Introduction to Data and Knowledge Engineering Lösungsvorschlag zu Tutorium 9

TECHNISCHE UNIVERSITÄT DARMSTADT

9. 2 Finden von Literalen I

TECHNISCHE UNIVERSITÄT DARMSTADT

In einem Verein soll ein neuer Vorsitzender gewählt werden. Es soll ein neues Prädikat `votes_for(A, B)` gelernt werden, welches bestimmt, wann ein Vereinsmitglied A für ein anderes Vereinsmitglied B stimmt. Das Hintergrundwissen ist mithilfe der folgenden Prädikate angegeben:

- `candidate(A)` - A ist ein Kandidat bei der Wahl zum Vorsitzenden.
- `friend(A, B)` - A ist mit B befreundet.

Wir wollen uns in dieser Aufgabe darauf beschränken, alle Literale aufzulisten, welche zum Aufstellen des Prädikats `votes_for(A, B)` überprüft werden müssten. Zur Vereinfachung verzichten wir auf das Einführen neuer Variablen in Literalen, d.h. es werden nur Variablen betrachtet, die bereits im zu lernenden Prädikat vorkommen.

9. 2 Finden von Literalen II

TECHNISCHE UNIVERSITÄT DARMSTADT

Lösungsvorschlag:

```
 candidate(A)     friend(A, A)     friend (B, B)
\+candidate(A)   \+friend(A, A)   \+friend(B, B)
 candidate(B)     friend(A, B)     friend (B, A)
\+candidate(B)   \+friend(A, B)   \+friend(B, A)
```

9. 2 Finden von Literalen III

TECHNISCHE UNIVERSITÄT DARMSTADT

Diskussion:

Je nach Definition der Prädikate `candidate(A)` und `friend(A, B)` sind nicht alle Literale sinnvoll zu betrachten und könnten bei einer Optimierung des Verfahrens ausgelassen werden.
Geht man davon aus, dass niemand mit sich selbst befreundet ist, können die folgenden 4 Literale entfernt werden:
`friend(A, A)`, `friend (B, B)`, `\+friend(A, A)`, `\+friend (B, B)`

Normalerweise würden auch Literale mit neuen Variablen betrachtet werden, die hier zur Vereinfachung ausgelassen wurden, beispielsweise `friend(C, A)` oder `friend(B, C)`.

9.3 Heuristiken I

TECHNISCHE UNIVERSITÄT DARMSTADT

In Separate-and-Conquer-Algorithmen kann zur Auswahl von Literalen eine Heuristik verwendet werden, welche nach Möglichkeit einen hohen Wert liefern soll, wenn die Hinzunahme des Literals die Regel verbessert. Was genau das heißt, ist schwierig zu definieren, daher kann es keine Heuristik geben, die in jedem Fall richtig ist. Trotzdem erscheinen manche Heuristiken in der Mehrheit der Fälle „besser“ zu sein als andere.
Überlegen Sie, welche der folgenden Formeln sinnvolle Heuristiken sind und diskutieren Sie über Vor- und Nachteile der jeweiligen Formel bzw. ob Formeln gleichwertig sind.
p und n beschreiben jeweils die Anzahl der positiven bzw. negativen Trainingsbeispiele, die durch diese Regel abgedeckt werden.

9.3 Heuristiken II

TECHNISCHE UNIVERSITÄT DARMSTADT

- p - Eine eher nicht sinnvolle Heuristik. Eine Regel ist nicht automatisch gut, nur weil sie viele positive Beispiele abdeckt. Sie könnte ja auch genauso viele negative Beispiele abdecken, was von dieser Heuristik nicht berücksichtigt wird.
- $p - n$ - Diese Heuristik ist besser als die vorige, da hier auch die Anzahl der abgedeckten negativen Beispiele berücksichtigt wird. Eine hohe Differenz zwischen p und n bedeutet, dass viele positive Beispiele abgedeckt werden, aber nur wenige negative, was bei einer guten Regel ja gewünscht ist. Allerdings weist diese Heuristik einer Regel, welche 0 positive und 0 negative Beispiele abdeckt (also keinerlei Zusatzwissen einbringt) denselben Wert zu wie einer Regel, die 3 positive und 3 negative Beispiele abdeckt, was unter Umständen hilfreicher für die weitere Regelentwicklung sein kann als die andere Regel und daher einen höheren Wert haben sollte.

9.3 Heuristiken III

TECHNISCHE UNIVERSITÄT DARMSTADT

- $(p - n)^2$ - Diese Heuristik ist eher nicht sinnvoll, da durch das Quadrieren das Vorzeichen verschwindet. Eine Regel, die 0 positive und 4 negative Beispiele abdeckt, bekommt den gleichen heuristischen Wert zugewiesen wie eine Regel, die 4 positive und 0 negative Beispiele abdeckt.
- $(p - n)^3$ - Bei dieser Heuristik wird das Vorzeichen beachtet. Der einzige Unterschied zu $p - n$ ist, dass ohnehin hohe Werte noch höher sind. Diese Heuristik ist zu $p - n$ gleichwertig, da die Ordnung von zugeordneten Werten sich durch das Potenzieren mit streng monotonen Funktionen nicht verändert. Hier besteht außerdem dasselbe Problem wie bei der Heuristik $p - n$, da 0^3 ebenfalls 0 ergibt.

9.3 Heuristiken IV

TECHNISCHE UNIVERSITÄT DARMSTADT

- $\frac{p}{p+n}$ - Diese Heuristik ist in einiger Hinsicht besser als $p - n$, da hier eine Regel, die 0 positive und 0 negative Beispiele abdeckt, einen niedrigeren Wert erhält als eine Regel, die gleich viele positive und negative Beispiele abdeckt (größer 0). Allerdings bewirkt das miteinander ins Verhältnis setzen auch, dass eine Regel, die 1 positives Beispiel und 0 negative abdeckt, denselben Wert erhält wie eine Regel, die 1000 positive und 0 negative Beispiele abdeckt, was bei $p - n$ nicht so wäre.
  Dieses Problem könnte man wiederum dadurch beheben, dass man den ganzen Bruch noch einmal mit p multipliziert, um nach der Anzahl der abgedeckten positiven Beispiele zu wichten (siehe Skript).

9.4 Gain I

TECHNISCHE UNIVERSITÄT DARMSTADT

Den „Gain“ einer Regel berechnet man, in dem man den Wert der gewählten Heuristik zur Auswahl von Literalen sowohl für die vorige Regel als auch als auch für die neue Regel, die unter Hinzunahme eines Literals entsteht, berechnet und den alten Gain vom neuen subtrahiert.
Geben Sie für die Heuristiken aus der vorigen Aufgabe jeweils die zugehörigen Gain-Formeln an.
Verwenden Sie p und n für die Anzahl der abgedeckten positiven bzw. negativen Beispiele aus der neuen Regel, p’ und n’ für die Anzahl der positiven bzw. negativen abgedeckten Beispiele in der vorherigen Regel (vor Hinzunahme des Literals).

9.4 Gain II

TECHNISCHE UNIVERSITÄT DARMSTADT

- $p - p'$
- $p - n - (p' - n')$
- $(p - n)^2 - (p' - n')^2$
- $(p - n)^3 - (p' - n')^3$
- $\frac{p}{p+n} - \frac{p'}{p'+n'}$

9.5 Separate-and-Conquer I

TECHNISCHE UNIVERSITÄT DARMSTADT

Wir betrachten das auf dem Aufgabenblatt gegebene Szenario mit Eseln und Pferden.
Es soll ein Prädikat `mule(A)` trainiert werden, welches bestimmt, wann ein Tier A ein Maultier ist.

Initialwerte des Algorithmus':
Abgedeckte Trainingsbeispiele durch die anfangs leere Regel:

- ⊕ `mule(feivel)`
- ⊕ `mule(rosy)`
- ⊕ `mule(lisa)`
- ⊖ `mule(sandy)`
- ⊖ `mule(berta)`
- ⊖ `mule(emil)`

Anfangswert für $\frac{p'}{p'+n'} : \frac{3}{3+3} = \frac{1}{2}$, da $p' = 3$, $n' = 3$

9.5 Separate-and-Conquer II

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Regel, 1. Literal:

Bisherige Regel: leer; $\frac{p'}{p'+n'} = \frac{1}{2}$

betrachtete Beispiele: $\oplus$ `mule(feivel)`, $\oplus$ `mule(rosy)`, $\oplus$ `mule(lisa)`, $\ominus$ `mule(sandy)`, $\ominus$ `mule(berta)`, $\ominus$ `mule(emil)`

	p	n	Gain	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	3	3	0									
`father(A, C)`	0	1	$-\frac{1}{2}$									
`father(C, A)`	3	3	0									
`horse(A)`	0	1	$-\frac{1}{2}$									
`horse(B)`	-	-	-									
`horse(C)`	-	-	-									
`donkey(B)`	-	-	-									
`donkey(C)`	-	-	-									
`mule(B)`	-	-	-									

9.5 Separate-and-Conquer III

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Regel, 2. Literal:

Bisherige Regel: `mule(A) :- mother(B, A)`; $\frac{p'}{p'+n'} = \frac{1}{2}$ (vorige p- und n-Werte);
betrachtete Beispiele: $\oplus$ `mule(feivel)`, $\oplus$ `mule(rosy)`, $\oplus$ `mule(lisa)`,
$\ominus$ `mule(sandy)`, $\ominus$ `mule(berta)`, $\ominus$ `mule(emil)`

	p	n	Gain	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	3	3	0	-	-	-						
`father(A, C)`	0	1	$-\frac{1}{2}$	0	1	$-\frac{1}{2}$						
`father(C, A)`	3	3	0	3	3	0						
`horse(A)`	0	1	$-\frac{1}{2}$	0	1	$-\frac{1}{2}$						
`horse(B)`	-	-	-	2	1	$\frac{1}{6}$						
`horse(C)`	-	-	-	-	-	-						
`donkey(B)`	-	-	-	0	2	$-\frac{1}{2}$						
`donkey(C)`	-	-	-	-	-	-						
`mule(B)`	-	-	-	1	0	$\frac{1}{2}$						

9.5 Separate-and-Conquer IV

TECHNISCHE UNIVERSITÄT DARMSTADT

An dieser Stelle haben wir die erste Regel gelernt:
`mule(A) :- mother(B, A), mule(B)`
Durch diese Regel wird das positive Beispiel `mule(lisa)` abgedeckt; es sind aber noch 2 positive Beispiele nicht abgedeckt, es muss also mindestens eine weitere Regel gelernt werden. Daher fahren wir fort mit den verbleibenden positiven Beispielen und den Negativbeispielen:

- ⊕ `mule(feivel)`
- ⊕ `mule(rosy)`
- ⊖ `mule(sandy)`
- ⊖ `mule(berta)`
- ⊖ `mule(emil)`

Anfangswert von $\frac{p'}{p'+n'}$: $\frac{2}{2+3} = \frac{2}{5}$, da $p' = 2$, $n' = 3$

9.5 Separate-and-Conquer V

TECHNISCHE UNIVERSITÄT DARMSTADT

2. Regel, 1. Literal:

Bisherige Regel: leer; $\frac{p'}{p'+n'} = \frac{2}{5}$;
betrachtete Beispiele: $\oplus$ `mule(feivel)`, $\oplus$ `mule(rosy)`,
$\ominus$ `mule(sandy)`, $\ominus$ `mule(berta)`, $\ominus$ `mule(emil)`

	p	n	Gain	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	2	3	0									
`father(A, C)`	0	1	$-\frac{2}{5}$									
`father(C, A)`	2	3	0									
`horse(A)`	0	1	$-\frac{2}{5}$									
`horse(B)`	-	-	-									
`horse(C)`	-	-	-									
`donkey(B)`	-	-	-									
`donkey(C)`	-	-	-									
`mule(B)`	-	-	-									

9.5 Separate-and-Conquer VI

TECHNISCHE UNIVERSITÄT DARMSTADT

2. Regel, 2. Literal:

Bisherige Regel: `mule(A) :- mother(B, A)`; $\frac{p'}{p'+n'} = \frac{2}{5}$;
betrachtete Beispiele: $\oplus$ `mule(feivel)`, $\oplus$ `mule(rosy)`,
$\ominus$ `mule(sandy)`, $\ominus$ `mule(berta)`,$\ominus$ `mule(emil)`

	p	n	Gain	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	2	3	0	-	-	-						
`father(A, C)`	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{5}$						
`father(C, A)`	2	3	0	2	3	0						
`horse(A)`	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{5}$						
`horse(B)`	-	-	-	2	1	$\frac{4}{15}$						
`horse(C)`	-	-	-	-	-	-						
`donkey(B)`	-	-	-	0	2	$-\frac{2}{5}$						
`donkey(C)`	-	-	-	-	-	-						
`mule(B)`	-	-	-	0	0	-						

9.5 Separate-and-Conquer VII

TECHNISCHE UNIVERSITÄT DARMSTADT

2. Regel, 3. Literal:

Bisherige Regel: `mule(A) :- mother(B, A), horse(B);` $\frac{p'}{p'+n'} = \frac{2}{3}$;
betrachtete Beispiele: $\oplus$ `mule(feivel)`, $\oplus$ `mule(rosy)`,
$\ominus$ `mule(sandy)`

	p	n	Gain	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	2	3	0	-	-	-	-	-	-			
`father(A, C)`	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{5}$	0	0	-			
`father(C, A)`	2	3	0	2	3	0	2	1	0			
`horse(A)`	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{3}$			
`horse(B)`	-	-	-	2	1	$\frac{4}{15}$	-	-	-			
`horse(C)`	-	-	-	-	-	-	-	-	-			
`donkey(B)`	-	-	-	0	2	$-\frac{2}{5}$	0	0	-			
`donkey(C)`	-	-	-	-	-	-	-	-	-			
`mule(B)`	-	-	-	0	0	-	0	0	-			

9.5 Separate-and-Conquer VIII

TECHNISCHE UNIVERSITÄT DARMSTADT

2. Regel, 4. Literal:

Bisherige Regel: `mule(A) :- mother(B, A), horse(B), father(A, C);`

$\frac{p'}{p'+n'} = \frac{2}{3}$;

betrachtete Beispiele: $\oplus$ `mule(feivel)`, $\oplus$ `mule(rosy)`,

$\ominus$ `mule(sandy)`

	p	n	Gain	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	2	3	0	-	-	-	-	-	-	-	-	
`father(A, C)`	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{5}$	0	0	-	0	0	-
`father(C, A)`	2	3	0	2	3	0	2	1	0	-	-	-
`horse(A)`	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{5}$	0	1	$-\frac{2}{3}$	0	1	$-\frac{2}{3}$
`horse(B)`	-	-	-	2	1	$\frac{4}{15}$	-	-	-	-	-	-
`horse(C)`	-	-	-	-	-	-	-	-	-	0	1	$-\frac{2}{3}$
`donkey(B)`	-	-	-	0	2	$-\frac{2}{5}$	0	0	-	0	0	-
`donkey(C)`	-	-	-	-	-	-	-	-	-	2	0	$\frac{1}{3}$
`mule(B)`	-	-	-	0	0	-	0	0	-	0	0	-

9.5 Separate-and-Conquer IX

TECHNISCHE UNIVERSITÄT DARMSTADT

Nun werden gar keine negativen Beispiele mehr abgedeckt. Die gefundene 2. Regel deckt außerdem alle verbleibenden positiven Beispiele ab, sodass der Separate-and-Conquer Algorithmus terminiert.

Die beiden gefundenen Regeln lauten:

```
mule(A) :- mother(B, A), mule(B)
mule(A) :- mother(B, A), horse(B), father (C, A), donkey(C)
```

Hinweis: Wir nehmen bei dem gelernten Prädikat an, dass wir insgesamt ein Datalog-Programm betrachten. In diesem Fall ist die Reihenfolge der Regeln unerheblich. Grundsätzlich geht es aber beim Lernen von Regeln auch zunächst nicht darum, fertige, ausführbare Programme zu generieren.

9.5 Separate-and-Conquer X

TECHNISCHE UNIVERSITÄT DARMSTADT

Diskussion

- Was passiert, wenn man im Falle von gleichen Gain-Werten andere Literale hinzunimmt als die Konvention der Aufgabe vorschlägt?

  Im vorliegenden Beispiel werden unter Umständen leicht andere Regeln gelernt. Bei vielen unterscheiden sich die dann gelernten Regeln nur in ihrer Reihenfolge und der Reihenfolge der Literale. Bei besonders ungünstiger Auswahl werden teilweise umständliche Regeln gelernt.

  Wird zum Beispiel bei der ersten Regel beim 1. Literal `father(C, A)` ausgewählt und danach die Auswahl wie vorher vorgenommen, kann die Regel `mule(A) :- father(C, A), mother(B, A), mule(B)` gelernt werden, die zwar nicht falsch ist, aber ein im Grunde überflüssiges Literal enthält.

9.5 Separate-and-Conquer XI

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was passiert, wenn man andere Kandidaten-Literale betrachtet?

  In dieser Aufgabe wurden bewusst bestimmte Vorgaben gemacht, um die Lösung eindeutig zu halten und zu zeigen, wie der Algorithmus mehrere Regeln lernt.

  Normalerweise würden noch zahlreiche andere Literale vom Lernalgorithmus betrachtet werden, beispielsweise die Verneinung jedes der bisher gegebenen Literale und andere Literale mit anderen Variablenkombinationen.

  Dann werden andere Regeln gelernt, die zwar zunächst weniger intuitiv erscheinen (Beispiel siehe nächste Folie), aber auf das gegebene Hintergrundwissen und die gegebenen Beispiele passen (vielleicht aber versagen würden, wenn man das Hintergrundwissen und das zugrundeliegende Szenario erweitert). Bei ungünstiger Literalauswahl kann es auch wieder passieren, dass sehr umständliche Regeln gelernt werden, die überflüssige Literale enthalten und damit unter Umständen zu speziell sind (Problem „Overfitting“).

  Es ist also sinnvoll, den Algorithmus mit mehreren Auswahlstrategien zu starten und am Ende aus den gefundenen Regeln nach bestimmten Kriterien auszuwählen, beispielsweise nach der Länge der Regel, der Anzahl der gefundenen Regeln usw.

9.5 Separate-and-Conquer XII

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Beispiel für andere Regel, die gelernt werden kann:**

	p	n	Gain	p	n	Gain	p	n	Gain
`mother(B, A)`	3	3	0	3	2	0	-	-	-
`father(C, A)`	3	3	0	3	2	0	3	2	0
`horse(A)`	0	1	$-\frac{1}{2}$	0	0	-	0	0	-
`not horse(A)`	3	2	$\frac{1}{10}$	-	-	-	-	-	-
`donkey(B)`	-	-	-	-	-	-	0	2	$-\frac{2}{5}$
`not donkey(B)`	-	-	-	-	-	-	3	0	$\frac{3}{5}$

`mule(A) :- not horse(A), mother(B,A), not donkey(B)` deckt hier alle positiven Beispiele ab, aber keine negativen. Die Regel ist auch allgemein wahr, wenn man von der „Closed-World-Assumption“ ausgeht, die in diesem Fall besagen würde, dass die „Welt“ nur aus Pferden und Eseln besteht.
Diese Regel wäre allerdings nicht angemessen, wenn das Szenario beispielsweise um andere Tiere ergänzt wird - das Prädikat müsste dann neu gelernt werden, unter Hinzunahme von anderen Beispielen.

Fazit dieses Tutoriums

TECHNISCHE UNIVERSITÄT DARMSTADT

- Regeln lernen ist auch in scheinbar einfachen Fällen schwierig und hängt von vielen verschiedenen Faktoren ab:
- dem Szenario: sinnvolles gegebenes Hintergrundwissen, geschickte Wahl der positiven und negativen Trainingsbeispiele, um zu beeinflussen, ob Ausnahmen enthalten sein sollen oder nicht (ohne Hinzunahme des Sonderfalls `lisa` kann beispielsweise die Regel `mule(A) :- mother(B, A), mule(B)` auch nicht gelernt werden - dies kann unter Umständen aber auch gewünscht sein)
- dem Algorithmus an sich: verwendete Heuristik, betrachtete Kandidaten-Literale, Auswahl der Literale im Falle eines gleichen heuristischen Wertes