Big Data und der Fluch der Dimensionalität

Die effiziente Suche nach Quasi-Identifikatoren in hochdimensionalen Daten

Hannes Grunert
Lehrstuhl für Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 22
hg(at)informatik.uni-rostock.de

Andreas Heuer
Lehrstuhl für Datenbank- und Informationssysteme
Universität Rostock
Albert-Einstein-Straße 22
ah(at)informatik.uni-rostock.de

Kurzfassung

In smarten Umgebungen werden häufig große Datenmengen durch eine Vielzahl von Sensoren erzeugt. In vielen Fällen werden dabei mehr Informationen generiert und verarbeitet als in Wirklichkeit vom Assistenzsystem benötigt wird. Dadurch lässt sich mehr über den Nutzer erfahren und sein Recht auf informationelle Selbstbestimmung ist verletzt.

Bestehende Methoden zur Sicherstellung der Privatheitsansprüche von Nutzern basieren auf dem Konzept sogenannter Quasi-Identifikatoren. Wie solche Quasi-Identifikatoren erkannt werden können, wurde in der bisherigen Forschung weitestgehend vernachlässigt.

In diesem Artikel stellen wir einen Algorithmus vor, der identifizierende Attributmengen schnell und vollständig erkennt. Die Evaluierung des Algorithmus erfolgt am Beispiel einer Datenbank mit personenbezogenen Informationen.

ACM Klassifikation

K.4.1 [**Computer and Society**]: Public Policy Issues—*Privacy*; H.2.4 [**Database Management**]: Systems—*Query Processing*

Stichworte

Datenbanken, Datenschutz, Big Data

1. EINLEITUNG

Assistenzsysteme sollen den Nutzer bei der Arbeit (Ambient Assisted Working) und in der Wohnung (Ambient Assisted Living) unterstützen. Durch verschiedene Sensoren werden Informationen über die momentane Situation und die Handlungen des Anwenders gesammelt. Diese Daten werden durch das System gespeichert und mit weiteren Daten, beispielsweise mit dem Facebook-Profil des Nutzers verknüpft. Durch die so gewonnenen Informationen lassen sich Vorlieben, Verhaltensmuster und zukünftige Ereignisse berechnen. Daraus werden die Intentionen und zukünftigen Handlungen des Benutzers abgeleitet, sodass die smarte Umgebung eigenständig auf die Bedürfnisse des Nutzers reagieren kann.

In Assistenzsystemen [17] werden häufig wesentlich mehr Informationen gesammelt als benötigt. Außerdem hat der Nutzer meist keinen oder nur einen sehr geringen Einfluss auf die Speicherung und Verarbeitung seiner personenbezogenen Daten. Dadurch ist sein Recht auf informationelle Selbstbestimmung verletzt. Durch eine Erweiterung des Assistenzsystems um eine Datenschutzkomponente, welche die Privatheitsansprüche des Nutzers gegen den Informationsbedarf des Systems überprüft, kann diese Problematik behoben werden.

Zwei Hauptaspekte des Datenschutzes sind Datenvermeidung und Datensparsamkeit. In §3a des Bundesdatenschutzgesetzes [1] wird gefordert, dass

> „[d]ie Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen [...] an dem Ziel auszurichten [sind], so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen.“.

Mittels einer datensparsamen Weitergabe der Sensor- und Kontext-Informationen an die Analysewerkzeuge des Assistenzsystems wird nicht nur die Datenschutzfreundlichkeit des Systems verbessert. Bei der Vorverdichtung der Daten durch Selektion, Aggregation und Komprimierung am Sensor selbst lässt sich die Effizienz des Systems steigern. Die Privatheitsansprüche und der Informationsbedarf der Analysewerkzeuge können als Integritätsbedingungen im Datenbanksystem umgesetzt werden. Durch die Integritätsbedingungen lassen sich die notwendigen Algorithmen zur Anonymisierung und Vorverarbeitung direkt auf dem Datenbestand ausführen. Eine Übertragung in externe Programme bzw. Module, die sich evtl. auf anderen Recheneinheiten befinden, entfällt somit.

Für die Umsetzung von Datenschutzbestimmungen in smarten Umgebungen wird derzeit das PArADISE[1]-Framework entwickelt, welches insbesondere die Aspekte der Datensparsamkeit und Datenvermeidung in heterogenen Systemumgebungen realisieren soll.

In [3] stellen wir ein einfaches XML-Schema vor, mit der sich Privatheitsansprüche durch den Nutzer von smarten Systemen formulieren lassen. Dabei wird eine Anwendung

[1] Privacy-aware assistive distributed information system environment

In: G. Specht, H. Gamper, F. Klan (eds.): Proceedings of the 26^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 21.10.2014 - 24.10.2014, Bozen, Italy, published at http://ceur-ws.org.

innerhalb eines abgeschlossenen Systems in ihre Funktionalitäten aufgeteilt. Für jede Funktionalität lässt sich festlegen, welche Informationen in welchem Detailgrad an das System weitergegeben werden dürfen. Dazu lassen sich einzelne Attribute zu Attributkombinationen zusammenfassen, die angefragt werden können.

Für einen unerfahrenen Nutzer ist das Festlegen von sinnvollen Einstellungen nur schwer möglich. Die Frage, die sich ihm stellt, ist nicht die, ob er seine persönlichen Daten schützen soll, sondern vielmehr, welche Daten es wert sind, geschützt zu werden. Zur Kennzeichnung schützenswerter Daten werden u.a. sogenannte Quasi-Identifikatoren [2] verwendet. In diesem Artikel stellen wir einen neuen Ansatz vor, mit dem Quasi-Identifikatoren schnell und vollständig erkannt werden können.

Der Rest des Artikels ist wie folgt strukturiert: Kapitel 2 gibt einen aktuellen Überblick über den Stand der Forschung im Bereich der Erkennung von Quasi-Identifikatoren. Im folgenden Kapitel gehen wir detailliert darauf ein, wie schützenswerte Daten definiert sind und wie diese effizient erkannt werden können. Kapitel 4 evaluiert den Ansatz anhand eines Datensatzes. Das letzte Kapitel fasst den Beitrag zusammen und gibt einen Ausblick auf zukünftige Arbeiten.

2. STAND DER TECHNIK

In diesem Kapitel stellen wir bestehende Konzepte zur Ermittlung von Quasi-Identifikatoren (QI) vor. Außerdem werden Techniken vorgestellt, die in unseren Algorithmus eingefloßen sind.

2.1 Quasi-Identifikatoren

Zum Schutz personenbezogener Daten existieren Konzepte wie k-anonymity [16], l-diversity [8] und t-closeness [7]. Diese Konzepte unterteilen die Attribute einer Relation in Schlüssel, Quasi-Identifikatoren, sensitive Daten und sonstige Daten. Ziel ist es, dass die sensitiven Daten sich nicht eindeutig zu einer bestimmten Person zuordnen lassen. Da durch Schlüsselattribute Tupel eindeutig bestimmt werden können, dürfen diese unter keinen Umständen zusammen mit den sensitiven Attributen veröffentlicht werden.

Während Schlüssel im Laufe des Datenbankentwurfes festgelegt werden, lassen sich Quasi-Identifikatoren erst beim Vorliegen der Daten feststellen, da sie von den konkreten Attributwerten der Relation abhängen. Der Begriff Quasi-Identifikator wurde von Dalenius [2] geprägt und bezeichnet „a subset of attributes that can uniquely identify most tuples in a table".

Für „most tuples" wird häufig ein Grenzwert p festgelegt, der bestimmt, ob eine Attributkombination ein Quasi-Identifikator ist oder nicht. Dieser Grenzwert lässt sich beispielsweise in relationalen Datenbanken durch zwei SQL-Anfragen wie folgt bestimmen:

$$p = \frac{\text{COUNT DISTINCT * FROM (SELECT <attr-list> FROM table)}}{\text{COUNT * FROM table}} \quad (1)$$

Wird für p der Wert 1 gewählt, so sind die gefundenen QI mit diesem Grenzwert auch Schlüssel der Relation. Um eine Vergleichbarkeit unseres Algorithmus mit dem von Motwani und Xu zu gewährleisten, verwenden wir ebenfalls die in (1) definierte „distinct ratio" (nach [12]).

Da es für den Ausdruck „die meisten" keinen standardisierten Quantor gibt, formulieren wir ihn mit dem Zeichen: $\overset{\geq p}{\forall}$, wobei p den Prozentsatz der eindeutig identifizierbaren Tupel (t_i) angibt. Ein Quasi-Identifikator $QI := \{A_1, ..., A_n\}$ ist für eine Relation R entsprechend definiert:

Quasi-Identifikator. $\overset{\geq p}{\forall}\ t_1, t_2 \in R\ [t_1 \neq t_2 \Rightarrow \exists\ A \in QI: t_1(A) \neq t_2(A)]$

Wie beim Datenbankentwurf reicht es auch für die Angabe von Quasi-Identifikatoren aus, wenn die minimale Menge von Attributen angegeben wird, welche die Eigenschaft eines QI hat. Eine solche Menge wird als minimaler Quasi-Identifikator bezeichnet.

minimaler Quasi-Identifikator. *X ist ein minimaler Quasi-Identifikator (mQI), wenn X ein Quasi-Identifikator ist und jede nicht-leere Teilmenge Y von X kein Quasi-Identifikator ist.*

X *ist mQI:* X *ist QI* $\wedge$ $(\nexists\ Y \subset X: (Y \neq \oslash) \wedge (Y$ *ist QI))*

Insbesondere ist X kein minimaler Quasi-Identifikator, wenn eine Teilmenge X-{A} von X mit A ∈ X existiert, die ein Quasi-Identifikator ist. Das Finden von allen Quasi-Identifikatoren stellt ein NP-vollständiges Problem dar, weil die Menge der zu untersuchenden Teilmengen exponentiell zur Anzahl der Attribute einer Relation steigt. Besteht eine Relation aus n Attributen, so existieren insgesamt 2^n Attributkombinationen, für die ermittelt werden muss, ob sie ein QI sind.

In [12] stellen Motwani und Xu einen Algorithmus zum effizienten Erkennen von minimalen Quasi-Identifikatoren vor. Dieser baut auf die von Mannila et. al [10] vorgeschlagene, ebenenweise Erzeugung von Attributmengen auf. Dabei wird die Minimalitätseigenschaft von Quasi-Identifikatoren sofort erkannt und der Suchraum beim Durchlauf auf der nächsten Ebene eingeschränkt.

Der Algorithmus ist effizienter als alle 2^n Teilmengen zu testen, allerdings stellt die von Big-Data-Anwendungen erzeugte Datenmenge eine neue Herausforderung dar. Insbesondere die hohe Dimensionalität und die Vielfalt der Daten sind ernst zu nehmende Probleme. Aus diesem Grund schlagen wir im folgenden Kapitel einen neuen Algorithmus vor, der auf den Algorithmus von Motwani und Xu aufsetzt.

2.2 Sideways Information Passing

Der von uns entwickelte Algorithmus verwendet Techniken, die bereits beim Sideways Information Passing (SIP, [4]) eingesetzt werden. Der grundlegende Ansatz von SIP besteht darin, dass während der Ausführung von Anfrageplänen Tupel nicht weiter betrachtet werden, sofern mit Sicherheit feststeht, dass sie keinen Bezug zu Tupeln aus anderen Relationen besitzen.

Durch das frühzeitige Erkennen solcher Tupel wird der zu betrachtende Suchraum eingeschränkt und die Ausführungszeit von Anfragen reduziert. Besonders effektiv ist dieses Vorgehen, wenn das Wissen über diese „magic sets" [14] zwischen den Teilen eines Anfrageplans ausgetauscht und in höheren Ebenen des Anfrageplans mit eingebunden wird. Beim SIP werden zudem weitere Techniken wie Bloomjoins [9] und Semi-Joins eingesetzt um den Anfrageplan weiter zu optimieren.

2.3 Effiziente Erfragung von identifizierenden Attributmengen

In [5] wird ein Algorithmus zur Ermittlung von identifizierenden Attributmengen (IA) in einer relationalen Datenbank beschrieben. Wird für eine Attributmenge erkannt,

dass diese eine IA für eine Relation R ist, so sind auch alle Obermengen dieser Attributmenge IA für R. Ist für eine Relation bestehend aus den Attributen A, B und C bekannt, dass B eine identifizierende Attributmenge ist, dann sind auch AB, BC und ABC eine IA der Relation.

Ist eine Attributmenge hingegen keine IA für R, so sind auch alle Teilmengen dieser Attributmenge keine IA. Wenn beispielsweise AC keine IA für R ist, dann sind auch weder A noch C identifizierende Attributmengen für R. Attributmengen, die keine identifizierende Attributmenge sind, werden als *negierte Schlüssel* bezeichnet.

Der in [5] vorgestellte Algorithmus nutzt diese Eigenschaften um anhand eines Dialoges mit dem Nutzer die Schlüsseleigenschaften einer bereits existierenden Relation festzulegen. Dabei wird dem Nutzer ein Ausschnitt der Relationstabelle präsentiert anhand derer entschieden werden soll, ob eine Attributkombination Schlüssel ist oder nicht. Wird in einer Teilrelation festgestellt, dass die Attributmenge Tupel mit gleichen Attributwerten besitzt, so kann die Attributkombination für die Teilmenge, als auch für die gesamte Relation kein Schlüssel sein.

3. ALGORITHMUS

In diesem Kapitel stellen wir einen neuen Algorithmus zum Finden von minimalen Quasi-Identifikatoren vor. Der Algorithmus beschränkt sich dabei auf die Einschränkung der zu untersuchenden Attributkombinationen. Der entwickelte Ansatz führt dabei den von [12] vorgestellten Bottom-Up-Ansatz mit einen gegenläufigen Top-Down-Verfahren zusammen.

3.1 Bottom-Up

Der von Motwani und Xu in [12] vorgestellte Ansatz zum Erkennen aller Quasi-Identifikatoren innerhalb einer Relation nutzt einen in [10] präsentierten Algorithmus. Dabei wird für eine Relation mit n Attributen ebenenweise von den einelementigen zu n-elementigen Attributkombinationen Tests durchgeführt. Wird für eine i-elementige ($1 \leq i < n$) Attributkombination AK festgestellt, dass diese ein Quasi-Identifikator ist, so werden alle Attributkombinationen in den höheren Ebenen, die AK als Teilmenge enthalten, nicht mehr berücksichtigt.

Die Methodik ist in Algorithmus 1 kurz skizziert. Zunächst erfolgt eine Initialisierung der Datenbank. Dabei wird zudem die zu untersuchende Relation einmalig überprüft, um festzustellen, wie viele Tupel vorhanden sind. Anschließend werden alle einelementigen Attributmengen gebildet. Für jede Attributmenge wird überprüft, wie viele einzigartige Tupel in der Relation vorhanden sind und das Verhältnis zur Gesamtzahl an Tupeln gebildet. Liegt der Anteil über einen vorher bestimmten Grenzwert (*threshold*), so ist diese Attributmenge ein Quasi-Identifikator und wird in die Menge aller minimalen QIs *qiLowerSet* aufgenommen.

Sind alle Attributkombinationen überprüft, werden mittels Algorithmus 2 die nächstgrößeren Attributkombinationen unter Rücksichtnahme der bekannten QI gebildet. Die Überprüfung wird solange fortgesetzt, bis alle potentiellen Attributkombinationen überprüft worden sind.

Der Algorithmus arbeitet sehr effizient, da durch das Bottom-Up-Vorgehen die Minimalität der gefundenen QI sofort festgelegt ist. Besonders gut eignet sich der Algorithmus, wenn die Relation viele, aus wenigen Attributen zusammengesetzte QIs besitzt, da so der Suchraum gleich zu Beginn stark eingeschränkt wird.

Algorithm 1: bottomUp

```
Data: database table tbl, list of attributes elements
Result: a set with all minimal QI qiLowerSet
initialization();
for element in elements do
    set := set ∪ {element}
end
while set is not empty do
    for Set testSet: set do
        double p := getPercentage(testSet, tbl);
        if p ≥ threshold then
            qiLowerSet := qiLowerSet ∪ {testSet};
        end
    end
    set := buildNewLowerSet(set, elements);
end
return qiLowerSet;
```

Algorithm 2: buildNewLowerSet

```
Data: current lower set lSet, list of attributes
      elements
Result: the new lower set lSetNew
Set lSetNew := new Set();
for Set set: lSet do
    for Attribut A: elements do
        if ∄q ∈ qiLowerSet : q ⊆ set then
            lSetNew := lSetNew ∪ {set ∪ {A}};
        end
    end
end
return lSetNew;
```

Der Nachteil des Algorithmus zeigt sich, wenn die Relation QIs besitzt, die aus vielen Attributen zusammengesetzt sind. In diesem Fall wird der Suchraum erst zum Ende eingeschränkt, wodurch die Anzahl der zu betrachtenden Attributmengen nur unmerklich geringer ist. Falls die Relation sogar keine QIs besitzt, so erkennt der Algorithmus dies erst nach Überprüfung aller Kombinationen.

3.2 Top-Down

Für die Erklärung des Algorithmus 5 wird noch das zum Bottom-Up-Vorgehen entgegengesetzte Top-Down-Vorgehen benötigt. Dieses Verfahren setzt auf die in [5] vorgeschlagenen negierten Schlüssel auf. Analog zu negierten Schlüsseln gilt, dass eine Teilmenge T kein Quasi-Identifikator ist, wenn eine Attributkombination existiert, die kein QI ist und T als Teilmenge enthält.

Die Überprüfung der Attributkombinationen erfolgt wie beim Bottom-Up-Verfahren ebenenweise, jedoch in umgekehrter Reihenfolge. Für eine Relation mit n Attributen wird zunächst die n-elementige Teilmenge gebildet und geprüft. Anschließend werden alle (n-1)-elementigen Teilmengen gebildet. Dies wird solange fortgesetzt, bis alle einelementigen Teilmengen überprüft wurden. Die gefundenen Quasi-Identifikatoren werden in *qiUpperSet* gespeichert.

Durch das Top-Down-Vorgehen ist die Minimalität der Quasi-Identifikatoren nicht gewährleistet. Dieser Nachteil

Algorithm 3: buildNewUpperSet

```
Data: current upper set uSet
Result: the new upper set uSetNew
Set uSetNew := new Set();
for Set set: uSet do
    for Attribut A: set do
        if ∄o ∈ optOutSet: set - {A} ⊆ o then
            uSetNew := uSetNew ∪ {set - {A}};
        end
    end
end
return uSetNew;
```

lässt sich dadurch beheben, dass wenn auf Ebene k ein QI gefunden wird, die entsprechenden Obermengen in den Ebenen mit mehr als k Attributen gestrichen werden. In den Algorithmen 3 und 4 ist das Top-Down-Vorgehen skizziert.

Algorithm 4: topDown

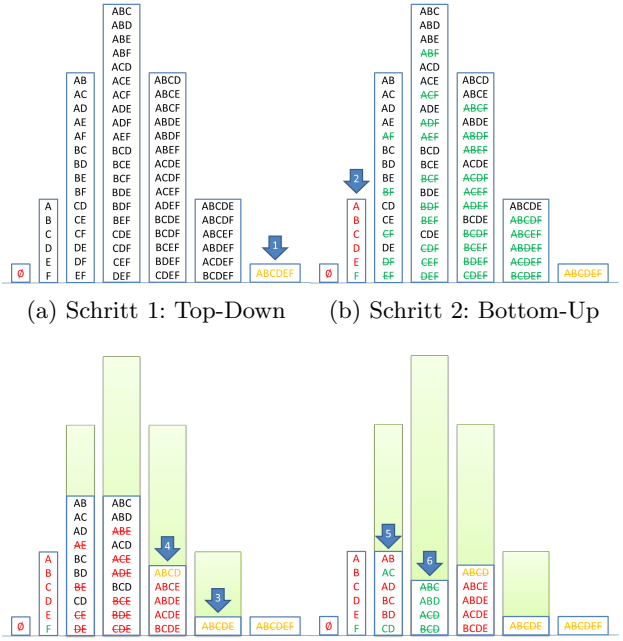
```
Data: database table tbl, list of attributes elements
Result: a set with all minimal quasi-identifier qiSet
initialization();
set := elements;
Set optOutSet := new Set();
Set qiUpperSet := new Set();
while set is not empty do
    for Set<String> testSet: set do
        double p := getPercentage(testSet, tbl);
        if p < threshold then
            optOutSet := optOutSet ∪ {subset};
        else
            qiUpperSet := qiUpperSet ∪ {testSet};
            for Set o: qiSet do
                if testSet ⊂ o then
                    qiUpperSet := qiUpperSet - {o};
                end
            end
        end
    end
    set := buildNewUpper(set);
end
return qiUpperSet;
```

Der Top-Down-Ansatz hebt die Nachteile des Bottom-Up-Vorgehens auf: der Algorithmus arbeitet effizient, wenn QIs aus vielen Attributen zusammengesetzt sind und für den Fall, dass die gesamte Relation kein QI ist, wird dies bei der ersten Überprüfung erkannt und der Algorithmus terminiert dann umgehend.

Besteht die Relation hingegen aus vielen kleinen QIs, dann wird der Suchraum erst zum Ende des Algorithmus stark eingeschränkt. Ein weiterer Nachteil liegt in der erhöhten Rechenzeit, auf die in der Evaluation näher eingegangen wird.

3.3 Bottom-Up+Top-Down

Der in diesem Artikel vorgeschlagene Algorithmus kombiniert die oben vorgestellten Verfahren. Dabei werden die Verfahren im Wechsel angewandt und das Wissen über (negierte) Quasi-Identifikatoren wie beim Sideways Information



(a) Schritt 1: Top-Down (b) Schritt 2: Bottom-Up

(c) Schritt 3+4: Top-Down (d) Schritt 5+6: Bottom-Up

Abbildung 1: Veranschaulichung des Bottom-Up+Top-Down-Algorithmus

Passing [4] untereinander ausgetauscht. Es wird pro Berechnungsschritt entweder die Top-Down- oder die Bottom-Up-Methode angewandt und das Ergebnis an die jeweils andere Methode übergeben. Der Algorithmus terminiert, sobald alle Attributebenen durch einen der beiden Methoden abgearbeitet wurden oder das Bottom-Up-Vorgehen keine Attributkombinationen mehr zu überprüfen hat. In Abbildung 1 ist die Arbeitsweise des Algorithmus anhand einer Beispielrelation mit sechs Attributen dargestellt. Die rot markierten Kombinationen stehen dabei für negierte QI, grün markierte für minimale QI und gelb markierte für potentiell minimale QI.

Um zu entscheiden, welcher Algorithmus im nächsten Zyklus angewandt wird, wird eine Wichtungsfunktion eingeführt. Die Überprüfung einer einzelnen Attributkombination auf Duplikate hat eine Laufzeit von O(n*log(n)), wobei n die Anzahl der Tupel in der Relation ist. Die Überprüfung der Tupel hängt aber auch von der Größe der Attributkombination ab. Besteht ein zu überprüfendes Tupel aus mehreren Attributen, so müssen im Datenbanksystem auch mehr Daten in den Arbeitsspeicher für die Duplikaterkennung geladen werden. Durch große Datenmengen werden Seiten schnell aus dem Arbeitsspeicher verdrängt, obwohl sie später wieder benötigt werden. Dadurch steigt die Rechenzeit weiter an.

Für eine vereinfachte Wichtungsfunktion nehmen wir an, dass alle Attribute den gleichen Speicherplatz belegen. Die Anzahl der Attribute in einer Attributkombination bezeichnen wir mit m. Für die Duplikaterkennung ergibt sich dann eine Laufzeit von O((n*m)*log(n*m)).

Da die Anzahl der Tupel für jede Duplikaterkennung konstant bleibt, kann n aus der Kostenabschätzung entfernt werden. Die Kosten für die Überprüfung einer einzelnen

Algorithm 5: bottomUpTopDown

```
Data: database table tbl, list of attributes attrList
Result: a set with all minimal quasi-identifier qiSet
attrList.removeConstantAttributes();
Set upperSet := new Set({attrList});
Set lowerSet := new Set(attrList);
// Sets to check for each algorithm
int bottom := 0;
int top := attrList.size();
while (bottom<=top) or (lowerSet is empty) do
    calculateWeights();
    if isLowerSetNext then
        bottomUp();
        buildNewLowerSet();
        bottom++;
        // Remove new QI from upper set
        modifyUpperSet();
    else
        topDown();
        buildNewUpperSet();
        top--;
        // Remove new negated QI from lower set
        modifyLowerSet();
    end
end
qiSet := qiLowerSet ∪ qiUpperSet;
return qiSet;
```

Attributkombination mit m Attributen beträgt demnach O((m*log(m)).

Die Gesamtkosten für das Überprüfen der möglichen Quasi-Identifikatoren werden mit W_{AVG} bezeichnet. W_{AVG} ergibt sich aus dem Produkt für das Überprüfen einer einzelnen Attributkombination und der Anzahl der Attributkombinationen ($AttrK_n$) mit n Attributen.

$$W_{AVG} := AttrK_n * log(m) * m \tag{2}$$

Soll die Wichtungsfunktion präziser sein, so lässt sich der Aufwand abschätzen, indem für jede Attributkombination X die Summe s über die Attributgrößen von X gebildet und anschließend gewichtet wird. Die Einzelgewichte werden anschließend zum Gesamtgewicht aufsummiert.

$$W_{AVG} := \sum_{X \in AttrK_n} log(s) * s;\ s = \sum_{A \in X} size(A) \tag{3}$$

Diese Wichtung eignet sich allerdings nur, wenn Zugang zu den Metadaten der Datenbankrelation besteht.

4. EVALUATION

Für die Evaluation des Algorithmus wurde die „Adult“-Relation aus dem UCI Machine Learning Repository [6] verwendet. Die Relation besteht aus anonymisierten, personenbezogenen Daten, bei denen Schlüssel sowie Vor- und Nachname von Personen entfernt wurden. Die übrigen 15 Attribute enthalten Angaben zu Alter, Ehestand, Staatsangehörigkeit und Schulabschluss. Die Relation besteht insgesamt aus 32561 Tupeln, die zunächst im CSV-Format vorlagen und in eine Datenbank geparst wurden.

Die Evaluation erfolgte in einer Client-Server-Umgebung. Als Server dient eine virtuelle Maschine, die mit einer 64-Bit-CPU (vier Kerne @ 2 GHz und jeweils 4 MB Cache) und 4 GB Arbeitsspeicher ausgestattet ist. Auf dieser wurde eine MySQL-Datenbank mit InnoDB als Speichersystem verwendet. Der Client wurde mit einem i7-3630QM als CPU betrieben. Dieser bestand ebenfalls aus vier Kernen, die jeweils über 2,3 GHz und 6 MB Cache verfügten. Als Arbeitsspeicher standen 8 GB zur Verfügung. Als Laufzeitumgebung wurde Java SE 8u5 eingesetzt.

Der Datensatz wurde mit jedem Algorithmus getestet. Um zu ermitteln, wie die Algorithmen sich bei verschiedenen Grenzwerten für Quasi-Identifikatoren verhalten, wurden die Tests mit 10 Grenzwerten zwischen 50% und 99% wiederholt.

Die Tests mit den Top-Down- und Bottom-Up-Algorithmen benötigten im Schnitt gleich viele Tablescans (siehe Abbildung 2). Die Top-Down-Methode lieferte bessere Ergebnisse bei hohen QI-Grenzwerten, Bottom-Up ist besser bei niedrigeren Grenzwerten. Bei der Laufzeit (siehe Abbildung 3) liegt die Bottom-Up-Methode deutlich vor dem Top-Down-Ansatz. Grund hierfür sind die großen Attributkombinationen, die der Top-Down-Algorithmus zu Beginn überprüfen muss.

Der Bottom-Up+Top-Down-Ansatz liegt hinsichtlich Laufzeit als auch bei der Anzahl der Attributvergleiche deutlich vorne. Die Anzahl der Tablescans konnte im Vergleich zum Bottom-Up-Verfahren zwischen 67,4% (4076 statt 12501 Scans; Grenzwert: 0.5) und 96,8% (543 statt 16818 Scans; Grenzwert 0.9) reduziert werden. Gleiches gilt für die Laufzeit (58,1% bis 97,5%; siehe Abbildung 3).

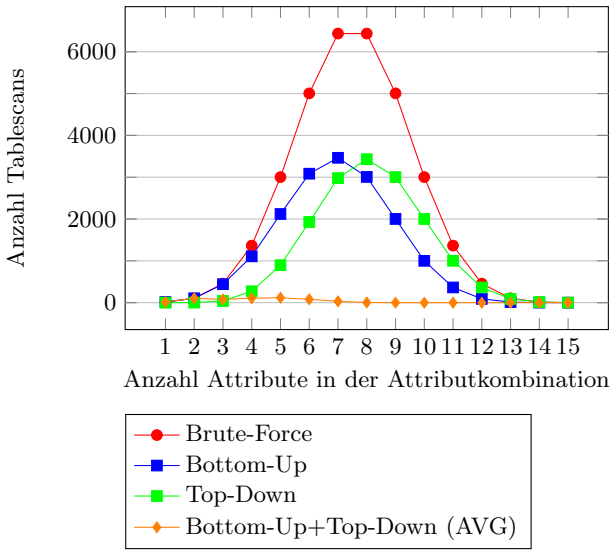


Abbildung 2: Verhältnis von der Anzahl der Attribute in den Attributkombinationen zur Anzahl von Tablescans (Adult-DB, Grenzwert 90%)

Wie in Abbildung 3 zu erkennen ist, nimmt die Laufzeit beim Bottom-Up+Top-Down-Verfahren im Grenzwertbereich von 70%-90% stark ab. Interessant ist dies aus zwei Gründen. Erstens nimmt die Anzahl der Quasi-Identifikatoren bis 90% ebenfalls ab (179 bei 50%, 56 bei 90%). Dies legt nahe, dass die Skalierung des Verfahrens neben der Dimension der Relation (Anzahl von Tupel und

Attributen) auch von der Anzahl der vorhandenen QIs abhängt. Um den Zusammenhang zu bestätigen, sind aber weitere Untersuchungen erforderlich.

Zweitens wird dieser Grenzwertbereich in der Literatur [13] häufig benutzt, um besonders schützenswerte Daten hervorzuheben. Durch die gute Skalierung des Algorithmus in diesem Bereich lassen sich diese QIs schnell feststellen.

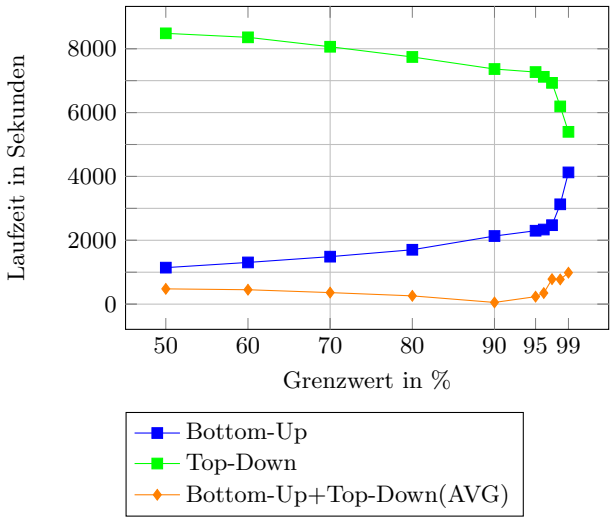


Abbildung 3: Vergleich der Laufzeit der verschiedenen Algorithmen (Adult-DB)

5. AUSBLICK

In dieser Arbeit stellten wir einen effizienten Algorithmus zur Erkennung von QI in hochdimensionalen Daten vor. Anhand eines Beispiels mit Sensordaten zeigten wir die Eignung in Assistenzsystemen. Darüber hinaus ermitteln wir derzeit, inwiefern sich QIs in temporalen Datenbanken feststellen lassen. Das so gewonnene Wissen über schützenswerte Daten wird in unser Gesamtprojekt zur datenschutzfreundlichen Anfrageverarbeitung in Assistenzsystemen eingebunden.

In späteren Untersuchungen werden wir testen, welche weiteren Quasi-Identifikatoren sich aus der Kombination von Daten verschiedener Relationen ableiten lassen. Der dafür verwendete Datensatz besteht aus Sensordaten, die im Smart Appliance Lab des Graduiertenkollegs MuSAMA durch ein Tool [11] aufgezeichnet wurden. Die Daten umfassen dabei Bewegungsprofile, die mittels RFID-Tags und einen Sensfloor [15] erfasst wurden, aber auch Informationen zu Licht und Temperatur. Eine Verknüpfung der Basis-Relationen erfolgt dabei über die ermittelten Quasi-Identifikatoren.

6. DANKSAGUNG

Hannes Grunert wird durch die Deutsche Forschungsgemeinschaft (DFG) im Rahmen des Graduiertenkollegs 1424 (Multimodal Smart Appliance Ensembles for Mobile Applications - MuSAMA) gefördert. Wir danken den anonymen Gutachtern für ihre Anregungen und Kommentare.

7. LITERATUR

[1] Bundesrepublik Deutschland. Bundesdatenschutzgesetz in der Fassung der Bekanntmachung vom 14. Januar 2003, das zuletzt durch Artikel 1 des Gesetzes vom 14. August 2009 geändert worden ist, 2010.

[2] T. Dalenius. Finding a Needle In a Haystack or Identifying Anonymous Census Records. *Journal of Official Statistics*, 2(3):329–336, 1986.

[3] H. Grunert. Privacy Policy for Smart Environments. `http://www.ls-dbis.de/pp4se`, 2014. zuletzt aufgerufen am 17.07.2014.

[4] Z. G. Ives and N. E. Taylor. Sideways information passing for push-style query processing. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 774–783. IEEE, 2008.

[5] M. Klettke. *Akquisition von Integritätsbedingungen in Datenbanken*. PhD thesis, Universität Rostock, 1997.

[6] R. Kohavi and B. Becker. Adult Data Set. `http://archive.ics.uci.edu/ml/datasets/Adult`, 1996. zuletzt aufgerufen am 17.07.2014.

[7] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *ICDE*, volume 7, pages 106–115, 2007.

[8] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[9] L. F. Mackert. R* optimizer validation and performance evaluation for distributed queries. In *Readings in database systems*, pages 219–229. Morgan Kaufmann Publishers Inc., 1988.

[10] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[11] D. Moos. Konzepte und Lösungen für Datenaufzeichnungen in heterogenen dynamischen Umgebungen. Bachelorarbeit, Universität Rostock, 2011.

[12] R. Motwani and Y. Xu. Efficient algorithms for masking and finding quasi-identifiers. In *Proceedings of the Conference on Very Large Data Bases (VLDB)*, pages 83–93, 2007.

[13] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, Technical report, SRI International, 1998.

[14] P. Seshadri, J. M. Hellerstein, H. Pirahesh, T. Leung, R. Ramakrishnan, D. Srivastava, P. J. Stuckey, and S. Sudarshan. Cost-based optimization for magic: Algebra and implementation. In *ACM SIGMOD Record*, volume 25, pages 435–446. ACM, 1996.

[15] A. Steinhage and C. Lauterbach. Sensfloor (r): Ein AAL Sensorsystem für Sicherheit, Homecare und Komfort. *Ambient Assisted Living-AAL*, 2008.

[16] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[17] M. Weiser. The computer for the 21st century. *Scientific american*, 265(3):94–104, 1991.