

Beschreibung der fachlichen Lösung

- Wir stellen für jeden Artikel x eine Assoziationsregel $\{x\} \rightarrow \{y\}$ auf, wobei $y \in (\text{Artikel} \setminus \{x\})$.
- Weiterhin bezeichnen wir die Menge aller Transaktionen ("Einkäufe") als D . Eine Transaktion ist dabei eine nicht-leere Menge von Artikeln. Damit gilt: $D \subseteq (P(\text{Artikel}) \setminus \{\})$, wobei D eine Multimenge ist. Weiterhin gilt: $\forall t \in D : t \in (P(\text{Artikel}) \setminus \{\})$.

- Zum Beispiel:

$$\begin{aligned}\text{Artikel} &= \{A1, A2, A3\} \\ P(\text{Artikel}) &= \{\{\}, \{A1\}, \{A2\}, \{A3\}, \{A1, A2\}, \\ &\quad \{A1, A3\}, \{A2, A3\}, \{A1, A2, A3\}\} \\ D &= \{\{A1\}, \{A1, A2\}, \{A1, A2\}, \{A2, A3\}\}\end{aligned}$$

- Der Parameter Konfidenz der Assoziationsregel $\{x\} \rightarrow \{y\}$ berechnet sich nun wie folgt:

$$\frac{|\{t \in D : \{x\} \cup \{y\} \subseteq t\}|}{|\{u \in D : \{x\} \subseteq u\}|} = \frac{|\{t \in D : \{x, y\} \subseteq t\}|}{|\{u \in D : \{x\} \subseteq u\}|}$$

In Worten:

$$\frac{\text{Anzahl aller Transaktionen die x und y enthalten}}{\text{Anzahl aller Transaktionen die x enthalten}}$$

- Der Parameter Support der Assoziationsregel $\{x\} \rightarrow \{y\}$ berechnet sich nun wie folgt:

$$\frac{|\{t \in D : \{x\} \cup \{y\} \subseteq t\}|}{|D|} = \frac{|\{t \in D : \{x, y\} \subseteq t\}|}{|D|}$$

In Worten:

$$\frac{\text{Anzahl aller Transaktionen die x und y enthalten}}{\text{Anzahl aller Transaktionen}}$$

Beschreibung der Implementierung

- **Funktion 'get_association_analysis_data':**

Um die Assoziationsanalyse auch auf Assoziationsregeln $\{x\} \rightarrow \{y\}$ beliebiger Teilmengen $x, y \subseteq D$ durchführen zu können, definieren wir uns eine Funktion `get_association_analysis_data(articles_antecedent, articles_consequent)`, wobei `articles-antecedent` $\doteq x$ und `articles-consequent` $\doteq y$.

Die Funktion liefert als Rückgabewert folgende Werte:

1. Anzahl der Transaktionen die x enthalten
2. Anzahl der Transaktionen die y enthalten
3. Anzahl der Transaktionen die x und y enthalten
4. Parameter Konfidenz
5. Parameter Support.

Der Begriff Transaktion ist dabei bei unserem Datenmodell synonym zur Rechnung.

Die Funktion ist wie folgt implementiert, wobei auf eine Hilfsfunktion

`get_bill_count` zurückgegriffen wird, die im nächsten Abschnitt beschrieben wird:

1. Bestimme **Anzahl aller Transaktionen**:
`count_all = Bill.count`
2. Bestimme **Anzahl aller Transaktionen die x enthalten**:
`count_x = get_bill_count(articles_antecedent)`
3. Bestimme **Anzahl aller Transaktionen die y enthalten**:
`count_y = get_bill_count(articles_consequent)`
4. Bestimme **Anzahl aller Transaktionen die x und y enthalten**:
`count_xy = get_bill_count(articles_antecedent | articles_consequent)`
5. Bestimme Konfidenz:
`confidence = count_xy / count_x`
6. Bestimme Support:
`support = count_xy / count_all`
7. Prüfe, ob bei der Berechnung der Parameter Konfidenz und Support eine Division durch 0 auftritt und liefere in diesem Fall als Ergebnis den Wert 0.0.
8. Liefere die Ergebnisse zurück:
`return [count_x, count_y, count_xy, confidence, support].`

- **Funktion 'get_bill_count':**

Hilfsfunktion mit einem Parameter `articles`, welcher eine Menge von Artikeln darstellt. Die Funktion berechnet die Anzahl aller Transaktionen, die alle Artikel aus `articles` enthalten.

Die einzelnen Transaktionen stellen wir aus Übersichtlichkeitsgründen nicht weiter dar. Wir haben jedoch zur Erweiterbarkeit eine Funktion `get_bills(articles)` definiert, die uns die entsprechenden Transaktionen (Rechnungen) als Array zurückliefert. `get_bill_count(articles)` greift dabei auf diese Funktion zurück und liefert die Anzahl des zurückgelieferten Arrays zurück:

```
return get_bills(articles).count
```

Die Funktion `get_bills(articles)` ist wie folgt implementiert:

1. Iteriere über jedes Element aus `articles` und speichere für jedes Element alle

Rechnungen die das jeweilige Element enthalten in Array. Speichere diese Arrays nun in ein übergeordnetes Array:

```
article_bills = articles.map{|article|  
  BillEntry.where(:article => article).map{|entry|  
    entry.bill}.uniq}
```

2. Iteriere über das übergeordnete Array und bilde dabei von jedem Element den Durchschnitt:

```
result = article_bills.reduce([])  
  {|accu, entry| accu & entry}
```

3. Liefere das Ergebnis zurück:

```
return result
```

- Wir stellen die numerische Ergebnisse unserer Analyse in der Anzeige eines Artikels $artikel_x$ dar, sofern ein Administrator eingeloggt ist.

In der Anzeige bestimmen wir für jeden anderen Artikel (hier $artikel_y$) die Analysedaten der Assoziationsregel $\{artikel_x\} \rightarrow \{artikel_y\}$ und geben sowohl die Bezeichnung von $artikel_y$ als auch die Parameter Support und Konfidenz aus.

Die Ergebnisse werden zuerst nach Konfidenz und danach nach Support sortiert. Es werden dabei nur solche $artikel_y$ angezeigt, bei denen die Parameter Support und Konfidenz größer als 0 sind.

Weiterhin wird in absoluten Zahlen angegeben, wieviele Transaktionen insgesamt getätigt wurden und in wievieler dieser Transaktionen $artikel_x$ enthalten war.

- Besucht ein Gast oder ein angemeldeter Benutzer, der kein Administrator ist, die Seite, werden ausschließlich die Bilder der Artikel angezeigt, die sich aus der Assoziationsregel $\{artikel_x\} \rightarrow \{artikel_y\}$ ergeben haben.

Die Ergebnisse werden zuerst nach Konfidenz und danach nach Support sortiert. Es werden dabei nur solche $artikel_y$ angezeigt, bei denen die Parameter Support und Konfidenz größer als 0 sind.