Web Reports Classification with Fuzzy Inductive Logic Programming

Jan Dědek Peter Vojtáš

***Abstract*— The abstract goes here. What you need to do is to insert your abstract here. Please try to make it less than 150 words. We suggest that you read this document carefully before you begin preparing your manuscript. IEEE does not want conference papers to have keywords so you should remove your keyword list. Also, at this time, IEEE only has some general guidelines about the format for conference papers. It is up to each individual conference to decide which format to use. In order to have a uniform look for all papers published in the proceedings, we require that every author follow the format of this sample paper. This sample paper is for latex users. Authors may use the sample paper here to produce their own papers by following the same format as this sample paper.**

I. Fuzzy inductive logic programming

In our application we are facing the challenge of induction and/or mining on several places. First we need an inductive procedure when extracting from web texts attributes of an accident.

Second we need an inductive procedure when trying to explain degree of seriousness of an accident by attributes of this accident (also called background knowledge).

Both places where induction has to be used have following requirements

- data are/can be fuzzy
- background knowledge is multirelational
- classification is fuzzy

Having in mind these requirements we chose Fuzzy inductive logic programming. To make the paper readable we present bellow short description of ILP techniques.

A. Classical ILP

In our presentation of Inductive Logic Programming (ILP) we follow the book of S. Džeroski and N. Lavrač [1].

Given is a set of examples $E = P \cup N$, where P contains positive and N negative examples, and a background knowledge B. The task is to find a hypothesis H such that

$$(\forall e \in P)(B \cup H \models e)$$

and

$$(\forall e \in N)(B \cup H \not\models e).$$

Typically, E consists of ground instances of the target predicate which has to be classified - in our case accidents. B typically consists of several predicates (relational tables) which describe properties of object which have to be classified - in our case properties of accidents. Background knowledge can contain also some rules. Hypothesis H typically consists of logic programming rules, which when added to B, explain all positive examples and no negative examples.

Main advantage of ILP is it's multirelational character, namely B can reside in several tables.

B. Fuzzy and GAP induction

In our presentation of Inductive Logic Programming (ILP) we follow the Paper of T. Horvath and P. Vojtas [2] about fuzzy inductive logic programming.

We use the approach of the fuzzy logic in narrow sense developped by J. Pavelka and P. Hajek. Formulas are of the form φ, x (φ is syntactically same as in the crisp case) are graded by a truth value $xin[0,1]$. A structure $\mathcal{M}$ consist of domain M and relations are interpreted fuzzy (we do not consider function symbols here). Evaluation $\|\varphi\|_{\mathcal{M}}$ of a formula φ uses truth functions of many valued connectives (our logic is extensional and/or truth functional). Satisfaction is defined by

$$\mathcal{M} \models_f (\varphi, x) \ iff \ \|\varphi\|_{\mathcal{M}} \geq x$$

Given is a fuzzy set of examples $\mathcal{E} : E \longrightarrow [0,1]$ and a fuzzy background knowledge $\mathcal{B} : B \longrightarrow [0,1]$. The task is to find a fuzzy hypothesis $\mathcal{H} : H \longrightarrow [0,1]$ such that

$$(\forall e, f \in dom(E))(\forall \mathcal{M})(\mathcal{M} \models_f B \cup H)$$

we have

$$E(e) > E(f) \Rightarrow \|e\|_{\mathcal{M}} \geq \|f\|_{\mathcal{M}}.$$

That is, it cannot happen that

$$E(e) > E(f) \wedge \|e\|_{\mathcal{M}} < \|f\|_{\mathcal{M}},$$

or rephrased, if E is rating e higher than f, then it can not happen in a model of $B \cup H$ that e is rated worse than f.

Typically, $\mathcal{E}$ consists of ground instances of the target predicate which are classified in truth degrees - in our case degree of seriousness of an accident. B typically consists of several fuzzy predicates (fuzzy relational tables) which describe properties of object which have to be classified - in our case fuzzy properties of accidents - degree od injury, degree of damage, ... Background knowledge can contain also some rules, so far only crisp rules are used. Hypothesis H typically consists of a fuzzy logic program, which when added to B, prevents of misclassification (better can not be declared to be worse, nevertheless can be declared as having same degree (for more detailed discussion on this definition

Jan Dědek is with the Department of software engineering, Charles University in Prague, Czech Republic (email: jan.dedek@mff.cuni.cz).

Peter Vojtáš is with the Institute of Computer Science, Academy of Sciences of the Czech Republic (email: vojtas@cs.cas.cz).

of fuzzy ILP we reffer to the paper [2])). Moreover, in practise, we use GAP - Generalized Annotated Programs, so graded formulas wil lbe sometimes understood as annotated (with crisp connectives and more complex annotation of head of rules.)

C. Translation of fuzzyILP task to several classical ILP tasks

As far as there is no implementation of fuzzy (GAP) ILP, we have to use a classical ILP system. Fortunately a fuzzy ILP task can be translated to several crisp ILP tasks (subject to some rounding and using finite set of truth values).

Assume, our fuzzy sets take values for a finite set of truth values $\{0,1\} \subseteq T \subseteq [0,1]$. For each predicate $p(x)$ in B we add an additional attribute for truth value $p(x,t)$. We construct a crisp background knowledge $\mathcal{B}_T^{mon}$ by a process called monotonization, as follows:

If $\mathcal{B}(p(x)) = t \in T$, then for all $t' \in T, t' \leq t$ we add $p(x,t') \in \mathcal{B}_T^{mon}$.

For all $t \in T$ we create a crisp example set $E_t = P_t \cup N_t$, where

$$e \in P_t \quad iff \quad \mathcal{E}(e) \geq t$$

and N_t is the rest.

For each $t \in T$ we create a crisp ILP task $E_t, \mathcal{B}_T^{mon}$ and get a hypothesis H_t guaranteeing examples of degree at least t. Note that variable boundings ib B have no boundings on truth value attribute, which was added to each predicate, and hence there are no variable boundings in H on truth value attribute. To predicate in E we did not add the additional truth value attribute

Let us assume C is the target predicate in the domain of $\mathcal{E}$. We define $\mathcal{H}$ with domain consisting of one GAP rule

$$C(y) : u(x_1, \ldots, x_m) \leftarrow B_1 : x_1 \& \ldots \& B_n : x_m,$$

here $B_1 : x_1 \& \ldots \& B_n : x_m$ is enumaration of all predicates in B.

Assume $B_1(y_1,t_1), \ldots, B_n(y_n,t_n)$ are some predicates in B (for simplicity enumerated from 1 to $n \leq m$). Then for each rule

$$R = C(y) \Leftarrow B_1(y_1,t_1), \ldots, B_n(y_n,t_n)$$

in H_t we give a constraint in definition of u as follows

$$U_R = u(x_1, \ldots, x_m) \geq t \text{ if } x_1 \geq t_1, \ldots, x_n \geq t_n.$$

Note that $x_{n+1}, \ldots, x_m$ have no restrictions.

We claim, that if all H_t were correctly learned by an crisp ILP system then for u the minimal solution of all constraints U_R for all $R \in H_t$, for all $t \in T$, the rule

$$C(y) : u(x_1, \ldots, x_m) \leftarrow B_1 : x_1 \& \ldots \& B_n : x_m,$$

is a correct solution to fuzzy ILP task given by $\mathcal{E}$ and $\mathcal{B}$. Our presentation is here a little bit simplified and we freely switch betwee fuzzy and GAP programs, which are know to be equivalent **citace Krajci, Lencses, Vojtas FSS clanek**

II. Solution

Citace tectogrammatical structure [3]
Citace ILP [4]
Citace Fuzzy ILP [2]
Citace Aleph:
A Learning Engine for Proposing Hypotheses (Aleph v5[1])

III. Results

See fig 3.

IV. Conclusion

We have presented a proposal of and experiments with a system for semantic computing of information from Czech text on Web pages. Our system relies on linguistic annotation tools from PDT [5] and the tree querying tool Netgraph [6]. Our contributions are an experimental chain of tools that enables semantic computing. In the third phase – data extraction – we formulate an inductive logic programming task over linguistically annotated data. Finally we describe transformation of these data to an ontology. Our initial experiments verified used methods and tools.

In future work we would like to test our method on another languages and compare our results with similar solutions. Our work is very close to domain dependent information extraction such as relation and event extraction. These tasks were considered as Semantic Evaluation in the first place in the MUC-6 conference 1995 [7]. Contemporary results of the ACE competition[2] show the difficulty of these problems, which are very close to ours.

Acknowledgment

This work was partially supported by Czech projects 1ET100300517, 201/09/0990 GACR and MSM-0021620838.

[1] http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/
[2] http://www.nist.gov/speech/tests/ace/

attribute name	distinct values	missing values	monotonic
size (of file)	49	0	yes
type (of accident)	3	0	no
damage	18	30	yes
dur_minutes	30	17	yes
fatalities	4	0	yes
injuries	5	0	yes
cars	5	0	yes
amateur_units	7	1	yes
profesional_units	6	1	yes
pipes	7	8	yes
lather	3	2	yes
aqualung	3	3	yes
fan	3	2	yes
ranking	14	0	yes

Fig. 1. Characteristics of accident attributes.

References

[1] S. Dzeroski and N. Lavrac, Eds., *Relational Data Mining*. Berlin: Springer, 2001. [Online]. Available: http://www-ai.ijs.si/SasoDzeroski/RDMBook/

[2] T. Horváth and P. Vojtáš, "Induction of fuzzy and annotated logic programs," *Inductive Logic Programming: 16th International Conference, ILP 2006, Santiago de Compostela, Spain, August 24-27, 2006, Revised Selected Papers*, pp. 260–274, 2007.

[3] M. Mikulová, A. Bémová, J. Hajič, E. Hajičová, J. Havelka, V. Kolářová, L. Kučová, M. Lopatková, P. Pajas, J. Panevová, M. Razímová, P. Sgall, J. Štěpánek, Z. Urešová, K. Veselá, and Z. Žabokrtský, "Annotation on the tectogrammatical level in the prague dependency treebank. annotation manual," ÚFAL MFF UK, Prague, Czech Rep., Tech. Rep. 30, 2006.

[4] S. Muggleton and L. de Raedt, "Inductive logic programming: Theory and methods," *Journal of Logic Programming*, vol. 19, pp. 629–679, 1994.

[5] J. Hajič, E. Hajičová, J. Hlaváčová, V. Klimeš, J. Mírovský, P. Pajas, J. Štěpánek, B. Vidová-Hladká, and Z. Žabokrtský, "Prague dependency treebank 2.0 cd-rom," Linguistic Data Consortium LDC2006T01, Philadelphia 2006, Philadelphia, 2006.

[6] J. Mírovský, "Netgraph: A tool for searching in prague dependency treebank 2.0," in *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT)*, J. Hajič and J. Nivre, Eds., no. 5, Prague, Czech rep., 2006, pp. 211–222.

[7] R. Grishman and B. Sundheim, "Message understanding conference-6: a brief history," in *Proceedings of the 16th conference on Computational linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 1996, pp. 466–471.

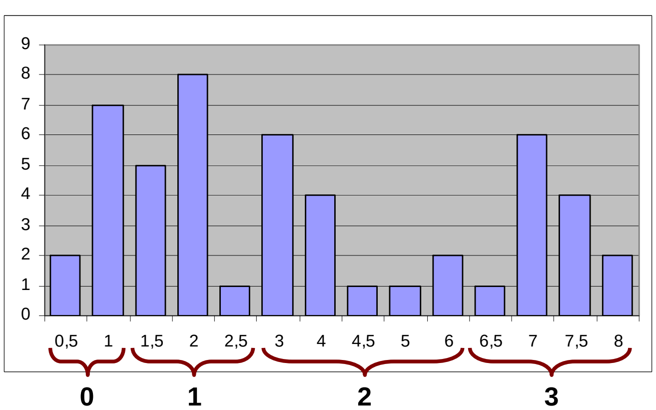

Fig. 2. Frequency histogram of accident ranking.

		Raw ILP	Fuzzy ILP
Fuzzy test set	TP:	42	57
positive: 64	FP:	7	6
negative: 36	Precision:	0,857	0,905
sum: 100	Recall:	0,656	0,891
	F-measure:	0,743	0,898
Crisp test set	TP:	12	15
positive: 25	FP:	13	10
negative: 75	Precision:	0,480	0,600
sum: 100	Recall:	0,480	0,600
	F-measure:	0,480	0,600

Fig. 3. Evaluation results

```
serious_0(ID) :- serious_atl_0(ID),
                 not(serious_atl_1(ID)),
                 not(serious_atl_2(ID)),
                 not(serious_atl_3(ID)).
```

Fig. 4. Conversion of results: monotone → crisp.

```
serious_atl_0(ID) :- serious_3(ID).
serious_atl_1(ID) :- serious_3(ID).
serious_atl_2(ID) :- serious_3(ID).
serious_atl_3(ID) :- serious_3(ID).
```

Fig. 5. Conversion of results: crisp → monotone.

```
damage_atl(ID,N) :-
        damage(ID,N), not(integer(N)).
damage_atl(ID,N) :-
        damage(ID,N2), integer(N2),
        damage(N), integer(N), N2>=N.
```

Fig. 6. Monotonization of attributes (damage → damage_atl).

```
serious_0(A):-dur_minutes(A,8).
serious_0(A):-type(A,fire), pipes(A,0).
serious_0(A):-fatalities(A,0), pipes(A,1),
              lather(A,0).
serious_1(A):-amateur_units(A,1).
serious_1(A):-amateur_units(A,0), pipes(A,2),
              aqualung(A,1).
serious_1(A):-damage(A,300000).
serious_1(A):-damage(A,unknown), type(A,fire),
              profesional_units(A,1).
serious_1(A):-dur_minutes(A,unknown),
              fatalities(A,0), cars(A,1).
serious_2(A):-lather(A,unknown).
serious_2(A):-lather(A,0), aqualung(A,1),
              fan(A,0).
serious_2(A):-amateur_units(A,2),
              profesional_units(A,2).
serious_2(A):-dur_minutes(A,unknown),
              injuries(A,2).
serious_3(A):-fatalities(A,1).
serious_3(A):-fatalities(A,2).
serious_3(A):-injuries(A,2), cars(A,2).
serious_3(A):-pipes(A,4).
```

Fig. 7. Crisp rules

```
serious_atl_0(A).
serious_atl_1(A):-injuries_atl(A,1).
serious_atl_1(A):-lather_atl(A,1).
serious_atl_1(A):-pipes_atl(A,3).
serious_atl_1(A):-dur_minutes_atl(A,unknown).
serious_atl_1(A):-size_atl(A,764),
     pipes_atl(A,1).
serious_atl_1(A):-damage_atl(A,8000),
     amateur_units_atl(A,3).
serious_atl_1(A):-type(A,car_accident).
serious_atl_1(A):-pipes_atl(A,unknown),
     randomized_order_atl(A,35).
serious_atl_2(A):-pipes_atl(A,3),
     aqualung_atl(A,1).
serious_atl_2(A):-type(A,car_accident),
     cars_atl(A,2), profesional_units_atl(A,2).
serious_atl_2(A):-injuries_atl(A,1),
     profesional_units_atl(A,3), fan_atl(A,0).
serious_atl_2(A):-type(A,other),
     aqualung_atl(A,1).
serious_atl_2(A):-dur_minutes_atl(A,59),
     pipes_atl(A,3).
serious_atl_2(A):-injuries_atl(A,2),
     cars_atl(A,2).
serious_atl_2(A):-fatalities_atl(A,1).
serious_atl_3(A):-fatalities_atl(A,1).
serious_atl_3(A):-dur_minutes_atl(A,unknown),
     pipes_atl(A,3).
```

Fig. 8. Monotonised rules