FI MU

Faculty of Informatics
Masaryk University

On Biases in Inductive Data Engineering

by

Jana Kuklová
Luboš Popelínský

FI MU Report Series FIMU-RS-95-01

March 1995

On Biases in Inductive Data Engineering

Jana Kuklová, Luboš Popelínský
Faculty of Informatics
Masaryk University, Brno
E-mail: {jana,popel}@fi.muni.cz

1 Introduction

An utilization of inductive reasoning [Flene94] in the database area looks promising [Agar93], [DeRae92], [DeRae93], [Flach93], [Kivi92], [Savnik93], [Manni93]. Inductive data engineering, as introduced in [Flach93], means a process of restructuring database by means of induction.

In our work we focus on exploitation of inductive logic programming for database schema design [Roll92]. We propose modifications of INDEX, described in [Flach93], namely new biases for narrowing search space, as well as stopping criterion.

In this article the case of a single relation is described.

2 Example

Let we have a relation `building/6` with attributes

```
Address, FlatNumber, Rent, NumRooms, Garden, Style
```

described by examples

```
building(a1,1,100,3,n,s1)
building(a1,2,100,2,n,s1)
building(a1,3,200,5,n,s1)
building(a2,1,100,4,n,s2)
```

```
building(a2,2,200,5,n,s2)
building(a2,3,200,2,n,s2)
building(a2,4,100,2,n,s2)
building(a3,1,200,4,n,s1)
building(a3,2,200,3,n,s1)
building(a3,3,300,5,n,s1)
building(a4,1,400,5,y,s3)
building(a5,1,200,6,y,s4)
building(a6,1,700,5,y,s3)
building(a7,1,600,7,y,s5)
building(a8,1,400,4,y,s4)
building(a9,1,400,5,n,s1)
building(a9,2,500,5,n,s1)
building(a9,3,500,5,n,s1)
building(a9,4,600,5,n,s1)
building(a10,1,400,7,y,s5)
```

The intended decomposition is

```
building(Address, FlatNumber, Rent, NumRooms, Garden, Style):-
        house(Address, Garden, Style),
        flat(Address, FlatNumber, Rent, NumRooms).
```

i.e. there are attributes related only to houses and flats respectively. Moreover, `house/3` relation can be decomposed as follows

```
house(Address, Garden, Style) :-
        houseStyl(Address,Style),
        houseGarden(Style, Garden).
```

3 INDEX

INDEX, a program for inducing attribute dependencies and an interactive decomposition of database relations, is described elsewhere [Flach93]. Here we only describe some topics which are important for our task.

Definition: *Given a relation R, attribute Attr2 is* **functionaly dependent** *on attribute Attr1 (written as [Attr1]–>[Attr2]) iff whenever two tuples of R* *agree on their Attr1-value, they also agree on their Attr2-value.*

That is, at any instant in time each value of Attr1 has exactly one value of Attr2 associated with it. The left side is called antecedent, the right one consequent.

Definition: *Given a relation R(Attr1, Attr2, Attr3), the* **multivalued dependency** *[Attr1]->->[Attr2] holds in R iff the set of Attr2-values matching a given (Attr1-value, Attr3-value) pair in R depends only on the Attr1-value and is independent of the Attr3-value.*

Both kinds of dependencies can be defined analogicaly for more than one attributes in antecedent.

INDEX finds all both functional and multivalued dependencies which hold on examples of the given relation. Then, user is asked to choose one of dependencies which is most suitable for decomposition of the relation.

3.1 Biases in INDEX

Search space of INDEX may be limited by declaration of attributes which may occur in consequents of dependencies.

Use of heuristics built in INDEX is another possibility. However it seems not to be easy for a database designer to set the appropriate parameters of those heuristics. We will not discuss it here more.

The main problem of INDEX is that, as a rule, too much dependencies is found, and user **has to know** which of them to choose for building database schema.

The solution for making database schema design easier is described in the following sections.

4 New biases for INDEX

4.1 Restriction of the initial set of candidates

switch fd, switch mvd

For our purposes it was useful to propose two new switches, fd and mvd, switching on/off the induction of functional and multivalued dependencies. For all computation the switch mvd was off, i.e. only functional dependencies were generated.

attributes in antecedent

As sometimes it has no sense to generate combinations of all attributes in antecedents, we suggest to declare those attributes which are not relevant for a given task.

However, the principal improvement of INDEX is described in the following section.

4.2 Stopping criterion

The idea of decreasing the number of dependencies being induced is based on the fact that the occurence of many dependencies follows from the existence of the relation key.

The relation may contain a lot of keys, one of them is chosen as the primary key. It is a task of designer to choose the appropriate primary key. The system can only propose a candidate key, that is a set of attributes which forms a key of relation.

If we know the primary key, then

- every dependence which antecendent part is a subset of primary key corrupt the correct design of relation
- every dependence which antecendent attributes contains the primary key is correct

More, every dependence containing a subset of the primary key on its consequent part is probably correct. It implies that if we knew the primary key we could minimize the number of generated dependencies.

Consider relation R(A1,...,An), let K is a subset of A1,...,An. K is a candidate key of R when K satisfy following two condition: [Hugh91].

1. The value of K uniquely identifies each tuple in R.
2. No attribute in K can be discarded without destroying 1.

Let us formulate these conditions, actually a stopping criterion, using functional dependencies. All non-key attributes of R must be functionally dependent on K. INDEX generates dependencies with increasing number of attributes in antecedents. As the next trivial statement is holds

> *If there is a set of attributes A, B such that [A]–>[B] then for all sets of attributes C, [A,C]–>[B] holds.*

a computation can be stopped in the moment when all attributes have appeared in dependencies having been induced. The candidate key is formed by a set of all attributes appearing in antecedents. A confirmation that the candidate key found is a primary key is up to designer.

5 Example - 2nd part

Let us consider the relation `building/6` mentioned above. INDEX finds 77 dependencies for this relation. For finding key it is enough to generate only first 5 dependencies. (switch mvd off).

```
* building:[adr]->[garden]
  building:[styl]->[garden]
* building:[adr]->[styl]
* building:[adr,f_num]->[rent]
* building:[adr,f_num]->[rooms]
```

Dependencies signed by asterics imply that attributes [adr, f_num] form a candidate key. Once a designer confirmed the primary key, we are looking for dependencies which corrupt the correct design of relation. Attributes `Garden` and `Style` depend only on a prime attribute `Address`, i.e. for a given `Address` and for all values of `FlatNumber` we have just one value of `Garden` and just one value of `Style`. It means that a redundance has appeared in this relation. So we have to decompose the `building/6` into two parts using

```
[adr]->[styl,garden]
```

and we have

```
building(Address, FlatNumber, Rent, NumRooms, Garden, Style):-
        house(Address, Garden, Style),
        flat(Address, FlatNumber, Rent, NumRooms).
```

For house/3 relation INDEX finds the following dependencies:

```
* house:[adr]->[styl]
* house:[adr]->[garden]
  house:[styl]->[garden]
```

At this moment the candidate for next spliting is the dependence between non prime attributes

```
[styl]->[garden]
```

and the decomposition is

```
house(Address, Garden, Style) :-
       houseStyl(Address,Style),
       houseGarden(Style, Garden).
```

6 Conclusion

We have proposed new biases for INDEX which when used makes a database schema design easier. At present the case of more than one relation/object is examined for object-oriented database schema design. The future work will concern in improving dialog between designer and system. Information about intended database schema not included (difficult to express) in extensional data should be exploited. Modelling interaction in some kind of modal logics should leads to interesting results.

Acknowledgements

We would like to thank Peter Flach who enabled us to make experiments with INDEX, as well as Pavel Brazdil who has motivated us to this work.

References

[Agar93] Agarwal R., Imielinski T, Swami A.: Mining association rules between sets of items in large databases. In Proceedings of the 1993 Int. Conf. on Management of Data SIGMOD'93, pp.207-216, 1993

[DeRae92] De Raedt L., Bruynooghe M.: Belief updating from integrity constraints and queries. Artificial Intelligence 53(2-3):291-307, February 1992.

[DeRae93] De Raedt L., Bruynooghe M.: A theory of clausal discovery. Proceedings of ILP'93, pages 25-40.

[Flach93] Flach P.: Predicate invention in inductive data engineering. Proceedings of ECML'93, LNAI 667, Springer-Verlag, 1993.

[Flener94] Flener P., Popelínský L.: On the Use of Inductive Reasoning in Program Synthesis: Prejudice and Prospects. LOPSTR'94, Pisa, Italy.

[Hugh91] Hughes J. G.: Object-Oriented Databases. Prentice Hall International (UK) Ltd, 1991.

[Kivi92] Kivinen J., Mannila H.: Approximate Dependency Inference from Relations. 4th Int. Conf. on Database Theory ICDT'92, Berlin 1992, 86-98, Lecture Notes in Comp.Sci. 646, Springer-Verlag 1992.

[Manni94] Mannila H., Toivonen H., Verkamo A.I.: Improved Methods for Finding Association Rules. Dept. of Comp. Sci. University of Helsinki TR C-1993-65, 1993

[Roll92] Rolland C., Cauvet C.: Trends and Perspectives in Conceptual Modeling. In: Loucopoulos P., Zicari R.(eds.): Conceptual Modeling, Databases and CASE: an integrated view of information systems development. John Wiley & Sons, 1992.

[Savnik93] Savnik I., Flach P.: Bottom-up Induction of Functi- onal Dependencies from Relations. In: AAAI-93 Workshop "Knowledge Discovery in Databases", 1993.

Publications in the FI MU Report Series are in general accessible
via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic