal versus External cluster validation indexes. In: *International Journal of computers and communications* Bd. 5 (2011), Nr. 1, S. 27–34

[Rand71] Rand, William M: Objective criteria for the evaluation of clustering methods. In: *Journal of the American Statistical association* Bd. 66, Taylor & Francis (1971), Nr. 336, S. 846–850

[RaRa11] Rajalingam, N ; Ranjini, K: Hierarchical Clustering Algorithm - A Comparative Study. In: *International Journal of Computer Applications* Bd. 19, Citeseer (2011), Nr. 3, S. 42–46

[Rous87] Rousseeuw, Peter J: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. In: *Journal of computational and applied mathematics* Bd. 20, Elsevier (1987), S. 53–65

[RoZa09] Robertson, Stephen ; Zaragoza, Hugo: *The probabilistic relevance framework: BM25 and beyond* : Now Publishers Inc, 2009

[scik22] scikit-learn: *2.3. Clustering – scikit-learn 1.0.2 documentation.* URL `https://scikit-learn.org/stable/modules/clustering.html`. - abgerufen am 2022-04-14

[Sibs73] Sibson, Robin: SLINK: an optimally efficient algorithm for the single-link cluster method. In: *The computer journal* Bd. 16, Oxford University Press (1973), Nr. 1, S. 30–34

[Stei04] Steinley, Douglas: Properties of the Hubert-Arable Adjusted Rand Index. In: *Psychological methods* Bd. 9, American Psychological Association (2004), Nr. 3, S. 386

[StKaKu00] Steinbach, Michael ; Karypis, George ; Kumar, Vipin: A comparison of document clustering techniques (2000)

[Vos22] Vos, Nico de: *nicodv/kmodes: Python implementations of the k-modes and k-prototypes clustering algorithms, for clustering categorical data.* URL `https://github.com/nicodv/kmodes`. - abgerufen am 2022-04-14

Anhang

- Link zum Repo
- Überblick zum Repo

Selbstständigkeitserklärung

Ich, Hannes Dröse, versichere hiermit, dass ich die vorliegende Masterarbeit mit dem Thema

Generisches Clustern hoch-komplexer Produktdaten

selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Erfurt, den 25.04.2022

.....
Hannes Dröse