Generierung von synthetischen Banktransaktionsdaten

Bachelorarbeit im Fach Informatik

vorgelegt

von

Samuel Galliker

Ballwil, Luzern, Schweiz

Matrikelnummer 05-703-814

Angefertigt am

Institut für Informatik

der Universität Zürich

Prof. Dr. A. Bernstein

Betreuer: Jonas Luell

Abgabe der Arbeit: 22.09.2008

Inhaltsverzeichnis

1. Einleitung

Die vorliegende Arbeit entstand unter der Betreuung von Jonas Luell, der anlässlich seiner Dissertation Werkzeuge erarbeitet, die Fahndern in Banken auf der Suche nach internem Betrug und Geldwäscherei behilflich sind [Bernstein und Luell 2008]. Für seine Publikation ist Jonas Luell auf Transaktionsdaten angewiesen, die auf der einen Seite datenschutztechnisch unbedenklich sind und auf der anderen Seite die gleichen Eigenschaften wie die Originaldaten aufweisen. Das Ziel dieser Bachelorarbeit ist es, synthetische Banktransaktionsdaten mit realen Verteilungskennzahlen zu generieren, die zu diesem Zweck eingesetzt werden können.

Um die Werkzeuge zur Betrugserkennung trainieren und testen zu können, werden Daten benötigt, von denen bekannt ist, welcher Teil davon Betrüge darstellt. Die realen Daten eignen sich nicht dafür, da in diesen die Betrugsfälle nicht bekannt sind. Deshalb bietet sich der Einsatz von synthetischen Daten an, in die Betrugsfälle kontrolliert injiziert werden können.

In [Lundin *et al.* 2002] wird die von den Autoren vorgeschlagene Methodik zur Generierung von synthetischen Daten mit Betrugsfällen erläutert und in [Lundin *et al.* 2003] am Beispiel eines Video-on-Demand-Dienstes getestet. Dabei wird zwischen Betrugs- und Hintergrunddaten unterschieden. Erstere sind im Kontext der Banktransaktionen jene Daten, welche die Fälle von internem Betrug und von Geldwäscherei beinhalten. Alle ordnungsgemäss initiierten Transaktionen stellen die Hintergrunddaten dar. Das in der vorliegenden Arbeit verfolgte Ziel ist die Generierung solcher synthetischer Hintergrunddaten.

Der Weg zu diesem Ziel ist in zwei Etappen aufgeteilt: Der Transaction Evaluator analysiert die Originaldaten, bevor der Transaction Builder anhand der ermittelten Eigenschaften die synthetischen Daten generiert. Nachdem in Kapitel 2 etwas genauer auf diesen Prozess eingegangen wurde, werden in Kapitel 3 anhand von zwei Testmengen die Stärken und Schwächen des erstellten Java-Programms ermittelt.

2. Software

Als Erstes stellte sich die Frage, wie vorgegangen werden soll, um synthetische Banktransaktionsdaten zu erhalten, die möglichst nahe an der Realität angelehnt sind.

Ich prüfte anhand von SUBDUE [www.ailab.wsu.edu/subdue], ob Graphen ein geeignetes Mittel darstellen, um die Welt der Transaktionen darzustellen. Es stellte sich heraus, dass die von SUBDUE zur Verfügung gestellten Werkzeuge für den vorgesehenen Einsatz zu wenig mächtig sind. Es ist nicht möglich, zwei Knoten (Bankkunden) mit mehreren Kanten (Transaktionen) zu verbinden. Diese Idee wurde deshalb nicht weiter verfolgt.

Stattdessen habe ich den von meinem Betreuer vorgeschlagenen Weg über die Bildung von Clusters [Witten und Frank 2005] aufgenommen. Ich habe mir überlegt, welche Informationen über die Originaldaten benötigt werden, um realitätsnahe, synthetische Daten generieren zu können.

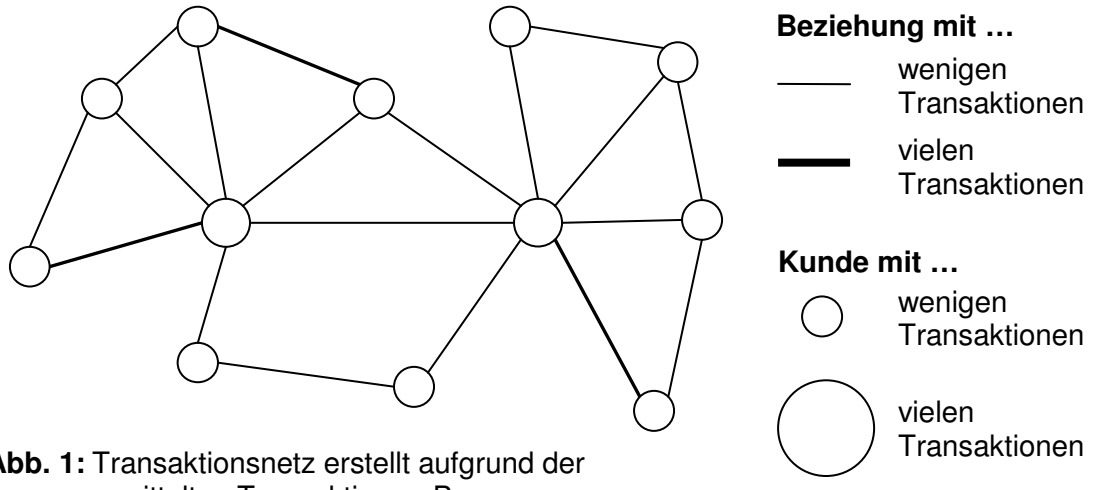

Abb. 1: Transaktionsnetz erstellt aufgrund der ermittelten Transaktionen-Bags

Zuerst wird die Höhe der Transaktionen evaluiert. Ich habe mich entschieden, die Transaktionen in so genannte Bags einzuteilen. Dazu werden zuerst die Abschnitte (z. B. Bag 1: CHF 1.00 – CHF 100.00, Bag 2: CHF 100.00 –

CHF 300.00 usw.) definiert, in welche danach die Originaltransaktionen eingeteilt werden. Die damit gewonnenen Informationen erlauben es, synthetische Transaktionen zu generieren, deren Beträge ungefähr wie in den Originaldaten verteilt sind. Dabei wird jedoch ausser Acht gelassen, dass nicht alle Bankkunden gleich viele Transaktionen ausführen. In Abbildung 1 ist ein für diese Situation charakteristisches Transaktionsnetz zu sehen (alle Knoten gleich gross).

Um der Realität näher zu kommen, müssen wir einen Schritt weitergehen und die in Bags eingeteilten Transaktionen den verschiedenen Kunden zuordnen. Damit bessere Resultate erzielt werden, ist zwischen aus- und eingehenden Transaktionen zu unterscheiden. Wir erhalten also schliesslich für jeden Kunden die Anzahl Transaktionen nach ein- und ausgehenden Transaktionen und Bags aufgeteilt (Tabelle 1).

	Ausgehende Transaktionen in …			Eingehende Transaktionen in …		
Kd.-Nr.	Bag 1	Bag 2	…	Bag 1	Bag 2	…
1	11	2	…	3	0	…
2	1	0	…	8	11	…
3	0	10	…	0	20	…
4	10	3	…	1	0	…
…	…	…	…	…	…	…

Kd.-Nr. = Kundennummer

Tab. 1: Anzahl Transaktionen pro Kunde und Bag

Anhand der Informationen in Tabelle 1 ist es möglich, synthetische Transaktionsdaten zu generieren, die berücksichtigen, dass nicht alle Kunden an gleich vielen Transaktionen beteiligt sind. In Abbildung 2 ist ein Transaktionsnetz zu sehen, das auf diese Weise entstehen könnte. Es ist deutlich zu erkennen, dass es Kunden mit vielen und solche mit wenigen Transaktionen gibt. Beim Vergleich mit der finalen Version des Transaktionsnetzes in Abbildung 3 fällt

jedoch auf, dass die Transaktionsbeziehungen zwischen den Kunden zufällig zustande kommen.

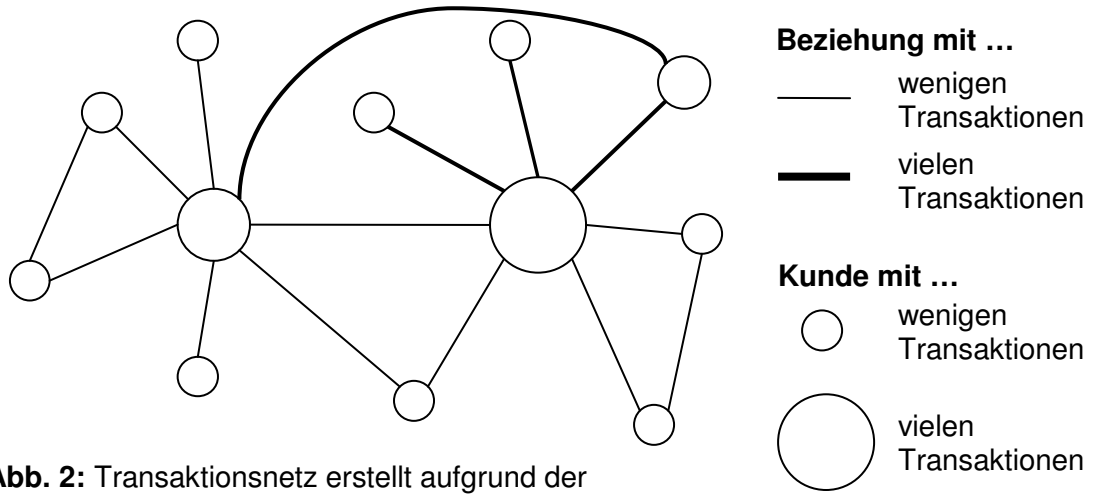

Abb. 2: Transaktionsnetz erstellt aufgrund der ermittelten Kundenverhaltens-Clusters

Um auch diesen Faktor steuern zu können, wird ein Konzept eingeführt, das in der Folge Kundenbeziehung genannt wird. Es wird gemessen, wie viele Transaktionen aus jedem Bag zwischen zwei Kunden in die eine und in die andere Richtung fliessen (Tabelle 2).

A	B	Cl.	Transaktion von A nach B in …			Transaktion von B nach A in …		
			Bag 1	Bag 2	…	Bag 1	Bag 2	…
1	2	1	2	1	…	3	0	…
1	5	2	2	0	…	0	2	…
2	4	2	1	0	…	1	2	…
3	6	3	4	2	…	0	1	…
…	…	…	…	…	…	…	…	…

A = Nummer Kunde A, B = Nummer Kunde B, Cl = Kundenbeziehungs-Cluster

Tab. 2: Anzahl Transaktionen pro Kundenbeziehung und Bag

Jene Kundenbeziehungen, die eine ähnliche Struktur der Transaktionen aufweisen, können zu Clusters zusammengefasst werden (z. B. Kundenbe-

ziehung 1/5 und Kundenbeziehung 2/4 in Tabelle 2). Gespeichert werden pro Cluster jeweils der Mittelwert und die Standardabweichung jedes Bags.

Kd.-Nr.	Cl.	KB 1	KB 2	KB 3	KB 4	KB 5	…
1	1	1	3	4	2	6	…
2	2	2	0	0	7	2	…
3	1	1	2	5	2	5	…
4	3	6	2	0	0	1	…
…	…	…	…	…	…	…	…

Kd.-Nr. = Kundennummer, Cl = Kundenverhaltens-Cluster, KB = Kundenbeziehung

Tab. 3: Anzahl Kundenbeziehung pro Kunde und Kundenbeziehungs-Cluster

Im zweiten Schritt wird für jeden Kunden evaluiert, wie viele Kundenbeziehungen er pro Kundenbeziehungs-Cluster unterhält. Wiederum werden Clusters gebildet: Dieses Mal mit Kunden, die ähnliche Kundenbeziehungen aufweisen (Tabelle 3).

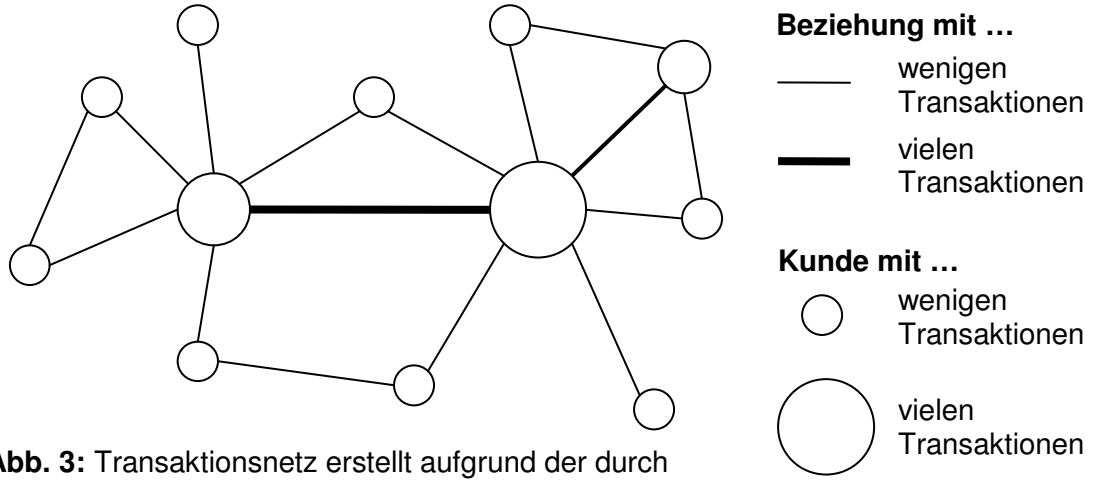

Abb. 3: Transaktionsnetz erstellt aufgrund der durch zweifaches Clustering generierten Daten

Durch die mittels dieses zweifachen Clusterings ermittelten Daten ist es möglich, Transaktionen zu generieren, die den Originaldaten sehr nahe kommen. Das Transaktionsnetz in Abbildung 3, das mit Daten gefüttert wurde,

die den beschriebenen Prozess durchlaufen haben, zeigt, dass nun auch die Kundenbeziehungen nicht mehr auf Zufall beruhen. Somit ist ein Werkzeug entstanden, mit dessen Hilfe es möglich ist, die Strukturen der originalen auf die synthetischen Daten zu übertragen.

In den folgenden beiden Abschnitten wird etwas genauer darauf eingegangen, wie der zuletzt beschriebene Prozess abläuft. Das Java-Programm ist in zwei Komponenten unterteilt: Während die Originaldaten durch den Transaction Evaluator auf deren Strukturen analysiert werden, hat der Transaction Builder die Aufgabe, aufgrund der ermittelten Verteilungen synthetische Daten zu generieren.

2.1 Transaction Evaluator

Die Aufgabe des Transaction Evaluators ist es, die originalen Transaktionsdaten zu gruppieren und die Kennzahlen der gebildeten Gruppen abzuspeichern, so dass daraus später vom Transaction Builder synthetische Transaktionen generiert werden können.

Das Programm ist in zwei Hauptmethoden (Abb. 4) unterteilt. Im ersten Teil (pairwiseBehaviourClustering) werden die Transaktionen nach ihren Beträgen in Bags eingeteilt und für jede Kundenbeziehung wird überprüft, wie viele Transaktionen aus jedem Bag in die eine und in die andere Richtung fliessen. Zum Abschluss des ersten Schrittes werden die zuvor ermittelten Kundenbeziehungen in Clusters eingeteilt; Kundenbeziehungen mit ähnlichem Transaktionsverhalten werden zusammengefasst. In der zweiten Hauptmethode (customerBehaviourClustering) wird für jeden Kunden ermittelt, an wie vielen Kundenbeziehungen aus jedem Cluster er als Auftraggeber (originator[1]) und als

[1] In dieser Arbeit gilt in einer Kundenbeziehung jener Kunde als „originator“, welcher mehr aus- als eingehende Transaktionen zu verzeichnen hat. Dementsprechend wird der andere Kunde als „beneficiary“ bezeichnet. Falls in beide Richtungen gleich viele Transaktionen fliessen, wird der Kunde mit der kleineren ID zum „originator“.

Begünstigter (beneficiary) beteiligt ist. Schliesslich werden auch die Kunden nach ihren Kundenbeziehungen in Clusters eingeteilt.

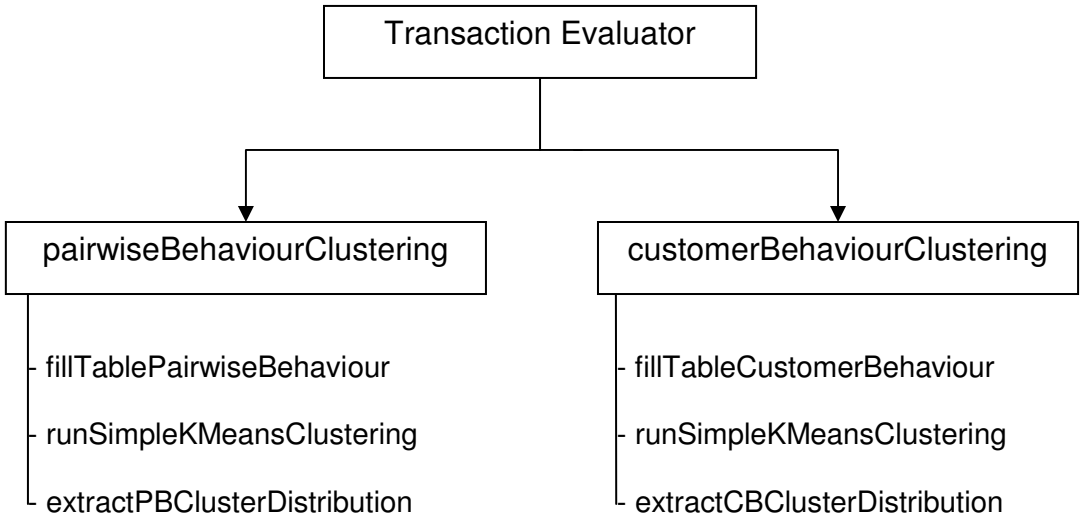

Abb. 4: Aufbau des Transaction Evaluators: Die wichtigsten Methoden

In den Kapiteln 2.1.1 und 2.1.2 wird etwas näher auf den Ablauf des Programms eingegangen. Zuerst wird die Methode pairwiseBehaviour-Clustering, in der auch das Einteilen der Transaktionen in Bags stattfindet, besprochen. Danach wird die andere Hauptmethode customerBehaviour-Clustering etwas genauer unter die Lupe genommen.

2.1.1 pairwiseBehaviourClustering

Nachdem in der Datenbank die Tabelle PAIRWISE_BEHAVIOUR (Tabelle 4) erstellt worden ist, geht es darum, diese mit den Originaldaten zu füllen. Dies geschieht in der Methode fillTablePairwiseBehaviour: Für jeden Kunden werden die Transaktionspartner, die eine grössere ID besitzen[2], ermittelt und es wird für jede Kundenbeziehung überprüft, wie viele Transaktionen aus jedem Bag in die

[2] Es werden nur grössere IDs berücksichtigt, da so beim Durchlauf durch alle Kunden jede Kundenbeziehung nur ein Mal evaluiert wird.

eine und in die andere Richtung fliessen. In die Datenbank wird jener Transaktionspartner als ORIGINATOR eingefügt, welcher in der betrachteten Kundenbeziehung mehr aus- als eingehende Transaktionen zu verzeichnen hat. Die TRXOUT stellen die vom ORIGINATOR ausgehenden, die TRXIN die beim ORIGINATOR eingehenden Transaktionen dar (Tabelle 4).

Um dieses Konzept etwas zu veranschaulichen, hier ein kurzes Beispiel: In der ersten Zeile der Tabelle 4 wird die Beziehung zwischen den Kunden 1 und 2 unter die Lupe genommen. Da ermittelt wurde, dass mehr Transaktionen von Kunde 1 zu Kunde 2 fliessen als umgekehrt, ist Kunde 1 als ORIGINATOR eingetragen. Der Wert 2 in der Spalte TRXOUT0 bedeutet, dass zwei Transaktionen von Kunde 1 (ORIGINATOR) zu Kunde 2 (BENEFICIARY) geflossen sind, die dem Bag 0 angehören.

O	B	CL	TRXOUT0	TRXOUT1	…	TRXIN0	TRXIN1	…
1	2	0	2	2	…	3	0	…
5	1	1	2	1	…	0	2	…
2	4	1	2	1	…	0	1	…
6	3	2	4	2	…	0	1	…
…	…	…	…	…	…	…	…	…

O = ORIGINATOR, B = BENEFICIARY, CL = CLUSTER (Kundenbeziehungs-Cluster)

Tab. 4: Datenbanktabelle PAIRWISE_BEHAVIOUR

Aufgrund der nun in der Datenbank eingetragenen Werte wird das Clustering durchgeführt. Es wird der „simple k-Means Clusterer“ aus der WEKA-Bibliothek[3] eingesetzt, da dieser die Möglichkeit bietet, die Anzahl der zu bildenden Clusters vor dem Start zu bestimmen. In der Methode runSimpleKMeans-Clustering werden die Kundenbeziehung also in eine vorgegebene Anzahl von

[3] Bibliothek mit verschiedenen Funktionen im Bereich des Data Mining; weitere Informationen in [Witten und Frank 2005]

Gruppen eingeteilt. Das Ziel ist es, Kundenbeziehungen mit ähnlichen Strukturen zusammenzufassen und so Kundenbeziehungs-Arten zu bilden.

In der letzten unter pairwiseBehaviourClustering aufgeführten Methode (extractPBClusterDistribution) wird mithilfe einer von der WEKA-Bibliothek zur Verfügung gestellten Funktion für jeden Cluster bzw. für jede Kundenbeziehungs-Art der Mittelwert und die Standardabweichung aller Transaktions-Bags ermittelt.

CL	TRXOUT0	TRXOUT1	…	TRXIN0	TRXIN1	…
0	2	2	…	3	0	…
1	2	1	…	0	1.5	…
2	4	2	…	0	1	…
…	…	…	…	…	…	…

CL = CLUSTER_ID (Kundenbeziehungs-Cluster)

Tab. 5: Datenbanktabelle PB_CLUSTER_MEANS

CL	TRXOUT0	TRXOUT1	…	TRXIN0	TRXIN1	…
0	0	0	…	0	0	…
1	0	0	…	0	0.707	…
2	0	0	…	0	0	…
…	…	…	…	…	…	…

CL = CLUSTER_ID (Kundenbeziehungs-Cluster)

Tab. 6: Datenbanktabelle PB_CLUSTER_DEVS

Nachdem die Daten in Tabelle 4 in Clusters eingeteilt worden sind, können die Mittelwerte (means; Tabelle 5) und Standardabweichungen (standard deviations; Tabelle 6) ermittelt werden. Da nur eine Kundenbeziehung in Cluster 0 eingeteilt wurde, entsprechen die Werte in der Tabelle PB_CLUSTER_MEANS gerade jenen der einen Kundenbeziehung und die Standardabweichung ist überall 0. Das Gleiche gilt für den Cluster 2. Der Cluster 1 ist die Zusammenfassung von zwei Kundenbeziehungen. Für TRXIN1

sind verschiedene Werte vorhanden und es werden der Mittelwert[4] 1.5 und die Standardabweichung[5] 0.707 berechnet.

Die hier ermittelten Mittelwerte und Standardabweichungen wird der Builder später verwenden, um synthetische Kundenbeziehungen zu generieren. Die Tabellen PB_CLUSTER_MEANS und PB_CLUSTER_DEVS werden also benötigt, um die Transaktionsdaten rekonstruieren zu können.

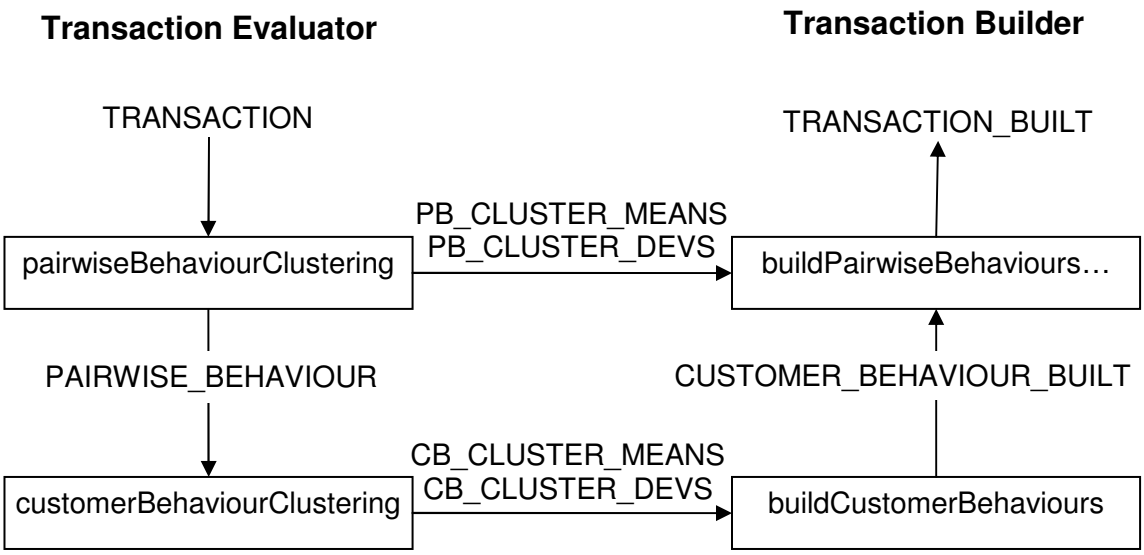

Abb. 5: Transaction Evaluator und Transaction Builder (Datenfluss)

4 $\text{Mittelwert} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{2+1}{2} = 1.5$; aus [Wikipedia: Mittelwert]

5 $\text{Standardabweichung} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{\sqrt{(2-1.5)^2 + (1-1.5)^2}}{2-1} = 0.707$

aus [Wikipedia: Standardabweichung]

2.1.2 customerBehaviourClustering

Während die Informationen über die verschiedenen Verteilungen für den Transaction Builder bestimmt sind, werden die Daten in der Tabelle PAIRWISE_BEHAVIOUR in der Methode customerBehaviourClustering verwendet (Abbildung 5). Das Ziel dieses zweiten Schrittes des Transaction Evaluators ist es, Kunden mit ähnlichem Verhalten zusammenzufassen, also Kundenverhaltens-Arten zu definieren.

Das Vorgehen in der Methode customerBehaviourClustering ist jenem in der Methode pairwiseBehaviourClustering sehr ähnlich (Abbildung 4). Es werden teilweise dieselben Methoden verwendet. In einigen Fällen ist dies jedoch nicht möglich, da die Abläufe voneinander abweichen. Auf diese Unterschiede wird in der Folge kurz eingegangen.

Auch in diesem zweiten Schritt wird die Tabelle nach deren Erstellung mit Daten gefüllt. Es geht jedoch nicht mehr darum, die Anzahl Transaktionen einer Kundenbeziehung in jedem Bag zu ermitteln, sondern es wird für jeden Kunden überprüft, an wie vielen Kundenbeziehungen er als ORIGINATOR und als BENEFICIARY beteiligt ist (Tabelle 7).

ID	CL	PB_ORI0	PB_ORI1	…	PB_BEN0	PB_BEN1	…
1	0	1	0	…	0	1	…
2	1	0	1	…	1	0	…
…	…	…	…	…	…	…	…

ID (Kundennummer), CL = CLUSTER (Kundenverhaltens-Cluster)

Tab. 7: Datenbanktabelle CUSTOMER_BEHAVIOUR

Für die Bildung von Clusters wird dieselbe Methode verwendet wie im ersten Schritt. In runSimpleKMeansClustering werden die in der Datenbank abgespeicherten Kundenverhalten in Clusters eingeteilt. Schliesslich werden mit der Methode extractCBClusterDistribution die Mittelwerte und Standardabweichungen der verschiedenen Kundenverhaltens-Clusters evaluiert und in die Daten-

bank geschrieben, um später vom Transaction Builder für die Generierung von Kundenverhaltens-Daten verwendet werden zu können (Abbildung 5). Für diesen Vorgang wird nicht dieselbe Methode wie beim pairwiseBehaviour-Clustering verwendet, da in diesem zweiten Schritt zusätzlich zu den Verteilungskennzahlen die Anzahl Kunden abgespeichert werden, die einem Kundenverhaltens-Cluster zugeordnet werden (Tabelle 8).

CL	NU	PB_ORI0	PB_ORI1	…	PB_BEN0	PB_BEN1	…
0	1	1	0	…	0	1	…
1	1	0	1	…	1	0	…
…	…	…	…	…	…	…	…

CL = CLUSTER_ID (Kundenverhaltens-Cluster), NU = NUMBEROFAPP (Anzahl Kunden im Cluster)

Tab. 8: Datenbanktabelle CB_CLUSTER_MEANS

Damit hat der Transaction Evaluator seine Aufgabe erfüllt, denn die Daten über die Verteilungen, die vom Transaction Builder benötigt werden (Abbildung 5), sind nun in der Datenbank abgelegt. In Kapitel 2.2 wird der Ablauf der Generierung der synthetischen Banktransaktionsdaten besprochen.

2.2 Transaction Builder

Nachdem in Kapitel 2.1 auf die Evaluation der Verteilungen in den Originaldaten eingegangen wurde, wird nun in diesem Kapitel aufgezeigt, wie daraus im Transaction Builder synthetische Transaktionen gebildet werden. Dabei wird der Prozess wieder in zwei Schritte unterteilt (Abbildung 6): Während in der Methode buildCustomerBehaviours die künstlichen Kundenverhalten entstehen, hat die Methode buildPairwiseBehaviourAndTransactions die Aufgabe, daraus über die Bildung von Kundenbeziehungen synthetische Transaktionen zu generieren.

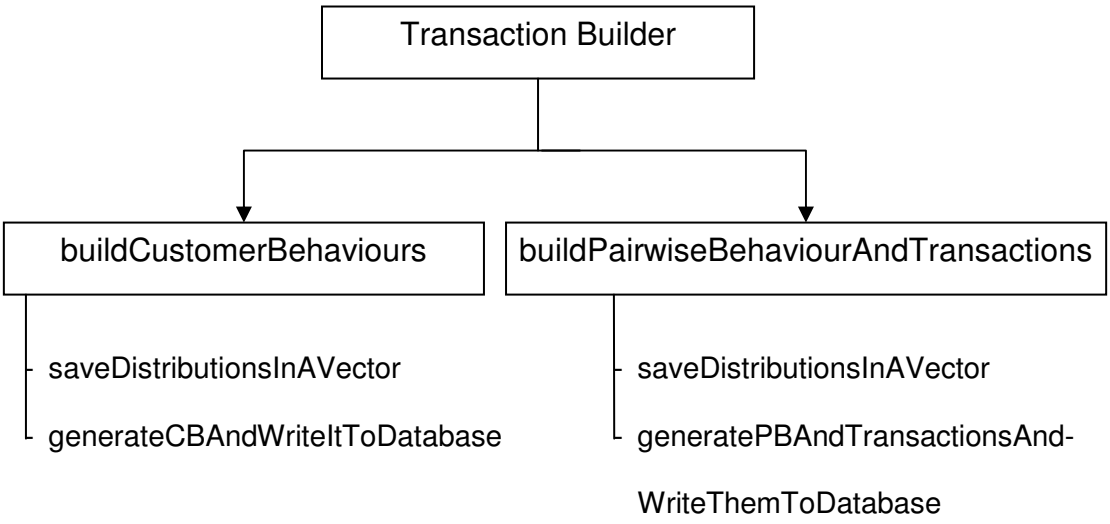

Abb. 6: Aufbau des Transaction Builders: Die wichtigsten Methoden

In den Kapiteln 2.2.1 und 2.2.2 wird etwas näher auf den Ablauf des Programms eingegangen. Es wird sich zeigen, dass die Generierung der synthetischen Banktransaktionsdaten sich schwieriger gestaltet als die zuvor behandelte Evaluation der Verteilungen.

2.2.1 buildCustomerBehaviours

Die Methode buildCustomerBehaviours hat die Aufgabe, aus den durch den Transaction Evaluator zur Verfügung gestellten Mittelwerten und Standard-abweichungen (Abbildung 5) synthetisches Kundenverhalten zu generieren. Dies stellt den komplexesten Schritt im ganzen Prozess dar, da hier im Hinblick auf den Ausfall gewisser Kundenbeziehungen infolge ungleicher Anzahl von Auftraggebern und Begünstigten Massnahmen eingeleitet werden müssen. Das heisst, es kann sein, dass in einem Kundenbeziehungs-Cluster entweder mehr Auftraggeber oder mehr Begünstigte generiert werden, die dann keine Partner finden. Diese Ausfallwahrscheinlichkeit muss in diesem Schritt kompensiert werden.

In der Methode buildCustomerBehaviours läuft nacheinander für jeden Kundenverhaltens-Cluster folgender Prozess ab: Die Mittelwerte und Standardabweichungen jedes Kundenbeziehungs-Clusters werden in Normalverteilungen umgesetzt. Dazu wird für alle Werte in einem definierten Bereich (siehe Abdeckung in Kapitel 2.3) die Wahrscheinlichkeit ermittelt und gespeichert. Da die Werte, wie weiter oben bereits erwähnt, wegen den späteren Ausfällen an dieser Stelle etwas erhöht werden müssen, durchlaufen die Verteilungen eine Verfeinerung (Refinement). In diesem Prozess wird die Wahrscheinlichkeit von grösseren Werten so lange erhöht, bis der Mittelwert plus eine Kompensation (siehe pairwiseBehaviourFailure in Kapitel 2.3) erreicht ist.

ID	CL	PB_ORI0	PB_ORI1	…	PB_BEN0	PB_BEN1	…
0	0	1	0	…	0	1	…
1	1	0	1	…	1	0	…
…	…	…	…	…	…	…	…

ID (Kundennummer), CL = CLUSTER (Kundenverhaltens-Cluster)

Tab. 9: Datenbanktabelle CUSTOMER_BEHAVIOUR_BUILT

Im Zuge der Generierung der Verteilungen wird auch die Anzahl der Kunden in den Originaldaten, die dem betrachteten Kundenverhaltens-Cluster angehören, ermittelt. Diese gibt an, wie viele Kunden aus dem vorliegenden Kundenverhaltens-Cluster erstellt werden sollen (Erweiterung: siehe Verhältnis zwischen Modell und Original in Kapitel 2.3). Aufgrund der zuvor abgespeicherten Verteilungen wird das synthetische Kundenverhalten generiert.

2.2.2 buildPairwiseBehavioursAndTransactions

Die im vorhergehenden Kapitel besprochene Methode buildCustomerBehaviours hat bereits gute Vorarbeit geleistet, in dem sie die Anzahl der

Auftraggeber und der Begünstigten etwas erhöht hat, um die nun folgenden Ausfälle zu kompensieren.

In der in diesem Kapitel besprochenen Methode buildPairwiseBehaviours-AndTransactions wird ein Kundenbeziehungs-Cluster nach dem anderen behandelt. Zuerst wird die Vorbereitung der Verteilungen der verschiedenen Transaktionen-Bags in Angriff genommen. Aus den vom Transaction Evaluator gelieferten Mittelwerten und Standardabweichungen (Abbildung 5) werden an dieser Stelle Normalverteilungen gebildet. Da negative Werte und Werte ausserhalb der Abdeckung (siehe Abdeckung in Kapitel 2.3) wegfallen und somit die Mittelwerte nicht ganz erreicht werden, durchlaufen auch diese Verteilungen eine Verfeinerung (Refinement), welche sie bis zu einer gewissen Genauigkeit (siehe Genauigkeit in Kapitel 2.3) an den wahren Mittelwert heranbringt.

O	B	CL	TRXOUT0	TRXOUT1	…	TRXIN0	TRXIN1	…
0	1	0	2	2	…	3	0	…
17	0	1	2	1	…	0	1	…
1	14	1	2	1	…	0	2	…
…	…	…	…	…	…	…	…	…

O = ORIGINATOR, B = BENEFICIARY, CL = CLUSTER (Kundenbeziehungs-Cluster)

Tab. 10: Datenbanktabelle PAIRWISE_BEHAVIOUR_BUILT

Sobald die Verteilungen vorbereitet sind, geht es darum, Kundenbeziehungen zu bilden. Dazu werden die Auftraggeber und die Begünstigten im vorliegenden Kundenbeziehungs-Cluster ermittelt und es wird versucht, Kundenbeziehungen zu erstellen. Für die erstellten Kundenbeziehungen wird mit Hilfe der generierten Verteilungen die Anzahl Transaktionen pro Bag definiert. Jeder einzelnen Transaktion wird mittels einer Zufallszahl ein Betrag innerhalb der Bag-Grenzen zugeordnet und die Transaktion wird zusätzlich mit ID und Datum versehen in der Datenbank abgelegt.

2.3 Configuration

Die Klasse Configuration dient der zentralen Abspeicherung von diversen Informationen, die sowohl vom Transaction Evaluator als auch vom Transaction Builder benötigt werden. Um die Datenbank konfigurieren zu können, sind dort der Treiber, die URL, der Benutzername und das zugehörige Passwort abgelegt. Weiter kann angegeben werden, wie die Tabelle benannt ist, in der die zu evaluierenden Transaktionen abgespeichert sind. Die Grenzen der Transaktionen-Bags sowie die Anzahl der Kundenbeziehungs- und der Kundenverhaltens-Clusters werden ebenfalls in der Klasse Configuration hinterlegt.

Der zweite Teil der Einstellungen ist für die Konfiguration des Transaction Builders vorgesehen. Auf der einen Seite gibt es dort Konstanten, deren Werte vor dem Start des Programms verändert werden können oder sogar müssen, und auf der anderen Seite sind Werte hinterlegt, die experimentell ermittelt wurden und deshalb nicht verändert werden sollten.

Mit der Abdeckung (covering) wird eingestellt, welche Werte bei der Generierung der Verteilungen auf ihre Wahrscheinlichkeit überprüft werden. Es werden alle Werte berücksichtigt, die im Intervall [Mittelwert – (Abdeckung * Standardabweichung)] bis [Mittelwert + (Abdeckung * Standardabweichung)] liegen[6].

Die Genauigkeit (accuracy) gibt an, welche Zahl mit dem errechneten Wahrscheinlichkeitswert multipliziert wird, bevor das Produkt auf eine ganze Zahl gerundet wird[7].

Das Verhältnis zwischen Modell und Original (ratioModelOriginal) bietet die Möglichkeit, die synthetische Datenmenge bei gleich gross bleibender Quelle zu erhöhen. Für ratioModelOriginal = 2 zum Beispiel werden aus jedem Kunden-

[6] Mit einer Abdeckung = 2 werden 95.45% der Fälle abgedeckt, mit Abdeckung = 3 99.73% der Fälle. Aus [Wikipedia: Normalverteilung]

[7] Für Genauigkeit = 100 wird auf Prozent gerundet; für Genauigkeit = 1000 auf Promille.

verhaltens-Cluster doppelt so viele Kunden generiert, wie in den Originaldaten zu finden sind.

Da es bei der Bildung von Kundenbeziehungen vorkommen kann, dass ein Auftraggeber (originator), obwohl er noch mehr Beziehungen eingehen möchte, bereits mit allen noch zur Verfügung stehenden Begünstigten (beneficiaries) eine Kundenbeziehung aufgebaut hat, wird die Suche nach einer gewissen Zeit abgebrochen. In maxTriesToFindABeneficiary wird die maximale Anzahl Versuche definiert.

Dadurch, dass eine Zufallszahl mitbestimmt, wie oft ein Kunde als Auftraggeber (originator) und als Begünstigter (beneficiary) agiert, kann die Anzahl der Auftraggeber und der Begünstigten in jedem Kundenverhaltens-Cluster voneinander abweichen. Um die gleiche Anzahl Kundenbeziehungen wie in den Originaldaten zu erhalten, muss die Summe dieser Unterschiede ermittelt und unter pairwiseBehaviourFailure abgespeichert werden, damit die Ausfälle kompensiert werden können.

Ein Experiment hat gezeigt, dass die Abweichungen zwischen der Anzahl der Auftraggeber und der Anzahl der Begünstigten in einem Kundenbeziehungs-Cluster mit der Anzahl der Kundenbeziehungen in diesem Cluster wächst, jedoch weniger stark als linear. Durch den Einsatz einer Wurzelfunktion wird dieser Erkenntnis Rechnung getragen. Die Werte powerValueCluster, powerValueTotal, compensatorAdapter und meanReduction, die in diesem Zusammenhang auftreten, wurden experimentell ermittelt.

Da in den Verteilungen keine negativen Werte und keine Werte ausserhalb der Abdeckung vorkommen, wird der ermittelte Mittelwert der Anzahl Transaktionen in einem Bag vor der Verfeinerung nicht erreicht. Der meanExactness gibt an, wie viel der Mittelwert der Verteilung vom wirklichen Mittelwert höchstens abweichen darf, damit die Verfeinerung beendet wird.

Weiter ist in der Klasse Configuration hinterlegt, in welchen Zeitraum (zwischen startdate und enddate) die synthetischen Transaktionen stattfinden.

3. Evaluation

In diesem Kapitel wird der in Kapitel 2 erläuterte Programmcode anhand zweier verschiedener Datenmengen auf seine Leistungsfähigkeit überprüft. Dazu wurde die Klasse ResultChecker implementiert, welche den Transaction Builder mehrmals nacheinander startet und die Resultate am Ende jedes Durchlaufs aufzeichnet. Bevor anhand der Testdaten die Stärken und Schwächen ermittelt werden, wird erläutert, was der ResultChecker aufzeichnet.

3.1 ResultChecker

Der ResultChecker macht zwei verschiedene Auswertungen: Auf der einen Seite werden die Anzahl Kundenbeziehungen pro Kundenbeziehungs-Cluster evaluiert. Auf der anderen Seite wird abgespeichert, wie viele Transaktionen aus jedem Bag ausgeführt worden sind.

Vor dem Start kann angegeben werden, wie viele Durchläufe ausgeführt werden sollen. Nachdem der ResultChecker gestartet wurde, werden als Erstes die zwei oben beschriebenen Masse aus den Originaldaten herausgelesen. Danach wird erstmals der Builder Engine laufen gelassen, bevor dessen Resultate evaluiert und abgespeichert werden und allenfalls ein weiterer Durchlauf gestartet wird.

Da im Builder Engine Zufallszahlen einen Einfluss auf die Resultate haben, stimmen die für die realen und die synthetischen Daten ermittelten Werte nicht überein. Trotzdem gibt es eine Möglichkeit, die Leistung des Programms zu überprüfen: Wenn mehrere Durchläufe ausgewertet worden sind, sollte der Durchschnitt der Werte der Testläufe sich den Werten der Originaldaten angenähert haben.

3.2 Analyse

Wie in der Einleitung dieses Kapitels bereits erwähnt, wurden der Transaction Evaluator und der Transaction Builder mit zwei verschiedenen Datenmengen getestet. Die Testdaten wurden mit dem TVIS Simulator von Jonas Luell generiert. Die kleinere Datenmenge beinhaltet 13’793 Transaktionen, die grössere 52'844 Transaktionen. Sowohl die Datenmengen als auch die in der Folge beschriebenen Auswertungen sind auf der beiliegenden CD-ROM im Ordner Auswertungen zu finden.

Für beide Datenmengen wurden fünfzehn Durchläufe aufgezeichnet und es hat sich gezeigt, dass der Transaction Builder erfreuliche Resultate liefert. Die Abweichung zwischen den Originalwerten und den gemittelten Testwerten der Anzahl Kundenbeziehungen pro Kundenbeziehungs-Cluster beträgt im Durchschnitt über alle Clusters für die grössere Datenmenge gerade mal 2.12% (kleinere Datenmenge: 2.23%). Für die Anzahl Transaktionen pro Bag ist die Abweichung sogar noch kleiner: Sie beträgt 0.68% (0.75%).

Der Durchschnitt der Summe aller Kundenbeziehungen und Transaktionen über alle Testläufe bewegt sich sowohl für die kleinere als auch für die grössere Datenmenge um hundert Prozent der Originalwerte. Das heisst, der Transaction Builder generiert im Durchschnitt fast gleich viele Kundenbeziehungen und Transaktionen, wie in den Originaldaten vorhanden sind. Diese Resultate deuten darauf hin, dass die Verteilungen korrekt umgesetzt werden und somit gewährleistet ist, dass die durch den Transaction Evaluator ermittelten Verteilungsdaten optimal in den Transaction Builder hineinspielen.

Wenn davon ausgegangen wird, dass alle Kunden in einem Kunden-verhaltens-Cluster und alle Kundenbeziehungen in einem Kundenbeziehungs-Cluster ähnliche Eigenschaften zeigen, werden die synthetischen mit den realen Daten eine grosse Ähnlichkeit aufweisen. Im extremen Fall gibt es gleich viele Kundenverhaltens-Clusters wie Kunden und jeder Kunde verfügt über seine eigene Kundenverhaltens-Art. Da in diesem Fall, da die Standard-

abweichungen alle null sind, keine Differenzen zwischen der Anzahl der Auftraggeber und der Begünstigten zu erwarten sind, kann der pairwise-BehaviourFailure auf null gesetzt werden. Somit werden die Tabellen CUSTOMER_BEHAVIOUR und CUSTOMER_BEHAVIOUR_BUILT identisch sein. Das Gleiche gilt für die Tabellen PAIRWISE_BEHAVIOUR und PAIRWISE_BEHAVIOUR_BUILT, wenn die Anzahl Clusters mit der Anzahl Instanzen übereinstimmt. Auch die Höhe der Beträge der Transaktionen kann exakt rekonstruiert werden, falls für jede Betragshöhe ein Bag besteht.

Da in diesem beschriebenen Extremfall jedoch die Abstraktion und dadurch der beabsichtigte Datenschutz nicht gewährleistet sind, muss die Anzahl der Clusters bzw. Bags nach unten korrigiert werden. Wie die optimalen Einstellungen auszusehen haben, damit sowohl die Verbindung zu den Originaldaten gewahrt als auch eine gewisse Abstraktion erreicht wird, bleibt Gegenstand zukünftiger Forschung.

Für die Testdatenmengen wurde der pairwiseBehaviourFailure jeweils manuell ermittelt. Das heisst, der Builder wurde mehrmals gestartet und nach der Beendigung der Methode buildCustomerBehaviour gestoppt, um die Summe der Differenzen zwischen Auftraggebern und Begünstigten über alle Kundenbeziehungs-Clusters abzulesen. Schliesslich wurde der Durchschnitt dieser Summen ermittelt und in der Klasse Configuration hinterlegt. Die Automatisierung dieses Prozesses würde eine Verbesserung darstellen und es könnte damit einiges an Aufwand eingespart werden.

4. Fazit und Ausblick

Nachdem in Kapitel 2 der Transaction Evaluator und der Transaction Builder etwas genauer unter die Lupe genommen wurden, wurde in der Evaluation in Kapitel 3 deren Leistung überprüft. Es hat sich gezeigt, dass der eingeschlagene Weg zum Erfolg führt, jedoch die optimale Feinabstimmung noch zu ermitteln bleibt.

Zudem bleibt abzuwarten, wie das Java-Programm mit grossen Datenmengen fertig wird. Im Zusammenhang mit der Dissertation von Jonas Luell wird es den ersten Härtetest zu bestehen haben: Es werden voraussichtlich ungefähr eine Million synthetische Transaktionen generiert, die für das Trainieren und Testen der Werkzeuge zur Betrugserkennung eingesetzt werden können und in der Publikation veröffentlicht werden dürfen.

Wie bereits in Kapitel 2 erwähnt, kam die Idee des Clusterings von Jonas Luell. Die Ausarbeitung der Details, die Umsetzung in Java sowie die Verfassung der schriftlichen Arbeit konnte ich dann selbstständig vollziehen. Für das Clustering in Java habe ich die Weka-Bibliothek [Witten und Frank 2005] verwendet.

Zum Schluss bedanke ich mich bei meinem Betreuer Jonas Luell für seine Unterstützung während des gesamten Prozesses. In den vergangenen vier Monaten verbrachte ich viele interessante Stunden an der Bachelorarbeit und konnte schliesslich das zu Beginn formulierte Ziel erreichen: Der Transaction Evaluator und de Transaction Builder ermöglichen es, synthetische Banktransaktionsdaten mit realen Verteilungskennzahlen zu generieren.

5. Glossar

Um den Zusammenhang zwischen dem Java-Code und der vorliegenden Arbeit zu wahren und um Missverständnisse zu vermeiden, wird an dieser Stelle ein Glossar eingefügt.

Deutsch (schriftliche Arbeit)	**Englisch (Java-Code)**
Kundenbeziehung	pairwise behaviour
Kundenverhalten	customer behaviour
Auftraggeber (einer Transaktion)	originator
Begünstigter (einer Transaktion)	beneficiary
Mittelwert	mean
Standardabweichung	standard deviation

Literaturverzeichnis

1. Lundin, E., Kvarnström, H., Jonsson, E.: A Synthetic Fraud Data Generation Methodology. In: Proceedings of the 4th International Conference on Information and Communications Security (2002)
2. Lundin, E., Kvarnström, H., Jonsson, E.: Synthesizing Test Data for Fraud Detection Systems. In: Proceedings of the 19th Annual Computer Security Applications Conference (2003)
3. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Elsevier, Amsterdam (2005)
4. Bernstein, A., Luell, J.: Den Transaktionen auf der Spur. In: OecNews, 111, 18-19 (2008)
5. SUBDUE, www.ailab.wsu.edu/subdue (Abgerufen: 12.06.2008)
6. Mittelwert, http://de.wikipedia.org/wiki/Mittelwert (Abgerufen: 18.07.2008)
7. Standardabweichung, http://de.wikipedia.org/wiki/Standardabweichung (Abgerufen: 18.07.2008)
8. Normalverteilung, http://de.wikipedia.org/wiki/Normalverteilung (Abgerufen: 18.07.2008)

Bild- und Tabellennachweis

Alle Abbildungen und Tabellen sind eigene Darstellungen.