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: 29.04.2022

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 5800 Mitarbeiter an 44 Standort 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 mathematischem 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. 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 als auch Cluster bezeichnet.

Dieser Prozess des Clustering 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

Datenpunkte und -set

- Objekte oder Datenpunkte sind Vektoren aus numerischen und/oder kategorischen Attributen: $x = (a|a \text{ is numerical or categorical attribute})$
- Menge aus Datenpunkten x (= Datenset) mit Großbuchstaben angegeben $x \in X$
- i.d.R. $n = |X|$

Cluster

- Cluster sind spezielle Teilgruppen aus dem Datenset $C \subset X$
- Cluster können Datenpunkten oder andere Cluster enthalten: $C = e|e = C_i \vee x \in X$

- leere Cluster gibt es nicht $C \neq \emptyset$
- die Menge aller Cluster ist $K = C_1, C_2, ...$ und $k = |K|$
- Cluster haben häufig einen Mittelpunkt $c = centroid(C)$

Cluster-Zuordnungen

- Vektor aus Labels für jeden Datenpunkt in X: $Y = (l|l_i = \text{ label for } x_i \in X)$

2.3 Distanz- und Ähnlichkeitsmaße

2.3.1 Definition

Clustering erfolgt über Bestimmung der "Nähe" (engl. proximity) der Objekte zueinander. [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

Bestimmung der Nähe mittels Abstands- bzw. Distanzmaßen. Distanzmaß wird über Funktion d dargestellt [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

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

=> [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

Statt Distanzmaße auch Verwendung von Ähnlichkeitsmaßen $s(x, y)$ (engl. similarity) möglich. Ähnlichkeit i.d.R. im Interval $[0, 1]$ angegeben, wobei $s(x, x) = 1$. Wenn Distanz z.B. durch Normalisierung ebenfalls im Interval $[0, 1]$ liegt, dann gilt: $d(x, y) = 1 - s(x, y)$ [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

Distanz und Ähnlichkeit dadurch beliebig austauschbar, deshalb diese Verwendung empfehlenswert.

(Frage: Erwähnung der Proximity Matrix? ist eigentlich nicht wichtig für diese Arbeit)

2.3.2 Numerische Attribute

durch (rationale) Zahlen dargestellt [KaRo09, Kap. 1.2 Types of Data and How to Handle Them], mit stetigen (engl. continuous) Werten [Huan98]. Umfasst damit sowohl Daten in Intervall- und Verhältnisskalen (engl. interval and ratio data) [Bosl12, Kap. 1 Basic Concepts of Measurement]

Minkowski-Familie

Datenpunkte damit numerischen Vektoren also Punkte. Bestimmung des Abstand der Punkte => verschiedene Maße, gehören alle zur Minkowski-Familie [Cha07; und King15, Kap. 1.2 Capturing the Clusters]:

Tabelle 2.2: Übersicht gängiger Maße aus der Minkowski-Familie

Name	p-Norm	Formell
Minkowski	allgemein	$d(x,y) = \sqrt[p]{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p}$
Manhattan	$p = 1$	$d(x,y) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
euklidisch	$p = 2$	$d(x,y) = \sqrt{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^2}$
Chebyshev	$p = \infty$	$d(x,y) = \max \lvert x_i - y_i \rvert$

höhere p-Norm bedeutet i.d.R. robustere Bestimmung des Abstands. Manhattan => Winkel zwischen zwei Punkten, euklidisch => Länger der Geraden durch die Punkte [King15, Kap. 12.3 Which Proximity Measure Should Be Used?]

verschiedenste Versionen und Abwandlungen dieser Maße [siehe Cha07]

- aber die meisten basieren auf Minkowski-Familie
- oder lassen sich auf Minkowski abbilden (z.B. Maße basierend auf der Pearson Korrelation nutzen im Kern euklidischen Abstand zur Berechnung)
- liefern ähnliche Ergebnisse wie klassische Minkowski
- [siehe Cha07] Clustering mit verschiedenen Maßen und dann Ergebnisse geclustert. (eigene Erkenntnis aus dem Paper) im Kern jeder Cluster Gruppe war auch ein Vertreter der Minkowski-Familie

Normalisierung

manche numerischen haben logarithmischen Zusammenhang (10 zu 20 ist gleichbedeutend mit 100 zu 200), dann $x'_i = \log x_i$ für gleichmäßige Abstände

Projektion auf $[0, 1]$ => $x' = \frac{x - \text{avg } X}{\max X - \min X}$; empfehlenswert, wenn Attribute gleichgewichtet sein sollen

manchmal ist Normalisierung nicht sinnvoll, da dadurch verschiedene “Gewichtung” der Attribute bestimmtes Wissen dargestellt wird => erfordert Domänenwissen

Eventuell sogar arbeiten mit Gewichtsvektor für die Attribute => eher spezielle Anwendungsfälle

gesamter Abschnitt [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

2.3.3 Kategorische Attribute

statt Zahlen bestehen Datenpunkte aus ihnen zugeordneten Kategorien oder Labels [Bosl12, Kap. 5 Categorical Data]

Manchmal gibt es eine sinnvolle Reihenfolge der Labels (z.B. xs, s, m, l, xl), dennoch keine Aussage über Verhältnis oder Intervall zueinander möglich. mit Reihenfolge als ordinal bezeichnet, ohne als nominal [Bosl12, Kap. 1 Basic Concepts of Measurement, Kap. 5 Categorical Data]

Ordinale Attribute

zwei Möglichkeiten:

Reihenfolge ignorieren und als nominal verarbeiten => siehe folgende Abschnitte

aber eher empfohlen: Umwandlung und Verarbeitung als numerische Attribute:

- Nummerierung der Labels nach ihrer Reihenfolge von 1 bis n
- dann: $x' = \frac{x-1}{n-1}$
- dadurch im Interval $[0, 1]$ in $n - 1$ gleichmäßige Abschnitte eingeteilt

ganzer Abschnitt [KaRo09, Kap. 1.2 Types of Data and How to Handle Them]

Nominale Attribute

dieser Teil aus [KaRo09, Kap. 1.2.5 Nominal, Ordinal, and Ratio Variables]

Umwandlung in sog. binäre Attribute:

1. bei 2 Ausprägungen => 0 = erste Ausprägung, 1 = zweite Ausprägung z.B. yes/no
2. ab 3 =>

- entweder in 2 Ausprägungen komprimieren
- oder ein neues Attribut pro Kategorie definieren: 0 = gehört nicht zur Kategorie, 1 = gehört zur Kategorie

folgende aus [KaRo09, Kap. 1.2.5 Binary Variables]

für 1. => sog. “symmetrische” binäre Attribute, 2. => “asymmetrische”; Verarbeitung mit speziellen Ähnlichkeitsmaßen:

Simple matching $s(x, y) = \frac{|x \cap y| + |\bar{x} \cap \bar{y}|}{|x \cup \bar{x} \cup y \cup \bar{y}|}$

- jeder Match (beide Datenpunkte haben Label und beide Punkte haben ein Label **nicht**) wird gezählt
- nur für symmetrische geeignet
- gibt Varianten davon mit unterschiedlichen Gewichten für Matches usw. => lassen sich alle auf Simple matching abbilden

Jaccard-Koeffizent $s(x,y) = \frac{|x \cap y|}{|x \cup y|}$

- nur die tatsächlich vorhandenen Attribute (mit 1 codiert) werden miteinander verglichen
- nicht-vorhandene Attribute bei beiden Datenpunkten werden ignoriert
- gleiches Bsp:
  - 3 Farben (rot, grün, blau)
  - x ist rot; y ist blau
  - Simple matching würde $\frac{1}{3}$ ergeben, weil ja beide gemeinsam haben **nicht** grün zu sein
  - Jaccard hingegen sagt Ähnlichkeit $\frac{0}{2}$, weil keines von zwei vorhandenen Attributen matched

2.3.4 String-Attribute

freie Textfelder

Tokenization

[CRF03]

Bei nominalen Attributen mit vielen Ausprägungen (bsp Produkttitel => jeder ist anders)

- Zerlegen in einzelne Wörter => Tokenization
- Rückführung der Wörter auf ihren Stamm (z.B. Porter-Stemming)
- Entfernen von Füllwörtern => Stop-Word Removal

Am Ende erhalten wir eine feste Menge an Tokens oder Keywords => Menge kategorischen Daten => asymmetrisch binär verarbeiten

oder auch Überführung in Vektor-Space ...

String-Metrics

[CRF03] [RaRa11]

Alternativ kann Abstand auch mittels String-Metrics ermittelt werden.

String-Metrics messen die Ähnlichkeit zweier Strings

Bsp:

- Levenshtein: Anzahl an Buchstaben eingefügt/geändert/gelöscht um x auf y umzuwandeln
- Hamming: Anzahl an **ungleichen** Buchstaben => beide Strings müssen gleich lang sein
- Jaro: Verhältnis gleicher zu verschiedenen Buchstaben, performant mit guten Ergebnissen

- Jaro-Winkler: Erweiterung von Jaro mit höherem Gewicht auf den Anfang (Prefix) der Wörter

Verwendung:

- entweder String-Metrics selbst als Distanzmaß nutzen
- oder "ordinale" Reihenfolge mittels String-Metrics bestimmen und dann als ordinal betrachten

2.3.5 Gemischte Attribute

[KaRo09, Kap. 1.2.6 Mixed Variables]

verschiedene Ansätze:

Umwandlung in numerische Werte

- siehe ordinal
- siehe nominal => String-Metrics

Umwandlung in kategorische Werte

- ordinal als nominal betrachten (siehe Ordinale Attribute)
- numerisch Umwandeln durch Diskretisierung
  - Bildung von festen Bändern/Gruppen z.B. Alter => $10-19$, $20-29$ etc.

Separates Clustern je Attribut-Klasse und Subspace-Clustering

mehrmaliges Clustern mit jeweils nur Attributen eines Skalenniveaus

anschließend Vergleich der verschiedenen gebildeten Cluster

noch weitere komplexere Ansätze, wo verschiedenste Kombinationen an Attributen für das Clustern ausgewählt werden und verglichen werden => "Subspace-Clustering" [siehe JiCh17]

Kombinierte Distanzfunktion

Bewertung jedes Attributs mit geeignetem Distanzmaß. Anschließend zusammenrechnen mit Gleichgewichtung

ab hier aus [Huan98]

z.B. bei k-Prototype werden numerische und kategorische mit jeweils geeigneten Distanzmaßen verrechnet:

$d(x,y) = wd_numerical(x,y) + (1-w)d_categorical(x,y)$, also z.B. euklidisch und simple matching oder Manhattan und Jaccard etc.

w soll eine Gleichgewichtung erzeugen also z.B. $\frac{Anzahl numerischer Attribute}{Anzahl aller Attribute}$

2.4 Clustering-Verfahren

2.4.1 Partitionierendes Clustering

Überblick

Minimierungsproblem: initiale Cluster-Zuordnungen (Partitionen), dann Veränderungen der Cluster-Zuordnungen bis ein lokales Minimum gefunden worden ist. [King15, Kap. 4.1 Introduction]

Verschiedene Algorithmen und Varianten: initiale Selection der Cluster, Wahl der Distanzmaße, Vorgehen während der Minimierung [King15, Kap. 4.1 Introduction] und für welche Arten von Attribute geeignet, dazu später mehr [siehe Huan98]

Effizienz:

- neigt zu lokalen Minima => mehrmaliges Wiederholen des Clustering mit verschiedenen zufälligen Startpunkten
- dennoch sehr effizient lösbar: $\mathcal{O}(n \cdot k \cdot l)$
- n Datenpunkte, k Cluster, l Wiederholungen des Clusterings
- k und l kleine Werte, viel kleiner als n
- also: $\mathcal{O}(n)$
- [Huan98]

Aber: k muss vorher bekannt sein => verschiedene k probieren oder z.B. mit hierarchischem Verfahren k abschätzen [King15, Kap. 4.1 Introduction]

k-Means

klassischster Vertreter des Clusterings und weit verbreitet und genutzt [Huan98]

Jedes Cluster wird über einen Schwerpunkt (Mittelpunkt) repräsentiert [StKaKu00]. Es gilt die Datenpunkte so den Clustern zuzuordnen, dass die Summe der Abstände der Datenpunkte zum Mittelpunkt so klein wie möglich sind. Schwerpunkt wird über Mittelwert (engl. mean) der Clustermitglieder bestimmt. [King15, Kap. 4.5 K-Means Algorithm]

Funktioniert daher nur mit numerischen Werten [Huan98]

Ablauf [King15, Kap. 4.5 K-Means Algorithm]:

1. Wahl der initialen k Startpunkte
2. Zuordnen aller Datenpunkte zum nächstliegenden Schwerpunkt
3. Neuberechnung der Schwerpunkte mittels Durchschnittswert (engl. mean) der Datenpunkte
4. ab 2. wiederholen, solange bis keine/kaum noch Änderungen der Schwerpunkte

Für Wahl der initialen Schwerpunkte [King15, Kap. 4.3 The Initial Partition]:

- ersten k Datenpunkte
- oder gleichmäßig aus der gesamten Liste
- oder zufällig über gesamte Liste
- und weitere

z.B. mehrmals mit verschiedenen random Startpunkten durchführen und bestes Ergebnis nehmen => verteilte Berechnung möglich

Statt die Schwerpunkte nur einmal am Ende eines Durchlaufes neuberechnen (Forgy's Method) auch permanente Neuberechnung (mit jedem neu zugeordnetem Datenpunkt) möglich (MacQueen's Method) und weitere Abwandlungen [King15, Kap. 4.5 K-Means Algorithm]

k-Medoids

[ArVa16]

- Variante des k-Means
- median statt mean als Clusterschwerpunkte
- median Berechnung komplexer, da Abstand aller Punkte eines Clusters zueinander berechnet werden muss
- aber laut Autor bessere Ergebnisse, schnellere Konvergenz und dadurch trotzdem kürzere Rechenzeit

k-Modes

[Huan98]

Variante des k-Means für kategorische (Attribute)

Änderungen zum k-Means:

- Simple matching als Distanzmaß (allerdings auch andere z.B. Jaccard möglich)
- statt Durchschnittswertes (engl. mean) wird Clusterschwerpunkt aus dem Modus (engl. mode) bestimmt
- Update der modes eines Clusters während des Clustering anhand ihrer Häufigkeit (Frequenz)

nach einem Durchlauf, Set erneut durchgehen und checken, ob einige Punkte nicht einem anderen Cluster zugeordnet werden müssen, da sich Modus des Datensets während des Clusterns ja häufig ändert. nach Reassign => Modus Frequenz in beiden Clustern anpassen

k-Prototypes

[Huan98]

Kombi aus k-means und k-modes für gemischte Datensets

nutzt ein Distanzmaß für numerische und eins für kategorische Werte + Gewicht zur gleichgewichtung der Attribute siehe gemischte Attribute

Schwerpunkt der numerische Attribute ist der Durchschnitt (mean) und der nominalen ist der Modus (mode) => also Kombi aus beiden

2.4.2 Hierarchisches Clustering

Überblick

!!!QUELLEN!!!

schrittweise Zuordnung zu Clustern => top-down (divisive) oder bottom-up (agglomerative)

Visualisierung: Dendogramm

Vorteil: k muss vorher nicht bekannt sein, sondern Hierarchie mit beliebiger Feinheit, Erkenntnisse aus Kind- und Vater-Clustern

Nachteile: einmal kombiniert/geteilt kein Reassignment der Cluster mehr, teilweise seltsame Ergebnisse, Fehler ziehen sich immer weiter durch

rechenintensiv: Vergleich von jedem Datenpunkt mit jedem anderen Datenpunkt, also mindestens $\mathcal{O}(n^2)$ meistens noch aufwendiger, da Datenset k mal durchsucht werden muss. Da hierarchisch ist $k \approx n$, also meistens $\mathcal{O}(n^2 \log n)$ (mit Optimierungen) bzw. $\mathcal{O}(n^3)$

Agglomeratives Clustering (Bottom-Up)

!!!QUELLEN!!!

Ablauf:

1. alle Datenpunkte in einem eigenen Cluster
2. Ermittlung der beiden am nächsten gelegenen Cluster
3. Fusion dieser beiden Cluster zu einem größeren
4. ab 2. wiederholen bis alle Punkte in einem Cluster sind

Theoretisch könnte auch vorher abgebrochen werden, aber für den Anwender sind i.d.R. die obersten "größten" Cluster-Hierarchien am interessantesten

zentrale Frage: wie werden zwei Cluster mit mehreren Datenpunkten miteinander verglichen => Linkage

Bsp immer die beiden dichtesten Datenpunkt beider Cluster => Single-Link

Name	Formell	Beschreibung	Eigenschaften
Single-Link	$d(X,Y) = \min d(x_i, y_j)$	Abstand durch die beiden dichtesten Punkte definiert	neigt zur Bildung von langen Ketten (chaining effect), anfällig gegen Outlier
Complete-Link	$d(X,Y) = \max d(x_i, y_j)$	Abstand durch die beiden am entferntesten liegenden Punkte definiert	neigt zur Bildung von vielen sehr kleinen Clustern, anfällig gegen Outlier

Name	Formell	Beschreibung	Eigenschaften
Average-Link	$d(X,Y) =$ avg $d(x_i, y_j)$	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
Mediod-Link	$d(X,Y) =$ ...	Median Punkt eines Cluster repräsentiert das Cluster	ähnliches Verhalten wie Average-Link, aber effizienter zu berechnen, da der Median eines neu gebildeten Clusters nur einmal bestimmt werden muss
k-Centroid-Link ???	$d(X,Y) =$ ...	vergleicht den Durchschnitt der k zentralsten Punkte der Cluster miteinander; allgemeine Form von Mediod-Link ($k = 1$) und Average-Link ($k = \|C\|$)	liegt je nach k irgendwo zwischen Average- und Mediod-Link; k ist ein weiterer Parameter, der festgelegt werden muss
Ward-Method	$d(X,Y) = \frac{d(\bar{x},\bar{y})^2}{\frac{1}{\|X\|}+\frac{1}{\|Y\|}}$	berechnet die Varianz für jedem potenziellen Merge, Merge mit höchster Reduktion der Varianz wird ausgewählt	sehr aufwendig zu berechnen, da immer wieder jeder Punkt mit jedem anderen verrechnet werden muss

Insgesamt alle sehr rechenintensiv. single- und complete-link bei guter Implementierung in $\mathcal{O}(n^2)$, alle anderen mindestens in $\mathcal{O}(n^2 \log n)$ oder $\mathcal{O}(n^3)$ (vor allem bei Average-Link und Ward-Method)

Diversives Clustering (Top-Down)

Überblick ergibt für Menschen intuitivere Cluster, da wir ebenfalls mental so vorgehen [King15, Kap. 3.3 Agglomerative versus Divisive Clustering]

theoretisch weniger aufwendig als agglomerativ, wenn nicht komplette Hierarchie generiert wird => weniger Durchläufe da i.d.R. $k < n - k$ (von 1 Cluster zu k weniger Schritte als von n zu k Clustern)

Problem: erstes Cluster riesig groß mit theoretisch $2^{n-1} - 1$ möglichen Arten des Splitterns (also $\mathcal{O}(2^n)$)

in der Praxis wird dieses Problem durch geschicktes Vorgehen umgangen:

DIANA [KaRo09, Kap. Divisive Analysis (Program DIANA)]

klassisches, erstes Verfahren, wie Abspaltung bei einer Partei

Ablauf:

1. Start mit allen Punkten in einem großen Cluster
2. Berechnung der durchschnittlichen Abweichung aller Punkte voneinander
3. Punkt mit größter Abweichung => Startpunkt einer "Splittergruppe"
4. Berechnung der durchschnittlichen Abweichung ohne Punkt mit größter Abweichung
5. Differenz der beiden Abweichungen größer geworden und am größten => Punkt der "Splittergruppe" zuordnen
6. wiederholen 4. bis Differenzen nur noch negativ
7. Durchmesser der Cluster berechnen $\emptyset = \max d(x, y)$
8. Cluster mit größtem Durchmesser verwenden und ab 2. wiederholen, bis gewünschte Abbruchbedingung erreicht ist

Abbruchbedingung kann bestimmter Durchmesser sein oder ein festes k, spätestens Schluss, wenn $k = n$

Laufzeit (eigene Überlegung; kleiner Kommentar dazu in [StKaKu00])

- Berechnung der durchschnittlichen Abweichung: jeder Punkt mit jedem anderen vergleichen => $\mathcal{O}(n^2)$
- da Cluster immer kleiner werden, wird Berechnung mit jedem Schritt einfacher

[RaRa11] Versuche mit agglomerativ und divisive nach DIANA => DIANA immer doppelt so schnell fertig für das gesamte Datenset

Bisecting k-Means [StKaKu00]

zur Teilung des größten Clusters wird k-Means mit $k = 2$ genutzt

- dadurch viel effizienter da k-Means in $\mathcal{O}(n)$
- k-Means wird zwar k mal aufgerufen, aber gleichzeitig wird die Anzahl der Datenpunkte je Cluster mit jeder Teilung kleiner

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 2.4: 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]

$$rand = \frac{a+d}{a+b+c+d} = \frac{a+d}{\binom{n}{2}} \tag{2.1}$$

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 zu gewiesen 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 adequat 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])

$$arand = \frac{rand - rand_{expected}}{rand_{max} - rand_{expected}} \tag{2.2}$$

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]

$$arand = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]} \tag{2.3}$$

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 Formell zeigt eine breits vereinfachte Umstellung von Steinley. [HuAr85; und vereinfachte Formell 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 der zu erwartenden Übereinstimmung aufgrund von Zufall entspricht (oder sogar deutlich darunter liegt) und damit nicht signifikant ist. Höhere Zahlen nahe der 1 stehen nachwievor für eine perfekte Übereinstimmung, welche zusätzlich statistische Signifikanz aufweist. Auch der Bias zu hohen Werten bei einer hohen Anzahl an Clustern wird daurch 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.

$$s(x) = \frac{b(x) - a(x)}{\max a(x), b(x)} \tag{2.4}$$

$$a(x) = d(x, C_x) \tag{2.5}$$

$$b(x) = \min_{C_x \neq C_i} d(x, C_i) \tag{2.6}$$

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.

$$sil = \frac{1}{n} \sum_{i=1}^{n} s(x_i) \tag{2.7}$$

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 andere 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.

$$dbi = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} \frac{d(c_i, C_i) + d(c_j, C_j)}{d(c_i, c_j)} \tag{2.8}$$

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 Verechnung 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 des Kernfrage wird ein Konzept für das Clustern komplexerer Datenstrukturen erarbeitet und anschließend evaluiert. Da diese Fragestellung sehr vielfältig angegangen werden kann, werden zuerst einige Grundanforderungen definiert, welche den Rahmen effektiv einschränken sollen:

Diese Arbeit soll nicht nur eine theoretische Herleitung sein. Die Überprüfung des Konzeptes soll an einem möglichst realen Beispiel mit realitätsnahen Daten erfolgen. Daher wird eine Instanz eines populären Product-Information-Management-Systems angelegt und mit entsprechenden authentischen Produktdaten gefüllt.

Parallel wird ein Konzept für ein Clustering-Verfahren erarbeitet, welches die vielfältigen Datenstrukturen verarbeiten kann. Zu beachten dabei ist die "Praxistauglichkeit" des Clustering-Verfahrens. Konkret bedeutet das, dass bei der Auswahl des Verfahrens, speziell auf die Laufzeit und Effizienz geachtet wird. In Online-Shops kommen in der Praxis oft riesige Mengen an Produktdaten vor, sodass die Rechenzeit für das Clustering nicht ins unermessliche steigen kann.

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 wird im Rahmen des Konzeptes ebenfalls erarbeitet werden.

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 3.1: 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.

TODO: Bild von Akeneo-PIM Webfrontend

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. Produkten müssen diese aber nicht erfüllen. Das Produkt weißt 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 dem entsprechenden Abschnitt.

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 besser 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]. Die folgende Tabelle zeigt ein Auflistung aller möglicher Typen.

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

Type	Erklärung	Constraints
pim_catalog_identifier	eindeutige Id	max_characters, validation_rule, validation_regex
pim_catalog_text	Textzeile	max_characters, validation_rule, validation_regex
pim_catalog_textarea	mehrzeiliger Textblock	max_characters
pim_catalog_simpleselect	einfache Auswahl aus einer Liste an Optionen	
pim_catalog_multiselect	mehrfache Auswahl aus einer Liste an Optionen	
pim_catalog_boolean	Ja/Nein	default_value
pim_catalog_date	Datum	date_min, date_max
pim_catalog_number	einfache Zahl	number_min, number_max, decimals_allowed, negative_allowed
pim_catalog_metric	Zahl mit Recheneinheit	number_min, number_max, decimals_allowed, negative_allowed
pim_catalog_price_collection	Preise je Währung	number_min, number_max, decimals_allowed
pim_catalog_image	Bild	allowed_extensions, max_file_size
pim_catalog_file	sonstige Datei (z.B. Datenblatt als PDF)	allowed_extensions, max_file_size
pim_reference_data_simpleselect	einfache Auswahl einer “ReferenceEntity”	
pim_reference_data_multiselect	mehrfache Auswahl von “ReferenceEntities”	

(FRAGE: Die Constraints spielen eigentlich keine weitere Rolle, daher wäre es wahrscheinlich sinnvoller die wegzulassen, oder?Die Tabelle ist so einfach sehr unübersichtlich, finde ich)

Hinweis: “ReferenceEntities” sind in dieser Arbeit 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

Für den Import von realistischen Produktdaten wird auf Icecat (https://icecat.biz/de) zurückgegriffen. Icecat ist ein sehr umfangreicher offener online Produktkatalog. Über 33.000 Firmen 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. [Icec22]

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 **verschiedenen 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 Anforderung 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 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 verringert 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 [TODO:quelle]. Diversive Ansätze sind dagegen meistens effizienter (siehe Versuche von [RaRa11]) und liefern tendenziell für uns Menschen 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

Klasse	Akeneo-Typ
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, somit gibt es genauso viele "Kategorien" wie es Produkte gibt. Dies ist keine sinnvolle Form der Datenverarbeitung. Auch für diese Klasse muss eine gesonderte Art der Verarbeitung 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 Character 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 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 Der K-Prototypes verlangt keine spezielle Vorverarbeitung kategorischer 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.

Zu beachten ist lediglich, 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 eine 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.

Die Idee ist die folgende: 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 und harmonisiert die Endungen (Stemming), 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 ist so in der Literatur noch nicht beschrieben und wird in dieser Arbeit verwendet und evaluiert.

Mathematische Formulierung

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

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

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

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

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

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. 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.

Da immerzu `null`-Values vorkommen können, ist wichtig zu erwähnen, dass d_{num}, d_{cat} und d_{mul} nur dann aufgerufen werden, wenn beide Produkte einen Wert ungleich `null` im Attribut definiert haben. Mit dem Ausdruck $|x_1^{null}|$ ist die Anzahl an Attributen gemeint, die in x_1 `null` sind, in x_2 aber einen Wert aufweisen und umgekehrt. Je weniger Attribute von beiden Produkten gemeinsam gefüllt werden, desto höher ist damit die Distanz durch die letzten beiden Summanden. Der Grund für dieses Vorgehen ist, dass Produkte als unterschiedlich angesehen werden, wenn sie kaum oder keine Attribute gemeinsam haben.

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

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

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

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

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. Die verschiedenen Aspekte dieser Sinnhaftigkeit werden nun anhand von drei Hauptkriterien definiert: Stabilität, Qualität, 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 verschiedenen Startpunkten die Zuteilung in Cluster stets ähnlich ablaufen sollte.

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 betrachten 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 einem 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 natürlich "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-kategorischer 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" in 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

- Akeneo-PIM Instanz auf adesso VM
- AKeneo Client und Cache
- Import von Daten aus Icecat
- Implementierung Clustering-Bibliothek + weitere (z.B. Datenaufbereitung)
- Durchführung des Clusterings selbst

4.2 Akeneo-PIM

4.2.1 Installation und Deployment

- Überblick zur Projekt-Initialisierung
- Installation auf adesso VM
- Icecat Importer
- Anlegen verschiedener Basis-Attribute für Icecat

4.2.2 Akeneo-Client und -Cache

- Klasse zur Interaktion mit Akeneo-Api, behandelt Authentifizierung, Auflösung von Listen etc.
- Requests dauern mit unter einige Sekunden
- Cache fragt alle wichtigen Endpunkte ab und speichert JSON-payload in JSON-Dateien
- Abfrage von Daten aus dem Cache (lädt und parsed JSON-Dateien in Python-Datenstrukturen “dacite” Package)

4.3 Datenset

- Icecat Taxonomy => Import der Attribute
- händische Auswahl aller zu importierenden Attribute
- Suche nach Produkten auf icecat.biz
- Beschränkung auf Sponsoren-Vendors
- Import der Produkte mittels SKU und EAN; dann Icecat Importer “Enrich Products”
- Analyse der Daten und Korrektur verschiedener Fehler:

  - Multi-Select werden als Single-Select importiert
  - einige Single-Selects als Text
  - Import diverser “number” Attribute schlägt fehl => Import als Text und anschließend Umwandlung
  - Fix: Attribut dupliziert mit suffix “_fixed” und korrektem Typ
  - Außerdem: fehlerhafter Import diverser metrischer Attribute (wahrscheinlich Unit nicht richtig erkannt), aber ist bei allen gleich fehlerhaft importiert => wird eh normalisiert, daher ignoriert
- Statistiken zu Datenset: Attribute, Categories, Families, Produkte

4.4 Clustering

4.4.1 Überblick zu externen Libraries

- sklearn, nicovk
- erlauben keine null-Values !
- Umsetzung von Multi-Selects nicht möglich
- Orientierung an diesen Bibliotheken, aber keine Verwendung

4.4.2 Clustering-Package

- KMeans & Bisecting KMeans mit generischer Centroid-Klasse
- Centroid-Klasse erklären
- KMeans => Überblick
- Bisecting KMeans => Überblick

4.4.3 Distanzfunktionen

- Implementierung der Centroid-Klasse
- Map aus Attribut-Code zu Wert in Datentypen: float, string, set of strings
- Verarbeitung des Typs entsprechend
- Tracking des Means und der Modes je nach Attribut => Generierung eines künstlichen Datenpunktes für Distanzfunktion
- Gewichtung der Attribute möglich

4.4.4 Datenvorbereitung

- Produkte parsen:
- mit Filtern
- Normalisierung von float
- Tokenization von strings => Umwandlung in set of strings, genauso wie Multi-Selects

4.5 Evaluation

- Verwendung von metrics Funktionen aus SKLearn
- Durchführung mit Python-Packages in Jupyter-Notebooks
- Generierung aller möglichen Versuche
- Stabilität => 10 mal wiederholen da KMeans mit random-init, Vergleich mit adj_rand_index
- Qualität => Silhouettenkoeffizient, da in SKLearn am flexibelsten einsetzbar
- Korrektheit => Vergleich mit Categories (Smartphone Serien und Modellen) mit adj_rand_index

5 Auswertung

5.1 Datenset “Smartphone-Hüllen”

5.1.1 Verarbeitung multi-kategorischer Attribute

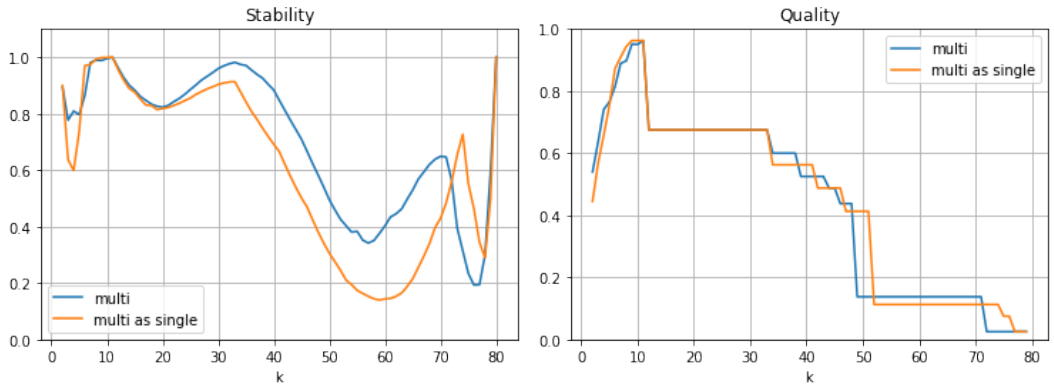

Abbildung 5.1: Stabilität und Qualität der Hüllen mit multi-kategorischem Attribut

Tabelle 5.1: Clustering der Hüllen mit multi-kategorischem Attribut

Name	Stabilität	Qualität	Erkennung: Generation	Modell	Ø
multi	70.8%	43.9%	2.1%	4.3%	30.3%
multi als single	61.0%	44.5%	0.4%	4.3%	27.5%

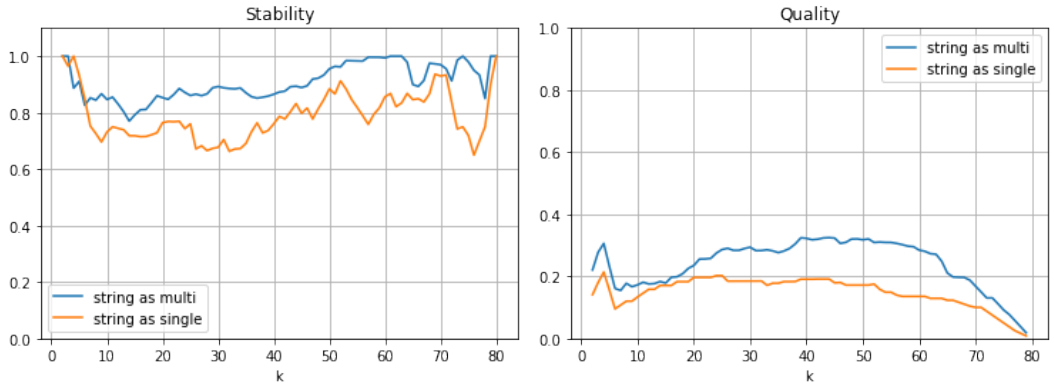

Abbildung 5.2: Stabilität und Qualität der Hüllen mit String-Attributen

Tabelle 5.2: Clustering der Hüllen mit String-Attributen

Name	Stabilität	Qualität	Erkennung: Generation	Modell	Ø
strings als multi	90.8%	24.1%	1.5%	17.5%	33.5%
strings als single	79.0%	15.0%	1.2%	7.5%	25.7%

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

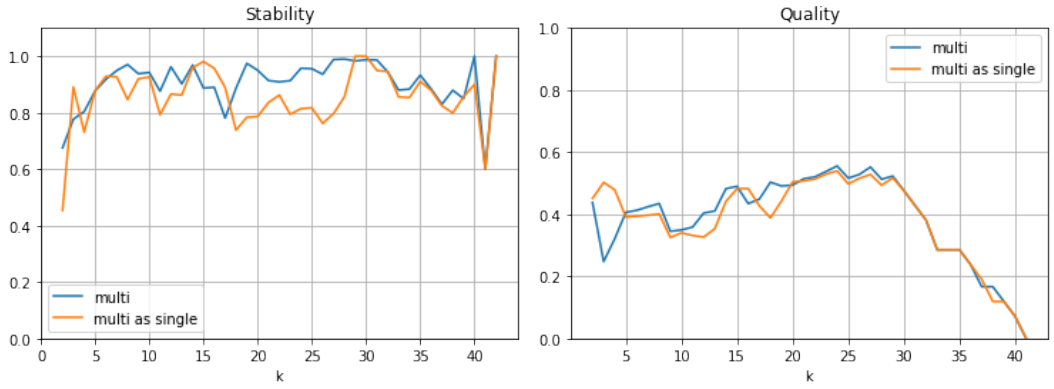

Abbildung 5.3: Stabilität und Qualität der Smartphones mit multi-kategorischen Attributen

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%

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%

Name	Stabilität	Qualität	Erkennung: Generation	Modell	Duplikate	Ø
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

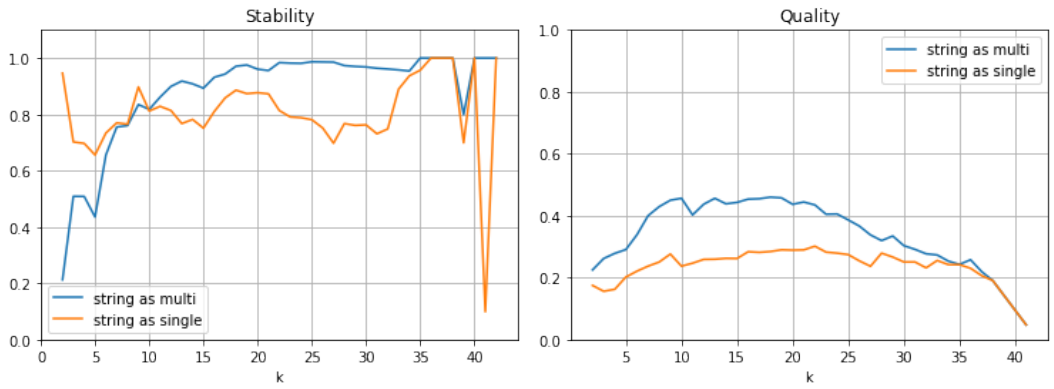

Abbildung 5.4: Stabilität und Qualität der Smartphones mit String-Attributen

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

[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

[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

[Icec22] Icecat: *About Icecat*. URL `https://icecat.com/about-us/`. - abgerufen am 2022-04-05

[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

[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

[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)

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 29.04.2022

.....
Hannes Dröse