Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Datenbanken und Informationssysteme

Verbesserung der Labelpropagierung im Datenbereinigungssystem Raha

Improving Label Propagation in the Cleaning System Raha

Bachelorarbeit

im Studiengang Informatik

von

Maximilian Siebenthaler

Prüfer: Prof. Dr. Ziawasch Abedjan
Zweitprüfer: Prof. Dr. Marius Lindauer
Betreuer: Prof. Dr. Ziawasch Abedjan

Hannover, 25.04.2022

Inhaltsverzeichnis

Kapitel 1

Einleitung

Datenbereinigung ist ein Verfahren zur Behandlung von Datenfehlern in einem Datensatz und dient zur Verbesserung der Datenqualität. Die Wichtigkeit von Datenbereinigung steigt seit Beginn der Digitalisierung mit dem wachsenden Trend großer Datenmengen [1]. Das Datenbereinigungsprogramm Raha [2] bietet für große Datenmengen eine benutzerfreundliche und effiziente Methode zur Datenbereinigung an. Damit Raha für den Benutzer noch bessere Ergebnisse beim Ermitteln vom Datenfehlern liefert, wird die Propagierung von Datenmarkierungen optimiert. In Raha werden alle Datenpunkte spaltenweise in Gruppen mit ähnlichen Eigenschaften aufgeteilt und anschließend mit den Markierungen 'richtig' oder 'falsch' versehen. Die Markierungen innerhalb einer Gruppe basieren auf zuvor getätigten Benutzermarkierungen für eine beliebige Anzahl von Zeilen, die eine Propagierung von Markierungen ermöglichen. Diese Benutzermarkierungen machen lediglich einen Bruchteil einer gesamten Gruppe aus, weshalb eine zusätzliche Gewichtung die Stärke einer Markierung ausdrückt. Die Gewichtung beschreibt von 0 bis 1, wie stark ein nicht markierter Datenpunkt zu der Markierung eines benutzermarkierten Datenpunktes tendiert. Die Gruppen einer Spalte unterscheiden sich je nach Datensatz im Aufbau und der Anzahl von Markierungen. Aus diesem Grund werden zwei Ansätze betrachtet, der eine Ansatz setzt mehr auf Quantität bei der Erweiterung von Markierungen und der andere eher auf die Qualität der Markierungen. Ist das Ziel die Quantität, wird mittels eines Ähnlichkeitsmaßes eine komplette Gruppe gewichtet. Das Ziel der Qualität beschäftigt sich hingegen genauer mit der Struktur innerhalb einer Gruppe und erweitert eine Markierung lediglich an Datenpunkte um eine naheliegende Benutzermarkierung herum. Beide Verfahren haben datensatzabhängige Vor-und Nachteile und die Auswertung erfolgt anhand der Standardkonfiguration von Raha mittels der F1-Bewertung.

Kapitel 2

Vorkenntnisse

Zum Einstieg in das Thema 'Verbesserung der Labelpropagierung im Datenbereinigungssystem Raha' bedarf es an Vorkenntnissen im Bereich Metrik, Clustering und Label Propagation. Wie werden innerhalb eines Datensatzes Gruppen gebildet und wie kann man innerhalb einer Gruppe Markierungen propagieren? Die Intention dieser Fragen ist grundlegend für die in dieser Arbeit dargestellten Ansätze.

2.1 Metrik

Eine Metrik, auch Abstandsfunktion genannt, gibt einen reellen, nicht negativen Wert für zwei Elemente aus einer Grundmenge zurück. Dieser Wert bezeichnet den Abstand der beiden Punkte zueinander [3].

2.1.1 Metrik für Vektoren

Ein Vektor ist ein n-dimensionaler Tupel bestehend aus Werten einer Grundmenge. Abstände zweier Vektoren werden häufig mithilfe der Hamming Distanz, der euklidischen Distanz, der Kosinus-Ahnlichkeit oder dem Skalarprodukt berechnet. Die Hamming Distanz bestimmt die Anzahl an unterschiedlichen Stellen zwischen zwei Vektoren. Der Wertebereich der Distanz reicht von 0, beide Vektoren sind exakt gleich, bis zur Größe der Dimension, alle Einträge sind verschieden [4]. Die euklidische Distanz bezeichnet den direkten Abstand zwischen zwei Punkten in einem Raum und ist sinnbildlich mit einem Lineal vergleichbar, dass auch in höheren Dimensionen anwendbar ist. Bei der euklidischen Distanz ist die größte Übereinstimmung bei einer Distanz von 0 gegeben [5]. Die Kosinus-Ähnlichkeit stellt ein Ähnlichkeitsmaß dar, anders als die euklidische Distanz. Anhand des Kosinus Winkels zwischen zwei Punkten wird eine Aussage über die Ähnlichkeit getroffen. Der Wertebereich des Ähnlichkeitsmaßes liegt zwischen -1 und 1. Bei einer 1 zeigen beide Vektoren in die exakt selbe Richtung und sind genau gleich. Bei

einer -1 sind sie entgegen gerichtet und bei einer 0 stehen die beiden Vektoren orthogonal zueinander. Bei Vektoren ohne negative Vorzeichen liegt der Wertebereich zwischen 0 und 1 [6]. Das Skalarprodukt benutzt auch den Kosinus Winkel zur Berechnung einer Ähnlichkeit. Es wird jedoch zusätzlich die Länge der beiden Vektoren mit dem Kosinus Winkel multipliziert. Der Wertebereich liegt zwischen 0 und 1, wenn es keine negativen Werte gibt und das Ergebnis normiert ist [7].

2.1.2 Metrik für binäre Vektoren

Binäre Vektoren sind n-dimesionale Tupel bestehend aus Elementen der Menge {0,1}. Die Metriken für Vektoren sind anwendbar, aber auch logische Metriken bei denen 0 als 'falsch' und 1 als 'wahr' betrachtet wird. Die logischen Metriken geben Ähnlichkeiten zwischen zwei binären Vektoren an und basieren auf vier möglichen Kombinationen (wahr wahr, wahr falsch, falsch wahr und falsch falsch, s. Tabelle 2.1). Die Variablen u und v sind die Einträge an der Stelle m zweier binärer Vektoren V_u, V_v, mit $0 \leq m \leq |V|$.

Tabelle 2.1: Kontingenztafel, 1=wahr und 0=falsch

Variablen u,v	v=1	v=0
u=1	a	b
u=0	c	d

Die folgenden Methoden, Matching, Jaccard, Dice und Sneath, stellen unterschiedliche Metriken für binäre Vektoren da. Die einfachste Variante die beiden binären Vektoren V_u und V_v miteinander zu vergleichen, ist Matching (engl.). Matching setzt die Anzahl an gleichen Einträgen ins Verhältnis zur Dimension der Vektoren mit $\frac{a+d}{a+b+c+d}$. Um ein Ähnlichkeitsmaß sensibler für gleiche 'wahr'-Einträge zu machen, wird Jaccard oder Dice verwendet. Jaccard ist $\frac{a}{a+b+c}$ und Dice ist $\frac{2a}{2a+b+c}$ Es ist leicht zuerkennen, dass Dice auf Jaccard basiert und lediglich die Gewichtung in Richtung a bzw. wahr-wahr-Einträge angepasst wurde. Matching dividiert durch die gesamte Anzahl von Vektoreinträgen, weshalb der Wertebereich von 0, keine Ähnlichkeit, bis 1,exakt gleich, reicht. Jaccard und Dice hingegen vernachlässigen beide d und falls $V_u und V_v$ beide Nullvektoren sind, ist das Ergebnis unbestimmt. Zusätzlich gibt es die Metrik Sneath, die eine größere Gewichtung auf a und d legt, mit $\frac{2(a+d)}{2(a+d)+b+c}$. Sneath ist eine Mischung aus Matching und Dice und hat den Wertebereich von 0 bis 1 [8].

2.2 Clustering

Clustering unterteilt Datenpunkte in Gruppen, damit innerhalb von umfangreichen Datensätzen optimale Muster und Zusammenhänge erkannt werden.

Die Einteilung in Gruppen erfolgt mittels geeigneter Methoden bzw. Metriken, basierend auf der Ähnlichkeit zwischen Datenpunkten und deren Eigenschaften [9].

2.2.1 Clusteringverfahren

Clusteringverfahren sind Algorithmen zur Bildung von Gruppen innerhalb eines Datensatzes. Zur Bestimmung der Gruppen wird eine Metrik verwendet, die die Datenpunkte anhand ihrer Eigenschaften oder Distanzmaße einer Gruppe zuweist [9]. Die zwei bekanntesten Verfahren zur Gruppenbildung sind das partitionierende und das hierarchische Verfahren. Beide Verfahren starten mit einem nicht gruppierten Datensatz und den Eigenschaften der Datenpunkte. Das partitionierende Verfahren benötigt eine festgelegte maximale Anzahl von Gruppen in die der Datensatz aufgeteilt wird. Das hierarchische Verfahren hingegen basiert auf einer Baumstruktur, in der die Anzahl von Gruppen nicht vorgeschrieben ist. Beide Verfahren kennen nur approximativ die optimale Anzahl an Gruppen, da die Anzahl von der gewählten Metrik und der Wahl der Eigenschaften abhängig ist [9]. Raha verwendet das hierarchische Clusteringverfahren, weil der Benutzer beliebig viele Datenpunkte markieren kann [2, Kapitel 4.3].

2.2.2 Hierarchisches Clustering

Wie bereits erwähnt, besitzt das hierarchisches Clustering keine feste Anzahl von Gruppen. Darüber hinaus kann hierarchisches Clustering in zwei Vorgehensweisen unterteilt werden, das agglomerative und das divisive hierarchische Clusteringverfahren. Diese werden auch als Bottom-Up und Top-Down Verfahren bezeichnet. Beide Arten des hierachischen Clustering erhalten ihre Bedeutung anhand der zugrundeliegenden Baumstruktur.

Zu Beginn des agglomerativen hierarchischen Clusteringverfahrens ist jeder Datenpunkt eine eigene Gruppe der Größe 1. Durch Anwendug der Metrik werden zwei Gruppen zusammen geführt und es existiert anschließend eine Gruppe weniger. Das Zusammenführen wird solange wiederholt, bis lediglich eine große Gruppe existiert, die den gesamten Datensatz umfasst. Aus diesem Grund wird das agglomerative Clusteringverfahren auch als Bottom-Up bezeichnet. Zur Veranschaulichung wird ein Dendogram verwendet, weil es sich um einen baumartigen Ablauf handelt. Genau gegensätzlich zum Bottom-Up Verfahren verhält sich das divisive, bzw. Top-Down, Verfahren. Das divisive Verfahren startet mit einer großen Gruppe und verfeinert diese in jedem Schritt, bis jeder Datenpunkt eine Gruppe bildet. Welches Vefahren am geeignetsten ist, hängt vom Datensatz und der Feinheit der Unterteilung ab. Wird eine grobe Gruppenaufteilung bevorzugt, dann ist das divisive Verfahren effizienter. Wird hingegen eine feine Aufteilung bevorzugt, ist das agglomerative Verfahren effizienter. Ist eine optimale Gruppenkonstellation

erreicht, kann das Clusteringverfahren ohne Probleme beendet werden und die bis dahin gebildeten Gruppen können verwendet werden.

2.2.3 Linkage Clustering

Als Linkage Clustering bezeichnet man die Anwendung der Metrik in einem Clusteringverfahren. Es beschreibt, wie Datenpunkte zu bewerten sind bzw. trägt zur Definition bei, ab wann ein Datenpunkt einer Gruppe zugehörig ist [10]. Vier Beispiele für Linkage Methoden sind Single Linkage, Average Linkage, Complete Linkage und Ward Linkage.

Der einfachste Ansatz ist Single Linkage, weil zwei Datenpunkte direkt miteinander verglichen werden. Average Linkage hingegen verwendet den durchschnittlichen Abstand aller Datenpunkte aus jeder Gruppe. Complete Linkage berechnet die maximale Distanz zu Datenpunkten in anderen Gruppen und wählt anschließend zum Beispiel die Gruppe mit der minimalsten Maximaldistanz. Ward Linkage ist ein Varianzminimierungsverfahren und formt möglichst homogene und gleich große Gruppen [10].

2.3 Label Propagation

Label Propagation ist ein graphbasierter teilüberwachter Lernalgorithmus, der Markierungen innerhalb eines Datensatzes verbreitet [11]. Der Unterschied zwischen einem überwachten und einem teilüberwachten Lernalgorithmus ist die Möglichkeit dem Algorithmus direkt mitzuteilen, ob ein Ergebnis korrekt oder falsch ist, weil in einem überwachten Umfeld alle Markierung schon bekannt sind. Bei einem teilüberwachten Lernalgorithmus ist die Anzahl der bekannten Markierungen sehr gering im Verhältnis zu den unbekannten und der Algorithmus kann nur schätzen, wie korrekt ein Ergebnis ist, weil es keine Kontrolle gibt.

Label Propagation versucht anhand von Dateneigenschaften im Datensatz die optimalsten Markierungen für die nicht markierten Datenpunkte zu finden. Damit dies gelingt, bedarf es drei Annahmen: der Smoothness , Cluster und Manifold Annahme [12].

2.3.1 Smoothness Annahme

Die Smoothness Annahme besagt, dass zwei Punkte die nah beieinander liegen sehr wahrscheinlich dieselbe Markierung besitzen. Angenommen es existiert eine reelle stetige Funktion $f(x)$ mit $f : \mathbb{R} \to \mathbb{R}$ und gesucht sei $f(m)$, mit $m > 0$, dann gibt es zwei Werte $m_-, m_+ \neq m$ mit $m_- < m < m_+$. Der Wertebereich von $f(m)$ ist dann durch $\frac{f(m_-)+f(m_+)}{2} \pm \epsilon$ definiert, Epsilon ϵ ist abhängig von dem Glättegrad von $f(x)$. Werden m_- und m_+ minimal an m gewählt und die Funktion $f(x)$ ist sehr glatt, wird laut der Smoothness Annahme ϵ gegen 0 gehen. In Bezug auf einen Datensatz existieren zwei

Datenpuntke d_1, d_2 mit den Markierungen l_1, l_2, wobei $l_1 = l_2$ ist. Gesucht wird eine Markierung für den Datenpunkt d_3, der nah an d_1 und d_2 liegt. Die Markierung von d_3 wird laut Smoothness Annahme gleich l_1 und l_2 sein. Sind die Markierungen von d_1 und d_2 ungleich, $l_1 \neq l_2$, so kann mithilfe einer Metrik eine Tendenz für die Markierung von d_3 herangezogen werden [12][13].

2.3.2 Cluster Annahme

Die Cluster Annahme besagt, dass Datenpunkte innerhalb einer Gruppe wahrscheinlich dieselbe Markierung besitzen. Wird ein Datensatz mithilfe eines Clusteringverfahrens in Gruppen aufgeteilt und innerhalb einer Gruppe existiert nur eine Art von Markierung, dann besitzen alle Datenpunkte der Gruppe wahrscheinlich dieselbe Markierung. Diese Aussage verliert allerdings an Gewichtung, wenn es große Gruppen mit vielen Eigenschaften gibt [12][14].

2.3.3 Manifold Annahme

Die Manifold Annahme besagt, dass hochdimensionale Datenpunkte meistens auf einer niedrigdimensionalen Mannigfaltigkeit liegen. Datenpunkte, die einen hochdimensionalen Vektor haben, können innerhalb einer Gruppe auf weniger Dimensionen verglichen werden. Ein Beispiel wäre ein Datensatz, der 100 Bilder mit handschriftliche Zahlen beinhaltet, wobei jedes Bild $28x28$ Pixel hat. Damit jede '2' identifiziert wird, könnten von jedem Bild die 784 Pixel verglichen werden. Allerdings ähneln sich die Bilder meistens bis auf wenige Pixel und die Bilder mit einer '2' lassen sich mithilfe von nur 2 Pixeln bestimmen. Der Datensatz liegt dann auf einer niedrigdimensionalen Mannigfaltigkeit zur Bestimmung der '2' [15, Beispiel 1.2] [12].

2.3.4 Label Propagation Ablauf

Label Propagation ist wie bereits erwähnt ein graphbasierter Algorithmus und die drei Annahmen beschreiben jeweils globale und lokale Annahmen zu Datenpunkten und deren Umgebung. Der Algorithmus betrachtet den Datensatz als Graph und die Datenpunkte werden als Knoten dargestellt. Datenpunkte die innerhalb eines vorbestimmten Umfeldes um einen Datenpunkt liegen, werden als Nachbarpunkte betrachtet. Kanten verbinden einen Datenpunkt mit seinen Nachbarpunkten und werden zusätzlich mithilfe eines Ähnlichkeitsmaßes bzw. einer Metrik gewichtet. Es existiert nun ein Graph aus Knoten und Kante, der Markierungen von einzelnen Datenpunkten und deren Beziehung zu anderen Datenpunkten beinhaltet. Damit ein nicht markierter Datenpunkte eine Markierung erhält, wird anhand der Kantengewichtung eine Verteilung für jeden nicht markierten Datenpunkt bestimmt. Jeder Datenpunkte erhält anschließend die Markierung mit der

größten Wahrscheinlichkeit basierend auf der punktspezifischen Verteilung. Zusätzlich kann der Algorithmus noch optimiert oder verfeinert werden, indem die Größe des Umfeldes und die Tiefe von Nachbar-Nachbars Knoten variiert wird [11].

Kapitel 3

Raha

Raha agiert im Bereich der Datenbereinigung. Die Datenbereinigung prüft einen Datensatz auf Fehler und Inkonsistenzen, damit falsche Diagnosen und Auswertungen vermieden werden. Die Datenbereinigung soll grundsätzlich die Datenqualität und die Zuverlässigkeit des Datensatzes erhöhen. Blieben fehlerhaften Einträge weiterhin im Datensatz bestehen, so kann eine Auswertung des Datensatzes zu falschen Statistiken führen. Datenbereinigung hebt die Datenqualität und verbessert damit indirekt die Qualität der Statisiken [2, Introduction].

Datenbereinigung lässt sich nach Luber und Litzel grob in vier Schritte unterteilen. Die Definition der Anforderungen an die Datenqualität, die Analyse der vorliegenden Daten gemäß der definierten Anforderungen, gefolgt von der Auflistung der Probleme der Datenqualität während der Analyse, und abschließend die Bereinigung von problematischen Dateneinträgen [16].

3.1 Raha Ansatz

Zu Beginn von Raha wird der richtige Algorithmus zur Fehlererkennung ausgewählt und anschließend korrekt konfiguriert. Dies erfordert vom Benutzer Vorwissen über den Datensatz, weil die Datenbereinigung ansonsten wenig erfolgsversprechend ist. Denn je mehr der Benutzer über den Datensatz weiß, desto wahrscheinlicher ist es, einen Großteil der Fehler zu erkennen. Was, wenn dem Benutzer das nötige Vorwissen über den Datensatz und die Konfiguration des Algorithmus fehlen? Raha ist sich dieses Problems bewusst und bietet dem Benutzer ein Programm, dass ohne viel Vorwissen eine gute Datenqualität liefert. Das Ziel von Raha besteht darin, eine bestmögliche Datenqualität mit minimalen Benutzereingaben zu erhalten. Zusätzlich kann bei Notwendigkeit und ausreichendem Vorwissen Raha manuell konfiguriert werden.

3.2 Raha Ablauf

Wie in 3 schon erwähnt, basiert Datenbereinigung auf vier essentiellen Schritten, Raha ist hier keine Ausnahme. Eine Zeile eines Datensatzes besteht aus Datenpunkten. Dem Benutzer werden beliebig viele Zeilen gezeigt und jeder Datenpunkt dieser Zeilen wird mit 'richtig' oder 'falsch' markiert. Anhand der markierten Datenpunkte und deren Eigenschaften werden Gruppen gebildet und anschließend ausgewertet. Am Ende besitzt jeder Datenpunkt des Datensatzes eine Markierung und der Benutzer kann die fehlerhaften Datenpunkte korrigieren oder entfernen.

3.2.1 Fehlererkennung - Definition der Anforderungen an die Datenqualität

Es wird bei der Datenbereinigung zwischen vier Arten von Fehlern unterschieden. Für jede Art von Fehler existieren bereits Algorithmen und eine Erkennung ist innerhalb eines Datensatzes möglich. Diese Fehlererkennungen lauten Ausreißererkennung, Mustererkennung, Regelverletzung und wissensbasierte Verletzung. Raha bietet dem Benutzer eine Vielzahl von Fehlererkennungsalgorithmen basierend auf den vier Arten von Fehlern an, die bereits eine Standardkonfiguration haben [2, Kapitel 4.1]. Diese Standardkonfigurationen sind allerdings nicht für jeden Datensatz optimal, weshalb einige Ergebnisse unpräzise sind. Aus diesem Grund kann ein Ergebnis nicht als absoluter Wert behandelt werden, sondern dient als Ähnlichkeitsmaß zwischen Datenpunkten (s. Kapitel 3.2.2).

3.2.2 Eigenschaftenvektor - Analyse der vorliegenden Daten gemäß der definierten Anforderungen

Datenpunkte einer Zeile sind häufig in verschiedenen Formatierungen gespeichert, zum Beispiel als Namen, Zahlen oder Daten. Wird ein Fehlererkennungsalgorithmus auf jeden Datenpunkt einer Zeile angewendet, können die Datenpunkte der Zeile nicht verglichen werden, weil die Formate nicht gleich sind. Ähnlichkeiten sind nur möglich mit Datenpunkten des gleichen Formates. Dies ist innerhalb jeder Spalte eines Datensatzes gegeben. Innerhalb von Zeilen sind teilweise minimale Korrelationen zwischen Datenpunkten vorhanden, zum Beispiel kann eine Adresse nicht in der Stadt liegen. Solche Fehler werden mithilfe der regel- und wissensbasierten Verletzung detektiert. Raha betrachtet den Datensatz zwar spaltenweise, aber die Korrelationen innerhalb von Zeilen werden nicht vernachlässigt [2, Kapitel 4.2].

Raha wendet auf jeden Datenpunkt alle in Raha hinterlegten Fehlererkennungsalgorithmen an. Ein Ergebnis eines Fehlererkennungsalgorithmus wird auf 0 oder 1 projiziert, dadurch erhält jeder Datenpunkt einen Vektor bestehend aus 0 und 1 zugewiesen. Dieser Vektor wird als Eigenschaf-

tenvektor bezeichnet und für Datenpunkte innerhalb derselben Spalte als Ähnlichkeitsmaß verwendet. Da Ergebnisse nur 0 oder 1 sein können, wird der Vektor auch als binärer Vektor bezeichnet. Anschließend trimmt Raha die Länge von binären Vektoren derselben Spalte, indem überflüssige oder konstante Eigenschaften entfernt werden. Mit anderen Worten, besitzen alle Eigenschaftenvektoren einer Spalte an der Stelle m eine 1 bzw. 0, kann bei allen Vektoren die Stelle m entfernt werden, da jeder Datenpunkt diese Eigenschaft teilt.

3.2.3 Gruppierung - Auflistung der Probleme der Datenqualität während der Analyse

Raha kennt bisher die Eigenschaften von allen Datenpunkten und zusätzlich die Ähnlichkeitsmaße innerhalb von Spalten. Ziel der Gruppierung ist die Markierung von Datenpunkten mit 'richtig' oder 'falsch' für das Training eines Klassifizierers (s. Kapitel 3.2.4), einige Datenpunkte verbleiben ohne Markierung. Der Benutzer wird aufgefordert, eine beliebige Anzahl von Zeilen, die Raha optimal bestimmt, auszuwerten und jeden Datenpunkt der Zeile mit 'richtig' (0) oder 'falsch' (1) zu markieren. Die benutzermarkierten Datenpunkte sind ein Bruchteil des gesamten Datensatzes, somit gibt es noch eine große Menge nicht markierter Datenpunkte. Damit mehr Datenpunkte eine Markierung erhalten, verwendet Raha hierarchisches Clustering und Label Propagation (s. Kapitel 2.2, 2.3) [2, Kapitel 4.3]. Die Datenpunkte einer Spalte werden anhand der Eigenschaftenvektoren und der Anzahl markierter Zeilen in Gruppen aufgeteilt. Diese Gruppen sind spaltenspezifisch, also die Gruppierung einer Spalte gleicht nicht der Gruppierung einer anderen Spalten. Anschließend existieren drei Arten von Gruppen, die Gruppen ohne benutzermarkierte Datenpunkte, die Gruppen mit einheitlich benutzermarkierten Datenpunkten und die Gruppen mit verschiedenen Benutzermarkierungen. Die letzten beiden Gruppen sind für Raha von Interesse, weil Markierungen innerhalb einer Gruppe an Datenpunkte mit ähnlichen Eigenschaftenvektoren verteilt bzw. propagiert werden können. Durch diesen letzten Schritt erhalten viele Datenpunkte eine Markierung, ohne dass der Benutzer diese markieren muss [2, Kapitel 4.4].

3.2.4 Klassifikation - Bereinigung von problematischen Einträgen

Ein binärer Klassifikator bekommt als Eingabeparameter einen n-dimensionalen Vektor und weist diesen Vektor einer von zwei Gruppen zu [17]. Um eine solche Zuweisung zu machen, benötigt der Klassifikator Trainingsdaten, um innerhalb der Trainingsdaten Muster erkennen zu können. Da jede Spalte Eigenschaftenvektoren mit einer spaltenspezifischen Länge besitzt, werden alle Datenpunkte bzw. ihre Eigenschaftenvektoren, die markiert sind, dem

Klassifikator als Trainingsdaten übergeben. Anschließend werden dem Klassifikator die gesamten Datenpunkte einer Spalte übergeben und anhand der gelernten Muster wird jedem Datenpunkt die Klasse 'korrekt' oder 'fehlerhaft' zugewiesen. Alle Datenpunkte, die vom Klassifikator als fehlerhaft bewertet werden, kann der Benutzer einsehen und korrigieren oder löschen [2, Kapitel 4.4].

3.3 Homogene und heterogene Gruppen

Im dritten Schritt der Datenbereinigung gruppiert Raha die Datenpunkte und es entstehen Gruppen mit einheitlichen Markierungen, die homogene Gruppen, und Gruppen mit unterschiedlichen Markierungen, die heterogene Gruppen. Diese beiden Gruppen können für die Label Propagation verwendet werden, weil sie markierte Datenpunkte enthalten. Das Verhältnis der beiden Gruppen zueinander ist von der Anzahl des Parameters $LABELING\_BUDGET$ abhängig (s. Abbildung 3.1, Datensätze s. [2, Kapitel 6.1] oder Kapitel 4.2.3). Das $LABELING\_BUDGET$ gibt an, wie viele Zeilen der Benutzer maximal markieren kann, standardmäßig ist 20 angegeben. Je größer das $LABELING\_BUDGET$ gewählt wird desto mehr und kleinere Gruppen werden gebildet und folglich sinkt auch die Wahrscheinlichkeit für heterogene Gruppen. Die beiden verschiedenen Gruppen entstehen, weil es keinen Algorithmus gibt, der jeden Datensatz perfekt gruppieren kann. Deshalb ist zu beobachten, dass Datenpunkte mit ähnlichen Eigenschaften aber unterschiedlichen Markierungen in einer Gruppe landen.

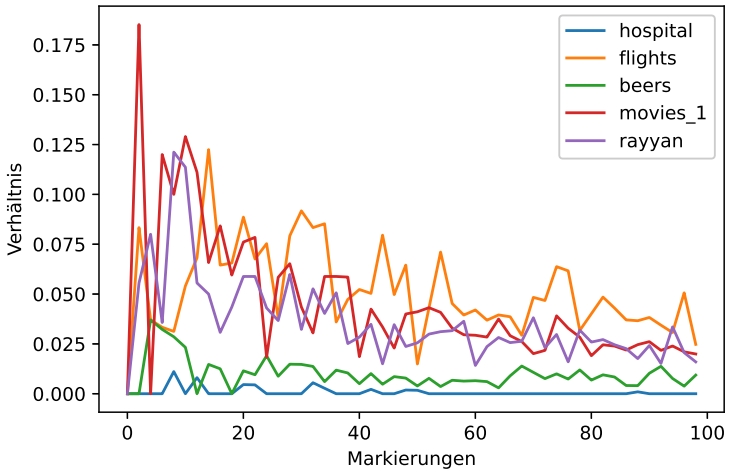

Abbildung 3.1: Heterogene Gruppen zu homogenen Gruppen Verhältnis

3.4 Bisheriger Ansatz - Label Propagation in Raha

Raha besitzt zwei Konfiguration, die beide auf der Cluster Annahme basieren, um homogene und heterogene Gruppen auszuwerten. Ziel der Label Propagation in Raha ist es möglichst viele Trainingsdaten für den Klassifikator zu sammeln. Die Menge der Trainingsdaten sollte groß sein, jedoch gilt es eine Balance aus Quantität und Qualität zu haben, denn lediglich ein Bruchteil des gesamten Datensatzes, die Benutzermarkierungen, sind sehr wahrscheinlich korrekt markiert. Rahas erste Konfiguration bezieht sich deshalb ausschließlich auf die homogenen Gruppen und markiert innerhalb einer homogenen Gruppe alle Datenpunkte mit der Markierung eines benutzermarkierten Datenpunktes. Die Anzahl der Trainingsdaten steigt damit um ein Vielfaches und die heterogenen und nicht-markierten Gruppen bleiben unmarkiert. Die zweite Konfiguration 'Majority' verhält sich gleich zum ersten Ansatz in homogenen Gruppen. Jede Gruppe erhält eine mehrheitliche Markierung, weil in homogenen Gruppen nur eine Markierung existiert. In heterogenen Gruppen hingegen existieren unterschiedliche Markierungen. Eine Markierung mithilfe der Mehrheit wird einige Datenpunkte falsch markieren, weshalb die erste Konfiguration effizienter markiert und als Standardkonfiguration vorgeschlagen wird [2, Kapitel 4.4].

3.5 Gewichtete Label Propagation in Raha

Raha gruppiert die Datenpunkte und erweitert die Benutzermarkierungen innerhalb von Gruppen mit Markierung. Die Markierungen, die erweitert wurden, basieren lediglich auf zwei von drei theoretischen Ansätzen von Clustering und Label Propagation und es gilt die Smoothness Annahme zuerfüllen. Deshalb wird nicht benutzermarkierten Datenpunkten in homogenen und heterogenen Gruppen eine Gewichtung zugeteilt, die eine Tendenz für eine Markierung darstellt. Sind zwei Datenpunkte sich sehr ähnlich, tendiert die Gewichtung gegen 1, und bei wenig Übereinstimmung tendiert die Gewichtung gegen 0. Ziel der Gewichtung ist die Optimierung der drei Clustering Annahmen. Hierzu werden die homogenen und heterogenen Gruppen differenziert voneinander betrachtet und mittels unterschiedlicher Ansätze und Metriken ausgewertet. Mithilfe der Gewichtung wird die Anzahl an fehlerhaften Markierungen gesenkt, damit die Trainingsdaten für den Klassifikator eine höhere Qualität bekommen und ein Datensatz besser bereinigt wird.

Kapitel 4

Metrischer und dichtebasierter Ansatz

Diese Arbeit stellt zwei Lösungsansätze zur gewichteten Label Propagation in homogenen und heterogenen Gruppen vor. Der erste Ansatz, der metrische Ansatz, orientiert sich an den theoretischen Clustering Annahmen und nutzt eine Metrik. Alle Datenpunkte einer Gruppe werden als dieser Gruppe zugehörig betrachtet, die Gruppe ist somit vollständig bzw. komplett. Anschließend wird den Datenpunkten basierend auf einer Metrik eine Gewichtung zugewiesen.

Der zweite Ansatz, der dichtebasierte Ansatz, orientiert sich auch an den Cluster Annahmen, bezieht sich allerdings zusätzlich genauer auf den Aufbau des Datensatzes. Die Datenpunkte einer Gruppe bekommen anhand der Anzahl von naheliegenden Datenpunkten eine dichtes Gebiet (s. Kapitel 4.3) zugewiesen. Alle Datenpunkte, die nicht in einer nahen Umgebung um einen markierten Datenpunkte herum liegen, erhalten als Gewichtung automatisch 0. Innerhalb von naheliegenden Datenpunkten bzw. dichten Gebieten kann zusätzlich eine Metrik zur Gewichtung verwendet werden. Die Ansätze und Ergebnisse sind online verfügbar [18].

4.1 F1 Bewertung

Zu Beginn erhält der nicht markierte Datensatz vom Benutzer Markierungen. Anschließend werden weitere Datenpunkte, basierend auf den Eigenschaften und den Benutzermarkierungen markiert. Alle markierten Datenpunkte werden dem Klassifikator als Trainigsdaten übergeben. Der Klassifikator lernt Muster innerhalb der Datenpunkte zu erkennen und versieht, auf Grundlage dieser Muster alle Datenpunkte einer Spalte mit Markierungen. Überprüft man anschließend die Datenpunkte einer Spalte auf die Richtigkeit der Markierungen, gibt es vier mögliche Fälle (s. Tabelle 4.1) [19].

Tabelle 4.1: Wahrheitsmatrix

	Markierung 'falsch'	Markierung 'richtig'
Korrekt	korrekt falsch (KF)	korrekt richtig (KR)
Falsch	falsch falsch (FF)	falsch richtig (FR)

Der Klassifikator versieht Datenpunkte mit den Markierungen 'richtig' oder 'falsch', anhand der zuvor erlernten Muster. Diese errechneten Markierungen sind entweder korrekt oder nicht. Zur Bewertung der Markierungen werden drei Werte verwendet, die Genauigkeit (engl. precision), die Trefferquote (engl. recall) und die F1-Bewertung [4]. Die in Raha relevante Markierung ist 'falsch', weil Raha zur Fehlererkennung dient.

$$Genauigkeit = \frac{KF}{KF + FF} \quad Trefferquote = \frac{KF}{KF + KR}$$

Die F1-Bewertung beschreibt das Verhältnis zwischen Genauigkeit und Trefferquote. Die Evaluierung dieser Arbeit bezieht sich hauptsächlich auf die F1-Bewertung und zusätzlich auf die Standardabweichung.

$$F1 = 2 * \frac{Genauigkeit * Trefferquote}{Genauigkeit + Trefferquote}$$

4.2 Metrischer Ansatz

Nachdem die Datenpunkte gruppiert wurden, befinden sich Datenpunkte mit ähnlichen Eigenschaften in gleichen Gruppen. Raha hat die Cluster Annahme in homogenen Gruppen und die Manifold Annahme bereits erfüllt, weil irrelevante Eigenschaften entfernt wurden. Die Smoothness Annahme findet Anwendung mithilfe einer Gewichtung, die Ähnlichkeiten bzw. Tendenzen zwischen Datenpunkten veranschaulicht. Existieren in homogenen und heterogenen Gruppen sehr wenig benutzermarkierte Datenpunkte und ist zusätzlich die Anzahl an Eigenschaften groß, gibt es viele Datenpunkte, die sich möglicherweise am Rand einer Gruppe befinden. Ein einzelner benutzermarkierter Datenpunkt, der sich am Rand einer homogenen Gruppe befindet, kann die gesamte Gruppe laut der Cluster Annahme homogen erscheinen lassen. Liegt dem hingegen der benutzermakierte Datenpunkt im Zentrum der homogenen Gruppe, erhalten Datenpunkte am Rand dessen Markierung und es kann widerum zu falschen Markierungen kommen. Die Gruppierung ist jedoch nicht perfekt und Datenpunkte in einer Gruppe müssen nicht zwangsweise dieser zugehörig sein. Um diesen Konflikt zu lösen wird eine Metrik eingeführt, die ein Ähnlichkeitsmaß zwischen benutzermarkierten Datenpunkten und den restlichen Datenpunkten angibt. Das Ähnlichkeitsmaß für jeden markierten Datenpunkt wird zusätzlich dem binären Klassifikator übergeben. Hierzu wird, bei der Übergabe der Trainigsdaten, der Parameter *sample_weight* verwendet [17]. Wird *sample_weight*

als Parameter nicht angegeben, besitzen alle Trainigsdaten dieselbe Gewichtung und eine Unterscheidung mittels der Smoothness Annahme ist nicht möglich. Zur Gewichtung wird eine Single Linkage Metrik verwendet, weil die Anzahl an benutzermarkierten Datenpunkten sehr gering ist und zusätzlich die Positionen der benutzermarkierten Datenpunkte zufällig bestimmt wird, je nach ausgewerteter Zeile. Befindet sich zum Beispiel mehr als ein benutzermarkierter Datenpunkt in einer homogenen Gruppe, kann ein Mittelpunkt der Markierungen bestimmt werden und zusätzlich noch in die Auswertung mit einfließen, allerdings ist keine feste Position der benutzermarkierten Datenpunkte innerhalb einer Gruppe gegeben und die Mittelpunkte haben wenig Aussagekraft über die gesamte Gruppe. Die Single Linkage Metrik zur Gewichtung bezieht sich immer auf den nächstliegenden oder ähnlichsten benutzermarkierten Datenpunkt. Die Verteilung der benutzermarkierten Datenpunkte innerhalb der Gruppe wird außer Acht gelassen.

Heterogene Gruppen erfüllen die Cluster Annahme nicht, können jedoch mittels der Single Linkage Metrik auch gewichtet werden. Es gilt hauptsächlich fehlerhafte Markierungen zu vermeiden und eine Tendenz zur nächsten bzw. ähnlichsten Markierung zu haben. Die Metrik in heterogenen Gruppen ist gleich der Metrik in homogenen Gruppen, weil nur der nächstliegende bzw. ähnlichste benutzermarkierte Datenpunkt die Tendenz ausdrückt. Mithilfe einer Metrik ist das Problem der Cluster Annahme aber nicht vollständig gelöst. Der Ansatz für heterogene Gruppen vermeidet lediglich mehr falsche Markierungen, allerdings stellen dichte Gebiete ein Problem dar (s. Kapitel 4.3).

Damit eine Metrik mithilfe des *sample_weight* Parameters funktioniert, muss der Wertebereich der Metrik im Intervall von [0,1] liegen, 0 irrelevanter Datenpunkt und 1 relevanter Datenpunkt. Je ähnlicher sich ein benutzermarkierter Datenpunkt und ein nicht markierter Datenpunkt sind, desto mehr tendiert der Wert der Metrik gegen 1. Bei numerischen Distanzen, ist die geringste Distanz 0, wenn sich zwei Datenpunkte komplett ähneln, bis zu Werten größer 1, wenn es wenig Ähnlichkeit gibt. Deshalb müssen Distanzen auf das Intervall [0,1] projiziert werden. Bei der euklidischen Distanz werden die Vektoren normiert und es wird $1 - Distanz_{normiert}$ gerechnet, wodurch sich ein Wertebereich von 0 bis 1 ergibt. So ähnlich muss auch die Hamming Distanz angepasst werden. Bei den Ähnlichkeiten und den binären Metriken, entspricht der Wertebereich bereits den Anforderungen im Intervall zu liegen.

4.2.1 Aufbau

Raha kennt die Eigenschaften der Datenpunkte, die benutzermarkierten Datenpunkte und die Gruppen jeder Spalte. Um homogene Gruppen zu gewichten, werden innerhalb einer Gruppe die benutzermarkierten und nicht markierten Datenpunkte getrennt. Anschließend vergleicht die Single Linka-

ge Metrik jeden der nicht markierten Datenpunkte mit jedem benutzermarkierten. Es wird immer der benutzermarkierte Datenpunkt mit der höchsten Ähnlichkeit zu einem nicht markierten Datenpunkt gewählt und sowohl die Markierung, als auch das Ähnlichkeitsmaß werden gespeichert. Sei D der gesamte Datensatz und die benutzermarkierten Datenpunkte sollen getrennt werden von den nicht markierten Datenpunkten innerhalb einer Gruppe. Aus D wurde bereits für jede Spalte die Datenstruktur *gruppen* erstellt, die die Gruppen und die Datenpunkte jeder Gruppe enthält. Die neue Datenstruktur *datenpunkte* enthält je Spalte die Gruppen und innerhalb jeder Gruppe die benutzermarkierten und nicht markierten Datenpunkte getrennt voneinander, Index 0 = *benutzermarkiert* und Index 1 = *nichtmarkiert* (s. Algorithmus 1). In der Struktur *datenpunkte* wird jeder Punkt bzw. Datenpunkt in einem Wörterbuch (engl. dictionary) mit einem Tupel bestehend aus der Gewichtung und Markierung hinterlegt. Benutzermarkierte Datenpunkte erhalten als Gewichtung eine 1. Nicht markierte Punkte erhalten vorerst eine 0 als Gewichtung und -1 (ungültig) als Markierung.

Algorithmus 1 Trennung von Gruppen

$gruppen \leftarrow \{Spalte : \{Gruppe : Gruppendaten \subset D_j\}\}$
$datenpunkte \leftarrow \{Spalte : \{Gruppe : [\{\}, \{\}]\}\}$ ▷ [Index 0,Index 1]
for jede Spalte j **do**
  for jede Gruppe g **do**
    for jeden Punkt p in $gruppen[j][g]$ **do**
      if p benutzermarkiert **then**
        $datenpunkte[j][g][0][p] \leftarrow (1.0, get\_markierung(p))$
      else
        $datenpunkte[j][g][1][p] \leftarrow (0.0, -1)$
      end if
    end for
  end for
end for

Um die aufgeteilten Datenpunkte innerhalb jeder Gruppe zu gewichten, wird anschließend die Metrik auf die nicht markierten Datenpunkte angewendet. Die Gewichtung und Markierung 'ungültig' wird ergänzt. Sei G eine Gruppe aus einer Spalte von *datenpunkte* und *metrik* eine Single Linkage Metrik (s. Algorithmus 2).

Algorithmus 2 Metrischer Ansatz - homogene Gruppen

```
G ← [{markiert}, {nichtmarkiert}]
if g ist homogen then
    markierung ← get_gruppenmarkierung(G)
    for jeden Punkt p1 in nichtmarkiert do
        max ← 0
        for jeden Punkt p2 in markiert do
            w ← metrik(p1, p2)
            if w > max then max ← w
            end if
        end for
        G[p1] ← (max, markierung)
    end for
end if
```

In heterogenen Gruppen wird zusätzlich zur Gewichtung keine einheitliche Markierung, sondern die Markierung des naheliegendsten bzw. ähnlichsten benutzermarkierten Datenpunktes verwendet. Sei G wieder eine Gruppe aus einer Spalte von *datenpunkte* und *metrik* eine Single Linkage Metrik (s. Algorithmus 3).

Algorithmus 3 Metrischer Ansatz - heterogen Gruppen

```
G ← [{markiert}, {nichtmarkiert}]
if g ist heterogen then
    for jeden Punkt p1 in nichtmarkiert do
        max ← (0, 0)                          ▷ (Gewicht,Markierung)
        for jeden Punkt p2 in markiert do
            w ← metrik(p1, p2)
            if w > max[0] then max ← (w, get_markierung(p2))
            end if
        end for
        G[p1] ← max
    end for
end if
```

4.2.2 Übergabe an den Klassifikator

Damit die Trainigsdaten korrekt generiert werden, müssen die Datenpunkte mit den Standardeinträgen bei Gewichtung und Markierung heraus gefiltert werden. Hierzu wird eine einfaches Wörterbuch *alle_markierungen* verwendet, dem alle markierten Datenpunkte übergeben werden. Anschließend werden alle markierten Datenpunkte auf X, Y und Gewichtung aufgeteilt. X ist der Eigenschaftenvektor, der zur Eingabe für den Klassifikator dient. Y

ist das Ergebnis zu dem entsprechenden Eigenschaftenvektor X_Y. Die Gewichtung W ist der *sample_weight* Parameter (s. Algorithmus 4).

Algorithmus 4 Klassifikator

```
result ← {}
alle_markierungen ← []
for jede Spalte j do
    for jede Gruppe g do
        for jeden Punkt p in datenpunkte[j][g][0] do
            alle_markierungen[p] ← datenpunkte[j][g][0][p]
        end for
        for jeden Punkt p in datenpunkte[j][g][1] do
            if get_markierung(p) ≠ −1 then
                alle_markierungen[p] ← datenpunkte[j][g][1][p]
            end if
        end for
    end for
end for
for jede Spalte j do
    X ← [get_vektor(p) für jeden Punkt p in Spalte j]
    Y ← [get_markierung(p) für jeden Punkt p in Spalte j]
    W ← [get_gewichtung(p) für jeden Punkt p in Spalte j]
    trainiere_Klassifikator(X, Y, sample_weight = W)
    alle_spalten_punkte ← [jeder Punkt p in Spalte j]
    result.update(auswertung_Klassifikator(alle_spalten_punkte))
end for
```

Besonders in der letzten Schleife ist gut zu erkennen, dass Raha spaltenweise den Datensatz auswertet und auch einen spaltenspezifischen Klassifikator für jede Spalte neu trainiert. Das endgültige Ergebnis *result* beinhaltet die Markierungen für alle Datenpunkt einer Spalte basierend auf dem Muster des Klassifikators. Der Benutzer kann anschließend die Datenpunte, die als 'falsch' markiert wurden selbst verwalten und mithilfe der F1-Bewertung wird das Ergebnis evaluiert.

4.2.3 Ergebnisse

Zum Testen werden zwei Versuche untersucht. Der direkte Vergleich mit 100 Wiederholungen basierend auf der gleichen Gruppierung pro Durchlauf und ein Vergleich mit unterschiedlicher Anzahl an Benutzermarkierungen, von 0 bis zu 100 Markierungen. Die Versuche werden auf den Datensätzen Hospital, Flights, Beers, Rayyan und Movies [2, Kapitel 6.1] getestet. Es handelt sich dabei um reale Datensätze, die schon bereinigt wurden und deshalb eine gute Auswertung ermöglichen, weil zu jedem Datensatz eine bereinigte und unbereinigte Version vorliegt. Zusätzlich wird der Datensatz Tax benutzt um die Skalierbarkeit zu demonstrieren.

Tabelle 4.2: Charakteristiken der Datensätze

Name	Size	Error Rate
Hospital	1000 x 20	0.03
Flights	2376 x 7	0.30
Beers	2410 x 11	0.16
Rayyan	1000 x 11	0.09
Movies	7390 x 17	0.06
(Tax	200000 x 15	0.04)

Die verschiedenen Konfigurationen werden im ersten Ansatz mit der Standardkonfiguration von Raha verglichen und ausgewertet, die als 'Normal' angegeben ist. Der Ablauf eines Versuches mit dem direkten Vergleich, erstellt erst die Eigenschaftenvektoren, dann die Gruppen und anschließend werden die Markierungen erweitert. Als Anzahl an Markierungen wird der Raha Standardwert von 20 gewählt. Die Propagierung der Markierungen wird mit jeder Konfiguration ausgeführt. Sie sind vergleichbar, weil die Gruppen und die Wahl der benutzermarkierten Datenpunkte gleich sind. Die Aussagekraft der Ergebnisse ist dadurch genauer und bei 100 Durchläufen mit gleicher Gruppierung weniger stark vom Zufall beeinflusst. Der erste Teil des ersten Versuches findet nur in homogenen Gruppen statt und die heterogenen Gruppen werden ähnlich wie bei der Standardkonfiguration von Raha ignoriert. Die Standardkonfiguration von Raha ist als 'Normal' gekennzeichnet.

Tabelle 4.3: Metrischer Ansatz - Homogene Gruppen,
σ = Standardabweichung

Metrik	Beers		Flights		Hospital	
	$F1$	σ	$F1$	σ	$F1$	σ
Normal	0.992	0.015	0.827	0.027	0.716	0.082
Euclidean	0.992	0.016	0.829	0.026	0.712	0.082
Hamming	0.992	0.015	0.826	0.028	0.716	0.082
Matching	0.992	0.016	0.829	0.026	0.711	0.086
Dice	0.992	0.015	0.828	0.026	0.711	0.084
Jaccard	0.992	0.016	0.829	0.027	0.714	0.085
Skalar	0.992	0.016	0.828	0.026	0.714	0.085
Cosinus	0.992	0.015	0.829	0.026	0.713	0.083

Metrik	Movies		Rayyan	
	$F1$	σ	$F1$	σ
Normal	0.868	0.059	0.77	0.065
Euclidean	0.869	0.061	0.771	0.065
Hamming	0.866	0.061	0.771	0.068
Matching	0.869	0.061	0.772	0.064
Dice	0.868	0.061	0.771	0.063
Jaccard	0.867	0.062	0.772	0.065
Skalar	0.867	0.06	0.774	0.063
Cosinus	0.868	0.06	0.771	0.063

Die Tabelle 4.3 beschreibt gut, dass die F1-Bewertung und die Standardabweichung mit unterschiedlichen Metriken auf den fünf Datensätzen fast dieselben Werte wie 'Normal' produzieren.

Der nächste Versuch beschreibt die F1-Bewertung zur steigenden Anzahl von Benutzermarkierungen. In Schritten von 5 wird die Anzahl von Markierungen bis zu Grenze von 100 erhöht. Zu jeder Markierungsanzahl werden 3 Wiederholungen mit jeder Metrik gestartet, die dieselbe Gruppierung für jede Konfiguration beinhalten. Der erste Versuch wird theoretisch auf 3 Wiederholungen gekürzt und lediglich die Anzahl an Markierungen wird geändert.

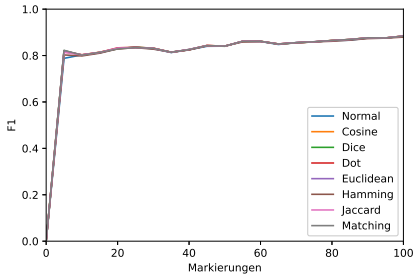

Abbildung 4.1: Anzahl Markierungen - Flights

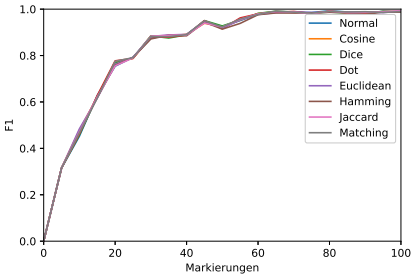

Abbildung 4.2: Anzahl Markierungen - Hospital

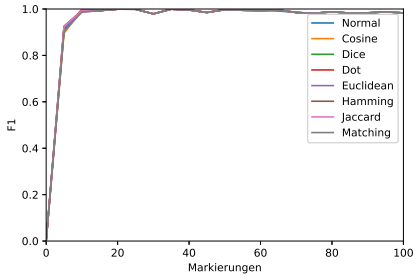

Abbildung 4.3: Anzahl Markierungen - Beers

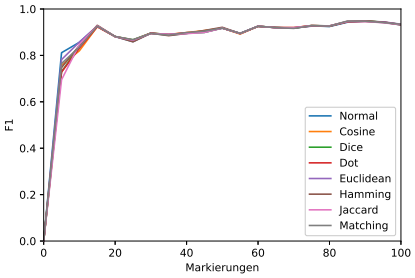

Abbildung 4.4: Anzahl Markierungen - Movies

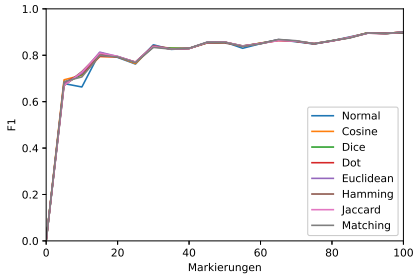

Abbildung 4.5: Anzahl Markierungen - Rayyan

Auch der zweite Versuch zeigt in homogenen Gruppen keine Verbesserung. Die Abbildungen 4.1 bis 4.5 zeigen keinen signifikanten Anstieg bei Verwendung einer Metrik im Vergleich zu 'Normal'. Lediglich im Bereich von 5 bis 20 Markierungen sind kleine Schwankungen ersichtlich. Nur bei dem Datensatz Rayyan, Abbildung 4.5, ist 'Normal' bei 15 Markierungen klar schlechter als eine Propagierung mit Metrik. Allerdings ist 'Normal' beim Datensatz Movies_1 bei 5 Markierungen besser. Diese Unterschiede können durch die geringe Anzahl an Wiederholungen pro Markierungsanzahl entstehen. Die Gruppierung und die Wahl der benutzermarkierten Datenpunkte

spielen besonders im Bereich von wenig Markierungen eine Rolle, ob eine Metrik unterschiedliche Werte liefert oder nicht.

Der zweite Teil des ersten Versuches bezieht die heterogenen Gruppen wie im Algorithmus 3 bereits dargestellt mit ein. Unter Berücksichtigung der heterogenen Gruppen wird die Anzahl der Trainingsdaten erhöht, also mehr Quantität an Daten. Die Qualität kann erst anhand der Ergebnisse ermittelt werden, da es sich um einen teilüberwachten Algorithmus handelt.

Tabelle 4.4: Metrischer Ansatz - Heterogene Gruppen,
σ = Standardabweichung

Metrik	Beers		Flights		Hospital	
	$F1$	σ	$F1$	σ	$F1$	σ
Normal	0.994	0.013	**0.828**	0.024	0.717	0.089
Euclidean	0.994	0.013	**0.828**	0.025	0.714	0.086
Matching	0.994	0.013	0.827	0.024	**0.718**	0.086
Hamming	0.994	0.013	0.827	0.024	0.717	0.086
Dice	0.994	0.012	**0.828**	0.024	0.716	0.084
Jaccard	0.994	0.012	0.827	0.025	0.715	0.088
Dot	0.994	0.012	0.827	0.025	0.716	0.085
Cosine	0.994	0.012	0.827	0.024	0.716	0.084

Metrik	Movies		Rayyan	
	$F1$	σ	$F1$	σ
Normal	0.875	0.054	0.781	0.054
Euclidean	0.875	0.054	0.785	0.051
Matching	0.874	0.053	0.784	0.053
Hamming	0.875	0.055	0.782	0.056
Dice	0.875	0.054	0.784	0.052
Jaccard	0.874	0.054	0.785	0.052
Dot	**0.876**	0.054	**0.786**	0.052
Cosine	**0.876**	0.054	0.784	0.052

Die Ergebnisse aus Tabelle 4.4 weisen größere Unterschiede auf als die Ergebnisse des metrischen Ansatzes von homogenen Gruppen. Vor allem im Datensatz Rayyan hat eine Metrik erfolgreich funktioniert, weil die meisten Metriken 1% bis 2% besser sind als 'Normal'. In den restlichen Datensätzen liegt 'Normal' eher im Mittelfeld, allerdings meistens nur mit $\pm$1% Unterschied. Die Standardabweichung weist keine Besonderheiten auf, ähnlich zum ersten Versuch.

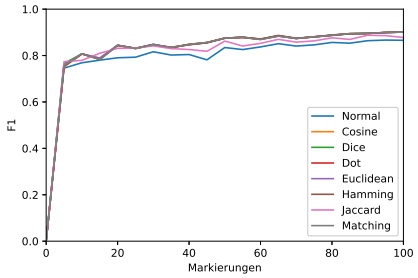

Abbildung 4.6: Anzahl Markierungen - Flights

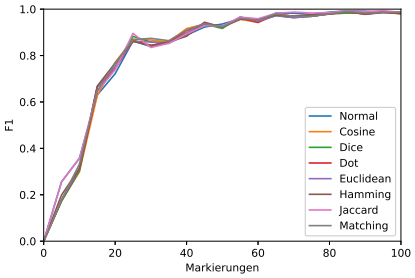

Abbildung 4.7: Anzahl Markierungen - Hospital

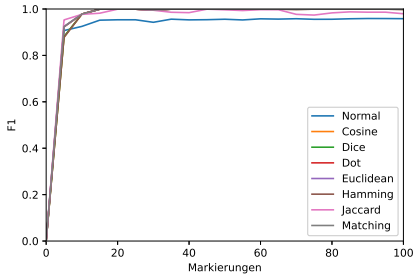

Abbildung 4.8: Anzahl Markierungen - Beers

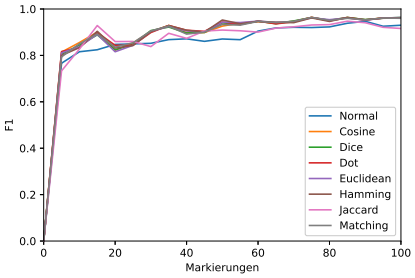

Abbildung 4.9: Anzahl Markierungen - Movies

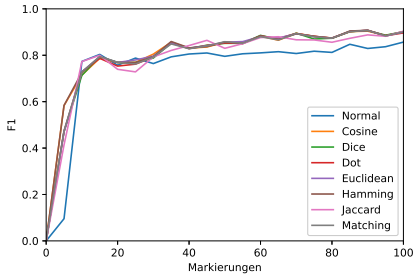

Abbildung 4.10: Anzahl Markierungen - Rayyan

Betrachtet man den zweiten Versuch mit heterogenen Gruppen, gibt es schon deutliche Veränderungen. Es wurde zur Kontrolle zusätzlich die 'Majority' Konfiguration von Raha mit in Betrachtung gezogen, hier bezeichnet als 'Normal_het'. Dies ist die Mehrheitskonfiguration in heterogenen Gruppen. Sie passt sich in Datensätzen mit einer geringen Anzahl an heterogenen Gruppen der 'Normal' Kofiguration an. Zum Beispiel besitzt der Datensatz Hospital kaum heterogene Gruppen mit steigender Markierungsanzahl. Die Datensätze Flights, Movies oder Rayyan haben dem hingegen auch bei vielen Markierungen noch heterogene Gruppen (s. Abbildung 3.1). In den Abbil-

dungen 4.6, 4.7, 4.9 und 4.10 kann man gut erkennen, dass 'Normal_het' die heterogenen Gruppen nicht korrekt markiert und deshalb das Ergebnis schlechter ist als 'Normal'. Die Metriken Euclidean, Machting, Hamming, Dice, Dot und Cosine verhalten sich in allen Datensätzen sehr ähnlich und liegen meistens minimal über 'Normal'. Jaccard hingegen ist meistens gleich zu 'Normal' und hat zusätzlich einen ähnlichen Verlauf im Gegensatz zu den restlichen Metriken. Die Qualität der Trainigsdaten scheint trotz der größeren Quantität nicht viel beeinflusst worden zu sein. Allerdings sind die Ergebnisse minimal besser als die 'Normal' Konfiguration und vermutlich ist die Smoothness Annahme noch nicht optimal erfüllt.

4.3 Dichtebasierter Ansatz

Der metrische Ansatz hat gezeigt, dass eine Metrik zu minimal besseren F1-Bewertungen führt. Allerdings kann es bei Verwendung einer Metrik zu Komplikationen mit der Struktur bzw. dem Aufbau einer Gruppe kommen. Im metrischen Ansatz wurde das Problem betrachtet, dass ein Punkt eine gesamte Gruppe markieren kann. Das Problem wurde mithilfe der Metrik minimiert, aber nicht optimal gelöst. Eine bessere Lösung bietet sich, wenn die Struktur einer Gruppe betrachtet wird. Was sind dichte Gebiete und wo befinden sie sich? Dichte Gebiete sind 'Wolken' aus Datenpunkten. Zur Bestimmung eines dichten Gebietes, wird ein Maximalabstand und eine Anzahl von Nachbarn bestimmt [20]. Der Maximalabstand gibt an, ab welcher Distanz ein Datenpunkt einen anderen Datenpunkt als Nachbar betrachtet. Ist diese Distanz gegeben, können zu jedem Datenpunkt innerhalb der Gruppe die Nachbarsdatenpunkte bestimmt werden. Ein dichtes Gebiet sind Datenpunkte, die eine gewisse Anzahl von Nachbarn haben. Wird eine Gruppe in mehrere dichte Gebiete unterteilt, gibt es drei Arten von Datenpunkten, die Kerndatenpunkte, dichte-erreichbare Datenpunkte und Ausreißer-Datenpunkte [21]. Als Kerndatenpunkte werden alle Datenpunkte bezeichnet, die die Anzahl an Nachbarn erreicht haben. Wird die Anzahl an Nachbarn nicht erreicht, gibt es die dichte-erreichbaren Datenpunkte, die als Nachbar einen Kerndatenpunkte haben, und die Ausreißer Datenpunkte, die außerhalb eines dichten Gebietes liegen. Um die Markierung einer gesamten Gruppe basierend auf einem kleinen Bruchteil von markierten Datenpunkten zu vermeiden, wird der Ansatz der dichte Gebiete verwendet. Ein Datenpunkt, der vom Benutzer markiert wurde, liegt in einem dichten Gebiet oder nicht. Befindet sich in einer homogenen Gruppe ein markierter Datenpunkt und ist dieser außerhalb eines dichten Gebietes, dann wird nur der benutzermarkierte Datenpunkt den Trainingsdaten übergeben. Befindet sich ein markierter Datenpunkt hingegen in oder an einem dichten Gebiet, dann wird ausschließlich im dichten Gebiet die Markierung propagiert. Der dichtebasierte Ansatz achtet dementsprechend deutlich mehr auf die Qua-

lität der Trainingsdaten, als auf die Quantität.

Die heterogenen Gruppen erfüllen die Cluster Annahme bisher nicht, weshalb der zweite Ansatz in heterogenen Gruppen auf Subclustering [22] beruht. Subclustering ist das wiederholte Gruppieren einer bereits existenten Gruppe in Untergruppen, um Merkmale noch genauer darzustellen. Es existiert nun eine heterogene Gruppe mit mehr als zwei Markierungen. Diese Gruppe wird als Datensatz betrachtet und wiederholt neu gruppiert. Ziel ist es eine Aufteilung zu erhalten, bei der ausschließlich homogene Untergruppen existieren. Dieser Ansatz beruht auf dem Prinzip von hierachischem Clustering. Wird der gesamte Datensatz als große Gruppe betrachtet, existiert mindestens eine Konstellation mit homogenen Untergruppen. Dies gilt, weil jeder Datenpunkt nur eine Markierung besitzen kann und somit homogene Gruppen existieren müssen, wenn jeder Datenpunkt eine Gruppe darstellt. Die Gruppe also wird solange in Untergruppen aufgeteilt, bis jede Untergruppe homogen oder nicht markiert ist. Somit ist nicht nur in homogenen Gruppen, sondern auch in heterogenen Gruppen die Smoothness Annahmen erfüllt. Die homogenen Untergruppen werden wie homogene Gruppen mithilfe der dichten Gebiete markiert.

Die Gewichtung für homogene und heterogene Gruppen ist relativ ähnlich. Befindet sich ein Datenpunkt in einer homogenen Gruppe und liegt im selben dichten Gebiet wie ein benutzermarkierter Datenpunkt, dann ist seine Gewichtung 1. Es lässt sich zusätzlich in dichten Gebieten mit Markierung eine Metrik anwenden, die eine Gewichtung zwischen 0 bis 1 liefert, ähnlich wie im metrischen Ansatz. Es gilt jedoch noch zu definieren, ab wann ein dichtes Gebiet vorliegt. Hierzu müssen die Parameter Maximalabstand und Anzahl von Nachbarn bestimmt werden. Die Anzahl von Nachbarn ist als standardgemäß 5 gesetzt und der Maximalabstand kann mithilfe der Distanzen innerhalb der Gruppe bestimmt werden [23]. Zur Bestimmung wird die Distanz des naheliegendsten Datenpunktes zu jedem Datenpunkt der Gruppe berechnet. Anschließend werden die Distanzen aufsteigend sortiert. Es existieren nun eine Liste L, die n Distanzen aufsteigend sortiert enthält und es wird der Index $0 < i \leq n$ betrachtet. Beträgt die Distanz $L[i]$ mehr als 1% zur Distanz $L[i-1]$, dann wird die Distanz $L[i]$ als Maximalabstand gewählt. Ein Datenpunkt ist folglich ein Kerndatenpunkt, wenn 5 weitere Datenpunkt im Radius von $L[i]$ liegen und wird als dichtes Gebiet behandelt.

4.3.1 Aufbau

Der Aufbau vom dichtebasierten Ansatz beginnt wie der metrische Ansatz mit der Trennung der benutzermarkierten Datenpunkte von den restlichen Datenpunkten einer Gruppe (s. Algorithmus 1). In homogenen Gruppen wird danach mithilfe von NearestNeighbors [24] zu jedem Datenpunkt der naheliegendste Nachbarpunkt ermittelt und die Distanz gespeichert. Alle Distanzen werden aufsteigend sortiert und es wird nach der 1% Steigung

gesucht. Ist der Maxmialabstand bestimmt, werden DBSCAN [21] die beiden Parameter Maximalabstand und Nachbaranzahl übergeben. DBSCAN kann anschließend die dichten Gebiete einer Gruppe ermitteln. Jeder Datenpunkt besitzt nun eine Kennzeichnung, zu welchem dichten Gebiet er gehört, oder ob er als Ausreißer gekennzeichnet ist. Die dichten Gebiete, die ein benutzermarkierten Datenpunkt enthalten, bekommen dessen Markierung und eine Gewichtung von 1. Alle Datenpunkte der Gruppe, die nicht markiert sind oder außerhalb eines markierten dichten Gebietes liegen, erhalten die Gewichtung 0 und werden als irrelevant behandelt. Innerhalb von dichten Gebieten kann mittels einer Metrik noch zusätzlich die Gewichtung angepasst werden. Beginnend mit dem Algorithmus 1 entsteht die Datenstruktur *datenpunkte*. Sei G eine Gruppe aus einer Spalte von *datenpunkte* und $NearestNeighbors$, $DBSCAN$ zusätzliche Funktionen.

Algorithmus 5 Dichtebasierter Ansatz - Homogene Gruppen , keine Metrik

$G \leftarrow [\{markiert\}, \{nichtmarkiert\}]$
$liste \leftarrow NearestNeighbors(G)$
$liste \leftarrow sort(L, aufsteigend)$
$max\_dist \leftarrow 0$, $nachbarn \leftarrow 5$
for jede Distanz d_i in $liste$ **do** ▷ Maximalabstand berechen
  if $d_i > 1.01 * d_{i-1}$ **then**
    $max\_dist \leftarrow d_i$ break
  end if
end for
$markierung \leftarrow get\_gruppenmarkierung(G)$
$db \leftarrow DBSCAN(G, max\_dist, nachbarn)$
for jeden Punkt p in $markiert$ **do**
  for jeden Punkt p_{db} in $db[p]$ **do** ▷ Dichtes Gebiet von p
    $G[p_{db}] \leftarrow (1, markierung)$
  end for
end for

Damit eine Metrik verwendet wird, muss im letzten Schritt nun jeder Datenpunkt innerhalb eines dichten markierten Gebietes mit dem nächsten benutzermarkierten Datenpunkt im dichten Gebiet verglichen werden. Der Vergleich auf Abstand bzw. Ähnlichkeit geschieht mit denselben Metriken wie im metrischen Ansatz. Die Variablen *max_dist* und *nachbarn* bleiben gleich dem Algorithmus 5 und der Maximalabstand wurde bereits bestimmt. Zu berücksichtigen sind dichte Gebiete, die mehr als eine Benutzermarkierung enthalten. Bei der Trennung wird jedem nicht markierten Datenpunkt eine Gewichtung von 0 zugewiesen. Deshalb kann in dichten Gebieten mit mehreren Markierungen einfach das größte Gewicht je Datenpunkt gespeichert werden.

Algorithmus 6 Dichtebasierter Ansatz - Homogene Gruppen , mit Metrik

```
G ← [{markiert}, {nichtmarkiert}]
markierung ← get_gruppenmarkierung(G)
db ← DBSCAN(G, max_dist, nachbarn)
for jeden Punkt p in markiert do
    for jeden Punkt p_db in db[p] do                ▷ Dichtes Gebiet von p
        w ← metrik(p, p_db)
        if get_gewichtung(G[p_db]) < w then         ▷ nur größtes Gewicht
            G[p_db] ← (w, markierung)
        end if
    end for
end for
```

Um auch in heterogenen Gruppen die Benutzermarkierungen zu erweitern, wird im dichtebasierten Ansatz auf Subclustering gesetzt. Eine Gruppe wird solange in Untergruppen aufgeteilt, bis alle Untergruppen homogen sind. Um die Untergruppen einer Gruppe zu ermitteln, wird der KMeans [25] Algorithmus verwendet, mit dem Parameter $n\_clusters$, der die Anzahl an Untergruppen festlegt. Die Anzahl an Untergruppen wird solange erhöht, bis jede Untergruppe homogen oder nicht markiert ist. Die globale Variabel $SUB\_CLUSTER\_SIZE$ gibt die maximale Anzahl an Untergruppen an. Wird diese Anzahl erreicht und es existiert noch mindestens eine heterogene Untergruppe, dann werden lediglich die benutzermarkierten Datenpunkte aus der heterogenen Gruppe als Trainingsdaten verwendet. Diese Grenze existiert, damit Rechenzeit gespart werden kann. Besteht kein Interesse an der Rechenzeit, kann die Variabel sehr hoch eingestellt werden. Sobald die erste homogene Untergruppen Konstellation erreicht ist, wird der Algorithmus 5 oder 6 angewendet. Die homogenen Untergruppen werden gleich den homogenen Gruppen markiert und eine zusätzliche Gewichtung wie in Algorithmus 6 ist möglich. Es sei G eine Gruppe aus einer Spalte, $KMeans$ eine Funktion und $Algorithmus_{5,6}$ ein Verweis auf Algorithmus 5 und 6.

Algorithmus 7 Dichtebasierter Ansatz - Heterogene Gruppen

```
G ← [{markiert}, {nichtmarkiert}]
anzahl_untergruppen ← 0
for anzahl_untergruppen < SUB_CLUSTER_SIZE do
    homogene ← True
    untergruppen ← KMeans(anzahl_untergruppen)
    for jede untergruppe ug in untergruppen do
        if ug ist heterogen then
            homogen ← False
        end if
    end for
    if homogen then
        Algorithmus_{5,6}(untergruppen) break
    end if
end for
```

4.3.2 Ergebnisse

Dieselben zwei Versuche aus dem metrischen Ansatz werden auch im dichtebasierten Ansatz betrachtet. Zum einen der direkte Vergleich mit 100 Wiederholungen bei gleicher Gruppierung und zum anderen die unterschiedliche Anzahl der Benutzermarkierungen. Auch die Datensätze an denen getestet wird bleiben gleich und die Standardkonfiguration in Raha wird als 'Normal' gekennzeichnet. Getestet werden die Konfigurationen aus Tabelle 4.5.

Tabelle 4.5: Konfigurationen dichtebasierter Ansatz

Name	Homogene Gruppen		Heterogen Gruppen	
	Ungewichtet	Gewichtet	Ungewichtet	Gewichtet
Normal	X			
Marked				
Prop7	X			
Prop8		X		
Prop7_het4	X		X	
Prop7_het5	X			X
Prop8_het4		X	X	
Prop8_het5		X		X

Die Konfiguration 'Marked' erhält als Trainingsdaten für den Klassifikator nur die Benutzermarkierungen, es wird keine Label Propagation angewendet. Die Konfigurationen 'Prop7' und 'Prop8' verwenden den Algorithmus 5,6 für homogene Gruppen des dichtebasierten Ansatz und die Erweiterungen 'het4' und 'het5' verwenden den Algorithmus 7 für heterogene Gruppen. Die Metrik bei gewichteten Konfigurationen ist die euklidische

Distanz. Zum einen, weil die Ergebnisse vom metrischen Ansatz für die euklidische Distanz ähnlich bis besser als 'Normal' sind, und zum andern, weil der Maximalabstand für die Berechnung dichter Gebiete mit der euklidischen Distanz bestimmt wird.

Der erste Versuch startet mit Konfigurationen aus Tabelle 4.5, 20 Benutzermarkierungen, $SUB\_CLUSTER\_SIZE$ ist 20, 5 Nachbarn für ein dichtes Gebiet und als Metrik die euklidische Distanz.

Tabelle 4.6: Dichtebasierter Ansatz - Versuch 1,
σ = Standardabweichung

Konfiguration	Beers		Flights		Hospital	
	$F1$	σ	$F1$	σ	$F1$	σ
Normal	0.992	0.015	0.829	0.024	0.722	0.081
Marked	**0.997**	0.005	0.832	0.023	0.732	0.073
Prop7	0.992	0.015	**0.833**	0.022	**0.752**	0.075
Prop8	0.992	0.014	**0.833**	0.022	0.742	0.079
Prop7_het4	0.996	0.005	**0.833**	0.023	0.728	0.077
Prop7_het5	0.996	0.005	**0.833**	0.022	0.725	0.078
Prop8_het4	**0.997**	0.005	0.832	0.023	0.725	0.079
Prop8_het5	0.996	0.005	0.832	0.023	0.728	0.075

Konfiguration	Movies		Rayyan	
	$F1$	σ	$F1$	σ
Normal	0.863	0.062	0.778	0.053
Marked	0.869	0.058	**0.781**	0.061
Prop7	0.857	0.057	0.78	0.061
Prop8	0.859	0.055	0.779	0.064
Prop7_het4	0.87	0.058	0.778	0.063
Prop7_het5	**0.872**	0.057	**0.781**	0.063
Prop8_het4	0.871	0.056	0.779	0.062
Prop8_het5	0.869	0.058	0.778	0.062

In Tabelle 4.6 sieht man sehr gut, dass alle Konfigurationen eine bessere oder gleiche F1-Bewertung als die Konfiguration 'Normal' haben. Die Standardabweichungen verhalten sich außer beim Datensatz Rayyan ähnlich zur F1-Bewertung. Die Konfiguration 'Marked' ist ein sehr gutes Beispiel dafür, dass die Qualität der Trainingsdaten wichtiger ist als die Quantität. Auf jedem Datensatz ist eine Verbesserung zwischen 'Marked' und 'Normal' zu verzeichnen, ähnlich wie bei den Konfigurationen mit Erweiterung von heterogenen Gruppen. Im Detail ist besonders beim Datensatz Hospital die 'Prop7' Konfiguration auffällig, die 3% besser als 'Normal' ist und auch im Verhältnis zu den anderen Konfigurationen klar besser ist. Die Konfigurationen mit Erweiterung von heterogenen Gruppen schneiden zu 'Prop7' auf dem Datensatz Hospital verhältnismäßig schlecht ab. Allerdings ist 'Prop7' und 'Prop8' beim Datensatz Beers eindeutig schlechter, als die Konfigurationen mit Erweiterung und 'Marked'. Die durchschnittlich beste Konfiguration ist 'Prop7_het4', auf allen Datensätzen ist die F1-Bewertung besser als 'Normal' und auch die Standardabweichung befindet sich im oberen Mittelfeld im Vergleich zu allen Konfigurationen.

Der zweite Versuch wird mit der gleichen Anzahl von Nachbarn und der euklidischen Distanz als Metrik ausgeführt, wie im metrischen Ansatz. Es wird die Anzahl an Benutzermarkierungen erhöht und je Erhöhung drei Wiederholungen mit den Konfigurationen aus Tabelle 4.5 gestartet.

In den Abbildungen 4.11 bis 4.15 ist gut zu erkennen, dass sich die Konfigurationen mit Erweiterung und 'Marked' sehr ähnlich verhalten. Besonders im Bereich von wenig Benutzermarkierungen ist bei Konfigurationen mit Erweiterung von heterogenen Gruppen ein schnellerer Anstieg der F1-Bewertung bemerkbar. Dies ist zum Beispiel in Abbildung 4.14 und 4.15 gut sichtbar. Die Konfigurationen 'Prop7' und 'Prop8' verhalten sich dem hingegen eher gleich der 'Normal' Konfiguration, weil die heterogenen Gruppen nicht beachtet werden. Vor allem in den Abbildungen 4.11 und 4.15 befindet sich 'Normal' häufig unterhalb allen anderen Konfigurationen, jedoch in Abbildung 4.14 befinden sich 'Prop7' und 'Prop8' unterhalb von 'Normal'. Die Ergebnisse für die Skalierbarkeit auf dem Datensatz Tax wurden lediglich für die Konfiguration 'Prop7' mit ungewichteter Erweiterung in heterogenen Gruppen ('Prop7_het4') ausgeführt und die F1- Bewertung liegt zwischen 0.91 und 0.99.

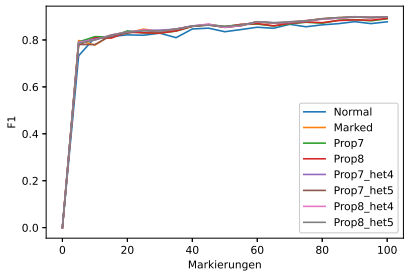

Abbildung 4.11: Anzahl Markierungen - Flights

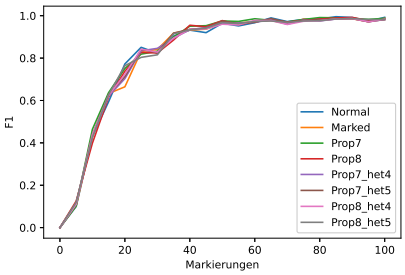

Abbildung 4.12: Anzahl Markierungen: Hospital

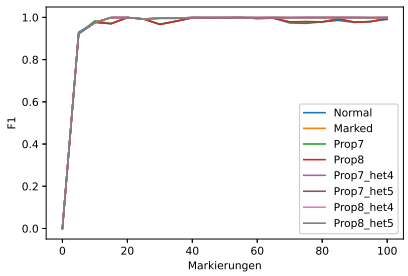

Abbildung 4.13: Anzahl Markierungen - Beers

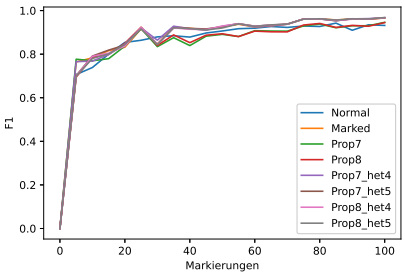

Abbildung 4.14: Anzahl Markierungen - Movies

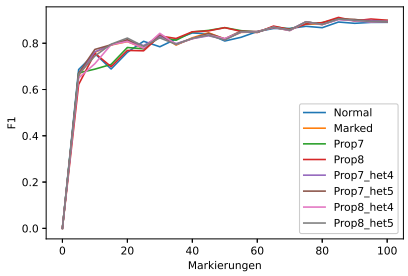

Abbildung 4.15: Anzahl Markierungen - Rayyan

4.4 Evaluierung

Die Versuche zeigen, dass es eine Verbesserung der F1-Bewertung gibt. Im ersten Versuch des metrischen Ansatzes hat die einfache Single Linkage Metrik in homogenen Gruppen zu keiner Verbesserung der F1-Bewertung oder der Standardabweichung geführt. Allerdings gibt es bereits bei der Anwendung einer Single Linkage Metrik in heterogenen Gruppen eine minimale Steigerung der F1-Bewertung im Vergleich zur Standardkonfiguration von Raha. Die 'Euclidean' Metrik hat mit einigen anderen besser abgeschnitten und wird aufgrund der Distanzbestimmung in $DBSCAN$ auch für den dichtebasierten Ansatz verwendet. Der dichtebasierte Ansatz ist mehr auf die Qualität der Markierungen fokussiert. Weniger Trainingsdaten für den Klassifikator, dafür wurden aber wahrscheinlich die richtige Markierungen propagiert. Die Konfiguration 'Marked' hat im Vergleich zu 'Normal' und den Konfigurationen des dichtebasierten Ansatzes (s. Tabelle 4.5) erstaunlich gute F1-Bewertungen erbracht. Die beiden Konfigurationen 'Prop7' mit gewichteter und ungewichteter Erweiterung sind auf allen Datensätzen vergleichsweise solide gelaufen. Zusätzlich sind die beiden Konfigurtaionen in Bereich mit wenig Benutzermarkierungen besser, als 'Normal', 'Prop7' und 'Prop8'. Besonders in den beiden Datensätzen Movies und Rayyan, die verhältnismäßig mehr heterogene Gruppen besitzen, sind die Konfigurationen mit Erweiterung von heterogenen Gruppen effizienter (s. Grafik 4.14, 4.15). In Tabelle 4.6 haben die Konfigurationen 'Prop7' und 'Prop8', ohne Erweiterung, auf dem Datensatz Hospital eine ziemlich gute F1-Bewertung. Der Datensatz Hospital enthält fast keine heterogenen Gruppen und eine Gewichtung mittels einer Metrik scheint unvorteilhaft für dichte Gebiete zu sein, weil 'Prop7' um 1% besser ist als 'Prop8'. Teilweise ist dieses Verhalten auch bei 'Prop8' mit Erweiterung erkennbar. 'Prop8' mit gewichteter und ungewichteter Erweiterung schneidet im Vergleich zu 'Normal' besser ab auf fast allen Datensätzen, jedoch ist 'Prop7' mit Erweiterung meistens noch ein bisschen besser. In Bezug auf die Standardabweichung ist eine einfach Metrik weder in homogenen noch in heterogenen Gruppen besser, als 'Normal'. Die dichtebasierenden Konfigurationen hingegen haben eine minimal geringere Standardabweichung, wobei die Konfigurationen 'Prop7' und 'Prop8' mit Erweiterungen von heterogenen Gruppen und 'Marked' auf fast allen Datensätzen gut abschneiden. Aus diesem Grund ist allgemein die optimale Konfiguration 'Prop7' mit ungewichteter Erweiterung knapp gefolgt von 'Prop7' mit gewichteter Erweiterung. Abschließend ist eine Erhöhung der Qualität, zum Beispiel 'Prop7' mit Erweiterung, die bessere Wahl, als die Erhöhung der Quantität, weil die Smoothness Annahme nicht optimal erfüllt ist.

Kapitel 5

Verwandte Arbeiten

Für die Optimierung von Raha ist eine Gewichtung nach der Label Propagagtion gefordert, die die Stärke einer Markierung angibt. Ansätze aus dem Bereich 'weighted Clustering' verwenden meistens eine Gewichtung, die bereits vor der Label Propagation definiert wurde. Die unterschiedlichen Eigenschaften eines Datenpunktes erhalten eine eigene Gewichtung, die bei der Label Propagation verwendet wird, den Ansatz verwendet auch der Artikel 'An Enhanced Regularized k-Means Type Clustering Algorithm With Adaptive Weights' [26]. Damit eine Gewichtung in Raha nützlich ist, muss sie nach der Label Propagation zusammen mit den propagierten Markierungen dem Klassifikator übergeben werden. Die Benutzung des Parameters *sample_weights* ermöglichst dies. Zur Gewichtung wird eine Single Linkage genutzt, weil es sehr wenig Benutzermarkierungen pro Gruppe gibt. Der Artikel 'Sample-weighted clustering methods' [27] beschreibt eine ähnlichen Ansatz zur Gewichtung mithilfe von *sample_weights*. Im Artikel 'Color text extraction with selective metric-based clustering' [28] wird die Verwendung einer Gewichtung in K-Means mittels metrischen Distanzen umgesetzt. Der dichtebasierte Ansatz wird im Artikel 'Dynamic graph-based label propagation for density peaks clustering' [29] für homogene Gruppen vorgestellt. Die Anwendung der dichtebasierenden Label Propagation gelingt mithilfe der Optimierung des Epsilon Parameters in DBSCAN.

Kapitel 6

Fazit

Die Annahme die Propagierung der Markierungen in Gruppen zu optimieren hat sich bestätigt. Wie bereits gezeigt wurde, erlangt die dichtebasierte Konfiguration 'Prop7' mit gewichteter und ungewichteter Erweiterung bessere F-Bewertungen als die Standardkonfiguration von Raha. Die Verwendung einer Metrik in ausschließlich homogenen Gruppen markiert dieselben Datenpunkte wie 'Normal' und fügt lediglich eine Gewichtung hinzu. Der Versuch die Quantität der Trainingsdaten zu erhöhen, indem heterogene Gruppen markiert werden, verlief wenig erfolgreich. Die minimale Verbesserung entsteht durch die korrekte Markierung von einigen mehr Datenpunkten. Die Standardkonfiguration ' Normal' belässt es bei der Markierung von homogenen Gruppen und die Konfiguration 'Majority' markiert Datenpunkte in heterogenen Gruppen falsch. Die Metrik für heterogene Gruppen markiert einen Großteil der Datenpunkte innerhalb einer heterogenen Gruppe richtig. Allerdings ist der Anteil an heterogenen Gruppen nicht besonders groß und die zusätzlich richtig markierten Datenpunkte erwirken nur die minimale Verbesserung der F1-Bewertung. Das Problem, dass ein einzelner Ausreißer-Datenpunkt eine komplette Gruppe markieren kann, wurde erst durch die dichten Gebiete minimiert. Jeder Datenpunkt einer Gruppe kann nur an Nachbardatenpunkte die Markierung erweitern, die restlichen Datenpunkte der Gruppe bleiben nicht markiert. Die Smoothness Annahme wird durch die dichten Gebiete optimiert und somit werden alle drei theoretischen Annahmen der Label Propagation verwendet. Dies zeigen auch die Ergebnisse des dichtebasierten Ansatzes, Qualität vor Quantität. Schon die einfache Konfiguration 'Marked' hat erstaunlich gute F1-Bewertungen erreicht ohne das Propagieren von Markierungen. Die dichtebasierten Konfiguration haben gut veranschaulicht, was die Smoothness Annahme bewirkt. Vorwissen kann dem Benutzer allerdings zu noch besseren Ergebnissen verhelfe. Besitzt der Datensatz viele heterogene Gruppen, sollte die Konfiguration 'Prop7' mit einer Erweiterung gewählt werden, besitzt der Datensatz jedoch fast keine heterogenen Gruppen ist 'Prop7' die beste Wahl. Das Testen der

verschiedenen Ansätze wurde durch die zufällige Wahl der Markierungen beeinflusst. Die exakt selben Ergebnisse lassen sich nicht erneut erzeugen und durch die hohe Anzahl von Wiederholungen sollen nur näherungsweise Mittelwerte bestimmt werden. Im Großen und Ganzen ist die Optimierung ein Erfolg gewesen. Die F1-Bewertung und die Standardabweichung von dichtebasierten Konfigurationen mit der Erweiterung heterogener Gruppen sind eindeutig höher als die Werte der Standardkonfiguration von Raha.

Literatur

[1] Rebekah Carter. Accessed: 10-03-22. URL: https://findstack.com/de/big-data-statistics/.

[2] Mohammad Mahdavi u. a. "Raha: A Configuration-Free Error Detection System". In: *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. Hrsg. von Peter A. Boncz u. a. ACM, 2019, S. 865–882. DOI: 10.1145/3299869.3324956. URL: https://doi.org/10.1145/3299869.3324956.

[3] O. Forster. *Analysis 2: Differentialrechnung im IRn, gewöhnliche Differentialgleichungen*. Grundkurs Mathematik. Springer Fachmedien Wiesbaden, 2017. ISBN: 9783658194116. URL: https://books.google.de/books?id=qZo3DwAAQBAJ.

[4] Yutaka Sasaki u. a. "The truth of the f-measure. 2007". In: *URL: https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07. pdf* (2007). Accessed: 05-04-22.

[5] Eric W. Weisstein. *Distance*. Accessed: 15-03-22. URL: https://mathworld.wolfram.com/Distance.html.

[6] Amit Singhal. "Modern Information Retrieval: A Brief Overview". In: *IEEE Data Eng. Bull.* 24.4 (2001), S. 35–43. URL: http://sites.computer.org/debull/A01DEC-CD.pdf.

[7] S. Lipschutz und M. Lipson. *Schaum's Outline of Linear Algebra Fourth Edition*. Schaum's Outline Series. McGraw-Hill Education, 2008. ISBN: 9780071543538. URL: https://books.google.de/books?id=RsPqtjiW50YC.

[8] John C Gower und Matthijs J Warrens. "Similarity, dissimilarity, and distance, measures of". In: *Wiley StatsRef: Statistics Reference Online* (2014), S. 1–11.

[9] Prof. Dr. Roland Gabriel. Accessed: 11-03-22. Sep. 2020. URL: https://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/technologien-methoden/Statistik/Clustering.

[10] SAS Institute. Version SAS/STAT(R) 9.2 User's Guide, Second Edition. Accessed: 15-03-22. URL: https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_cluster_sect012.htm.

[11] Xiaojin Zhu und Zoubin Ghahramani. "Learning from labeled and unlabeled data with label propagation". In: (2002).

[12] Olivier Chapelle, Bernhard Schölkopf und Alexander Zien, Hrsg. *Semi-Supervised Learning*. The MIT Press, 2006. ISBN: 9780262033589. DOI: 10.7551/mitpress/9780262033589.001.0001. URL: https://doi.org/10.7551/mitpress/9780262033589.001.0001.

[13] Gyeongho Kim. "Recent Deep Semi-supervised Learning Approaches and Related Works". In: *CoRR* abs/2106.11528 (2021). arXiv: 2106.11528. URL: https://arxiv.org/abs/2106.11528.

[14] Olivier Chapelle, Jason Weston und Bernhard Schölkopf. "Cluster Kernels for Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*. Hrsg. von Suzanna Becker, Sebastian Thrun und Klaus Obermayer. MIT Press, 2002, S. 585–592. URL: https://proceedings.neurips.cc/paper/2002/hash/d6288499d0083cc34e60a077b7c4b3e1-Abstract.html.

[15] Dennis Heinen. "Numerische Methoden zur Analyse hochdimensionaler Daten". Beispiel 1.2. dissertation. Georg-August-Universität Göttingen, 2014. URL: http://dx.doi.org/10.53846/goediss-4680.

[16] Dipl.-Ing. (FH) Stefan Luber und Nico Litzel. Accessed: 10-03-22. Juli 2019. URL: https://www.bigdata-insider.de/was-ist-datenbereinigung-a-843546/.

[17] F. Pedregosa u. a. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011). sklearn.ensemble.GradientBoostingClassifier, S. 2825–2830.

[18] URL: https://github.com/maxsiebenthaler/Raha-Extention.git.

[19] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), S. 861–874.

[20] Ricardo JGB Campello u. a. "Density-based clustering". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10.2 (2020), e1343.

[21] Martin Ester u. a. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Bd. 96. 34. 1996, S. 226–231.

[22] Tingwei Feng Ruizhang Zhou Joju Mori. Online Book. URL: http://ocd.git.dbis.rwth-aachen.de/Online-Buch/Auflage_2/Subspace_Clustering/.

[23] Mohammad N.t. Elbatta. Diss. URL: http://hdl.handle.net/20.500.12358/18778.

[24] F. Pedregosa u. a. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011). sklearn. neighbors. NearestNeighbors, S. 2825–2830.

[25] F. Pedregosa u. a. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011). sklearn. cluster. KMeans, S. 2825–2830.

[26] Ziheng Wu und Zixiang Wu. "An enhanced regularized k-means type clustering algorithm with adaptive weights". In: *IEEE Access* 8 (2020), S. 31171–31179.

[27] Jian Yu, Miin-Shen Yang und E. Stanley Lee. "Sample-weighted clustering methods". In: *Computers Mathematics with Applications* 62.5 (2011), S. 2200–2208. ISSN: 0898-1221. DOI: https://doi.org/10.1016/j.camwa.2011.07.005. URL: https://www.sciencedirect.com/science/article/pii/S0898122111005591.

[28] Céline Mancas-Thillou und Bernard Gosselin. "Color text extraction with selective metric-based clustering". In: *Computer Vision and Image Understanding* 107.1 (2007). Special issue on color image processing, S. 97–107. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2006.11.010. URL: https://www.sciencedirect.com/science/article/pii/S107731420600213X.

[29] Seyed Amjad Seyedi u. a. "Dynamic graph-based label propagation for density peaks clustering". In: *Expert Systems with Applications* 115 (2019), S. 314–328. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2018.07.075. URL: https://www.sciencedirect.com/science/article/pii/S0957417418304998.

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 25.04.2022

Maximilian Siebenthaler