LWBS CA5-KB1

Prüfungsleistung im Fach Wissensbasierte Systeme

Gruppe 6

Patricia Schwander [pschwander@yahoo.com]
Florian Krüger [florian.krueger@projectserver.org]

Trainingsdaten

gruppe_ca5_kb1.csv

Inhaltsverzeichnis

Szenario der Versionenraummethode

Das vorliegende Szenario beschäftigt sich mit der Vorhersage der Buchauswahl einer durch bestimmte Attribute charakterisierten Testperson. Dazu werden die Testpersonen gebeten, in einem Raum zu warten, in dem drei Bücher auf einem Tisch liegen. Den Testpersonen wird nahegelegt in der Wartezeit in einem dieser Bücher zu lesen. Im Anschluss wird die Auswahl des Buches sowie die Attribute der Person aufgezeichnet.

Implementation im Bezug auf die Vorlesung

Die Implementation der Aufgabe erfolgt in der objekt-orientierten Programmiersprache Java. Klassen-, Methoden- und Variablenbezeichnungen sowie die Dokumentation innerhalb des Programms sind in Englisch verfasst.

Der vorgegebene Datensatz (Format .csv) dient als “teaching input”. In der Implementation wird dieser als “trainingData” bezeichnet. Da es sich bei der gestellten Aufgabe um ein Lernsystem, das Kundenpräferenzen erlernen soll, handelt ist die “trainingData” die Grundmenge (Menge der (möglichen), beschreibenden Tupel der Kundendaten). Die einstellige, charakteristische Funktion auf dieser Grundmenge, das Konzept, ist in der Implementation auch als solches bezeichnet. Ein Beispiel entspricht einer Zeile aus der .csv Datei, also einem Teil der Grundmenge, in der Versionenraummethode auch Tupel genannt. Je nach dem um welches Buch es sich handelt, ist dieses als solches ein positives Beispiel und die Beispiele der anderen beiden Bücher sind negative Beispiele.

Das Konzept eines positiven Beispiel ist mit dem Wert 1 gleichzusetzen, das Konzept eines negativen Beispiels mit dem Wert 0. Die begrenzenden Mengen eines Versionenraum S und G sind in dieser Implementierung in der Klasse Star wiederzufinden. Erstellt werden diese beiden Mengen von der Methode “versionSpaceAlgo” (Teil der Klasse AlgorithmUtility), wobei in der Menge S die speziellsten Hypothesen (entsprechen in dieser Implementation der Klasse Concepts) und in der Menge G die allgemeinsten Hypothesen enthalten sind. Die Menge S besteht beim Stern Konzept aus einem positiven Beispiel. Durch den AQ-Algorithmus wird nun also für jedes positive Beispiel eine Menge G mit allen negativen Beispielen errechnet. Hypothesen entsprechen Konzepten und können als Constraints angesehen werden, da sie die Beispielmenge einschränken. S muss je nach Beispiel speziell verallgemeinert werden und G je nach Beispiel allgemein spezialisiert werden. Sobald die Mengen gleich sind, wird ein “Star” der beide enthält erstellt und das Konzept gilt als gelernt. Sollten S und/oder G leer werden, konnte das Konzept nicht erlernt werden und das Programm bricht ab.

Am Schluss ist für jedes Buch ein Konzept erlernt worden, anhand dessen nun mit der Methode “guessTheBook” (Teil der Klasse AlgorithmUtility) bestimmt werden kann, für welches

Buch sich eine Person mit bestimmten Attributen entscheiden würde.[1]

Attribute

Die folgenden Attribute einer Testperson stehen dem AQ-Basisalgorithmus bzw. der Versionenraummethode zur Verfügung.

Bezeichnung	JAVA Klasse[2]	Wertebereich
Altersgruppe	model.AgeClass	unter 18 Jahre, 19 - 24 Jahre, 25 - 35 Jahre, 36 - 49 Jahre, 50 - 65 Jahre, 65 Jahre und älter
Geschlecht	model.Gender	männlich, weiblich
Verheiratet	model.Married	ja, nein
Kinderzahl	model.Children	0 - unendlich[3]
Abschluss	model.Degree	keiner, Hauptschule, Realschule, Gymnasium, Hochschule, Promotion
Beruf	model.Profession	Angestellter, Arbeiter, Arbeitslos, Führungskraft, Hausfrau, Lehrer, Rentner, Selbstständig
Einkommen	model.Income	unter 1.000,

[1]Quelle: Skript Wissensbasierte System UNIT 2: Anwendung von Methoden Wissensbasierter Systeme, Teil: “Symbolische Lernverfahren”

[2]Bezieht sich auf die JAVA Klasse in dieser konkreten Implementierung. Zugunsten der Lesbarkeit wurde der volle Paketname gekürzt. Das hier abgedruckte Paket befindet sich tatsaechlich im Paket “dhbw.LWBS.CA5-KB1”.

[3]In dieser konkreten Implementierung wurde der Wertebereich auf sämtliche natürlichen Zahlen zwischen 0 und 20 eingeschränkt.

		1.000 - 1.999, 2.000 - 2.999, 3.000 - 3.999, 4.000 - 4.999, 5.000 und mehr

Tabelle 1: Übersicht aller verwendeten Attribute

Ziel des Szenarios

Das Ziel soll sein nach einer gewissen Anzahl von Beispielen (im Folgenden “Trainingsdaten” genannt) nur anhand der o.g. Attribute vorherzusagen, welches Buch die Testperson auswählen wird. Dazu soll der Benutzer des Systems die Daten der aktuell zu testenden Person eingeben können und daraufhin das Buch als Ergebnis angezeigt bekommen. Sollte die Versionenraummethode kein eindeutiges Ergebnis erzielen, so sollen alle möglichen Bücher angezeigt werden. Sollte keine Vorhersage möglich sein, wird dies ebenfalls ausgegeben.

Programmablauf

Der Programmablauf ist grob in zwei Phasen unterteilt: die Lernphase und die Bewertungsphase.

Lernphase

In der Lernphase werden der AQ-Basisalgorithmus sowie die Versionenraummethode angewandt um aus den Beispielen für jedes der drei Bücher eine Konzeptmenge (auch “Hypothesenmenge”) zu erzeugen.

Bewertungsphase

Im Anschluss an die Lernphase werden vorgelegte Beispieltupel (Attribute von Testpersonen) anhand der in der Lernphase erzeugten Konzeptmengen bewertet und für jedes Beispiel kein, ein oder mehrere mögliche Bücher als Ergebnis ausgegeben.

Grund der Verwendung bestimmter Strukturen innerhalb des Programms

Die Attribute der Eingabedaten werden in diesem Programmentwurf in Enums realisiert. Alle möglichen Werte sind als Konstanten festgelegt. Zum Beispiel gibt es das Enum “Married” welches die Konstanten “ja”, “nein”, “” und “*” enthält. Den Konstanten von Enums können Ids zugewiesen werden. Da im Algorithmus immer wieder Vergleiche durchgeführt werden, ist es wesentlich performanter Zahlen anstelle von Strings zu vergleichen.
Für alle Mengen innerhalb des Programmentwurfs werden entweder Listen oder Sets verwendet. Bei Listen kann ein Wert mehrere Male vorkommen, bei einem Set sind distinkte Werte garantiert. Es gibt unsortierte Sets wie z.B. HashSets und sortierte Sets wie z.B. TreeSets. Sortierte Sets werden immer dann verwendet wenn es wichtig ist in welcher Reihenfolge die Inhalte zurückgegeben werden. Deshalb werden diese für alle Mengen die S oder G entsprechen verwendet. Beispielsweise werden für positive und negative Beispiele Listen verwendet, da es hier durchaus möglich ist, dass doppelte Einträge vorhanden sind.

Eingabemerkmale

An drei unterschiedlichen Stellen im Programmablauf, können Daten von außen eingegeben werden: Zu Beginn die Trainingsdaten (verpflichtend) und im Anschluss entweder eine Liste von Testdaten zur automatischen Bewertung jedes einzelnen Tupels oder manuell als Eingabe in der JAVA Konsole eine beliebige Anzahl zur sequenziellen Bewertung.

Trainingsdaten

Die Trainingsdaten werden zu Beginn des Programmablaufs aus einer *.csv Datei gelesen. Dabei enthält die erste Zeile der .csv Datei die Spaltenbezeichner (siehe Tabelle 1: “Bezeichner”) und jede weitere Zeile je ein Tupel (die Attribute einer Testperson). Der Name dieser Datei wird dem Programm als Kommandozeilenparameter mitgegeben.

Zur Entwicklung des Systems stand eine .csv Datei mit 70 Datensätzen zur Verfügung.

Testdaten (Listenverarbeitung)

Sollen im Anschluss an den Lernprozess mehrere Testpersonen bewertet werden, ist es möglich dem Programm einen zweiten Kommandozeilenparameter mitzugeben. Diesen Parameter wird das Programm als Namen einer *.csv Datei interpretieren und versuchen die Daten der Testpersonen im Trainingsdatenformat einzulesen.

Testdaten (Einzelverarbeitung)

Wird keine Datei angegeben, wechselt das Programm im Anschluss an die Trainingsphase in den interaktiven Modus, was den Benutzer dazu befähigt die Attribute der Testperson einzeln einzugeben. Nach der vollständige Eingabe aller Attribute einer Person wird diese sofort bewertet und dem Benutzer das Ergebnis angezeigt. Für die Eingabe der Werte sind die in Tabelle 1 angegebenen Werte möglich.

```
1  Please enter the attributes of the person to be tested (blank for "_")
2
3  AgeClass : <18
4  Gender : w
5  Married : ja
6  Children : 0
7  Degree : Gymnasium
8  Profession : Angestellter
9  Income : <1000
10
11 Proof 0 (_): Buch_B
```

```
12 Do you want to test more persons? (y/n) : n
```

Listing 1: Eingabe der Attribute in der Einzelverarbeitung der Bewertungsphase

Zur Vereinfachung der Eingabe akzeptiert die Konsole bei den beiden klassifizierten Attributen Altersklasse (`AgeClass`) und Einkommen (`Income`) anstatt der Klassenbezeichnung (z.B. “19-24”) auch einen absoluten Wert, der dann in die richtige Klasse umgewandelt wird. Da die Klassen Lücken haben, wird bei konkreten Werten folgendes Mapping verwendet:

Attribut	**Klasse**	**Akzeptierter Wertebereich**	**Alternative Eingabe**
Altersklasse	< 18	(-∞ .. 18]	<18
	19 - 24	[19 .. 24]	19-24
	25 - 35	[25 .. 35]	25-35
	36 - 49	[36 .. 49]	36-49
	50 - 65	[50 .. 65]	50-65
	> 65	[66 .. ∞)	>65
Einkommen	< 1.000	(-∞ .. 1.000]	<1000
	1.000 - 1.999	[1.000 .. 1.999]	1000-1999
	2.000 - 2.999	[2.000 .. 2.999]	2000-2999
	3.000 - 3.999	[3.000 .. 3.999]	3000-3999
	4.000 - 4.999	[4.000 .. 4.999]	4000-4999
	5.000 und mehr	[5.000 .. ∞)	5000 und mehr

Ausgabemerkmale

Das Programm erzeugt während des Durchlaufen eine kontrollierbare Menge an Ausgaben auf der JAVA Konsole. Zur Ausgabe wurde das Apache.org log4j Framework[4] verwendet. Das Level der auszugebenden Details auf der Konsole kann in der Datei `log4j.properties` eingestellt werden.

Die folgenden Level werden vom Programm verwendet und stehen zur Verfügung:

Level	**Beschreibung**
`TRACE`	Das niedrigste Level mit den meisten Ausgaben. Es werden sämtliche Änderungen an allen Konzeptmengen mit Begründung der Änderung auf der Konsole ausgegeben. Es werden mehrere 1.000 Zeilen während der Lernphase erzeugt.
`DEBUG`	Ein Level auf dem immer noch sehr viele Details auf der Konsole ausgegeben werden, allerdings werden Aktionen in sehr tiefen Unterroutinen verschwiegen.
`INFO`	Die wichtigsten Aktionen während der Lernphase werden auf der Konsole ausgegeben. Dieses Level erzeugt Ausgaben über jeden Zwischenstand nach den einzelnen Durchläufen der Versionenraummethode und des AQ-Basisalgorithmus.
`WARN`	Die Aktionen der Algorithmen werden komplett verschwiegen, es erfolgt lediglich im Fehlerfall eine Ausgabe. (Der Standardwert)
`ERROR`	nicht verwendet
`FATAL`	nicht verwendet

Zur Änderung des Ausgabelevels muss der Wert `log4j.logger.dhbw.LWBS.CA5_KB1` in der Datei `log4j.properties` vor der Ausführung des Programms geändert werden. Es ist zu beachten, dass ein niedrigeres Ausgabelevel mit mehr Ausgaben aufgrund des erhöhten I/O Verkehrs zu einer signifikanten Verzögerung des Programmablaufs führt.

Lernphase

Anhand von Buch A soll eine Beispiel Lernphase ausgeben werden. Es werden Teile der Ausgabe zur besseren Übersicht ausgelassen, gezeigt wird nur die Ausgabe nachdem der AQ-Algorithmus und somit auch die Versionenraummethode durchgeführt wurden. Zur vollständigen Ausgabe kann das Programm in der JAVA Konsole mit dem Modus `TRACE` ausgeführt werden.

---

[4]http://logging.apache.org/log4j/1.2/manual.html

```
INFO  -
Star:
S:  (36-49, w, nein, 0, keiner, Hausfrau, <1000)
G:  (36-49, *, *, *, keiner, *, *)
    (*, *, *, *, *, Hausfrau, <1000)
    [...]
    (*, w, nein, 0, keiner, *, *)
    (36-49, w, nein, *, keiner, *, *)
    [...]
    (36-49, *, *, 0, *, Hausfrau, <1000)
    (36-49, w, nein, 0, *, Hausfrau, *)
    [...]
    (*, w, *, *, keiner, Hausfrau, <1000)
    (*, *, nein, *, keiner, Hausfrau, <1000)

DEBUG - [BEGIN] BEST CONCEPT
Best concept of current Star:(*, w, nein, 0, keiner, *, *)
DEBUG - [ END ] BEST CONCEPT
[BEGIN] ADD IFNOTALREADYCOVERED
[ END ] ADD IFNOTALREADYCOVERED
[FINISHED AQ ALGORITHM FOR BOOK A]

[CONCEPTS FOR BOOK A]:
 [(*, *, ja, *, keiner, *, *), (*, *, *, *, keiner, *, 1000-1999), (*, *, *, *, keiner, *, 2000-2999), (*, w, nein, 0, keiner, *, *), (36-49, *, *, *, keiner, *, *)]

[ LEARNING COMPLETED ]
```

Listing 2: Ausgabe der Lernerfolge bzgl. Buch A (Auszug)

Bewertungsphase (Listenverarbeitung)

Nach erfolgreichem Einlesen wird jedes einzelne Tupel bewertet und dem Benutzer die Ergebnisse der Auswertung wie in Listing 3 angegeben:

```
Proof 30 (Buch_A): Buch_A
Proof 31 (Buch_C): Buch_C
Proof 32 (Buch_C): Buch_C
Proof 33 (Buch_A): Buch_A or Buch_C
Proof 34 (Buch_C): Buch_C
```

Listing 3: Ausgabe der Bewertungsergebnisse in der Listenverarbeitung (Auszug)

Sofern in den Testdaten ein “korrektes” Ergebnis als Wert für Buch angeben wird, wird diese zwischen den Klammern angegeben. Hinter dem Doppelpunkt befindet sich das Ergebnis der Bewertung. In Zeile 4 liegt ein nicht-eindeutiges Ergebnis vor, deshalb wird sowohl Buch A als auch Buch C als mögliches Ergebnis ausgegeben.

Bewertungsphase (Einzelverarbeitung)

Die Ausgabe der Einzelverarbeitung beschränkt sich lediglich auf eine Zeile, welche die vom Bewertungsalgorithmus zurückgelieferten möglichen Bücher ausgibt.

```
1  Proof 0 (_): Buch_B
```

Listing 4: Ausgabe der Bewertungsergebnisse in der Einzelverarbeitung (Auszug)

Die Ausgabe wird durch dieselbe Methode erzeugt wie die Ausgabe der Listenverarbeitung, deshalb wird auch hier die Testnummer (immer “0”) und die “korrekte” Lösung (da in diesem Modus nicht gegeben immer “_”) angezeigt.