Fuzzy Classification of Web Reports with Linguistic Text Mining

Jan Dedek
Department of Software Engineering
Charles University in Prague, Czech Republic
email: jan.dedek@mff.cuni.cz

Peter Vojtas
Department of Software Engineering
Charles University in Prague, Czech Republic
peter.vojtas@mff.cuni.cz

Abstract

In this paper we present a fuzzy system which provides a fuzzy classification of textual web reports. Our approach is based on usage of third party linguistic analyzers, our previous work on web information extraction and fuzzy inductive logic programming. Main contributions are formal models and prototype implementation of the system and evaluation experiments.

1 Introduction

Our motivating example are messages of accident reports on the web. We would like to have a tool which is able to classify such message with degree of being it a serious accident.

Our solution is based first on information extraction (see emphasized information to be extracted) and second on processing this information to get fuzzy classification rules. In this paper we are concentrated on the second phase – a fuzzy classification.

Main contributions of this paper can be stated as follows:
• formal models for fuzzy classification of information form web reports
• prototype implementation of a fuzzy classification system
• experimental evaluation of the fuzzy classification system

2 Models, methods, design of the system

General schema of our system is in Fig 1. We use our previously developed web information extraction tools based on third party linguistic analyzer (the upper two dashed arrows). The classification is based on fuzzy ILP and its translation to several crisp ILP tasks. We assume that a small amount of learning data are annotated by a human.

Here we refer to our previous work [1]. A long path of tools starting with web crawling and resulting with the extracted structured information is developed in our previous work.

2.1 Fuzzy and GAP induction

In our presentation of Inductive Logic Programming (ILP) we follow [2] and [7], for fuzzy Inductive Logic Programming (fILP) we follow the paper of T. Horvath and P. Vojtas [4] about fuzzy inductive logic programming.

We use the approach of the fuzzy logic in narrow sense developed by J. Pavelka [8] and P. Hajek [3]. Formulas are of the form φ, x (φ is syntactically same as in the crisp case) are graded by a truth value $x \in [0, 1]$. 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

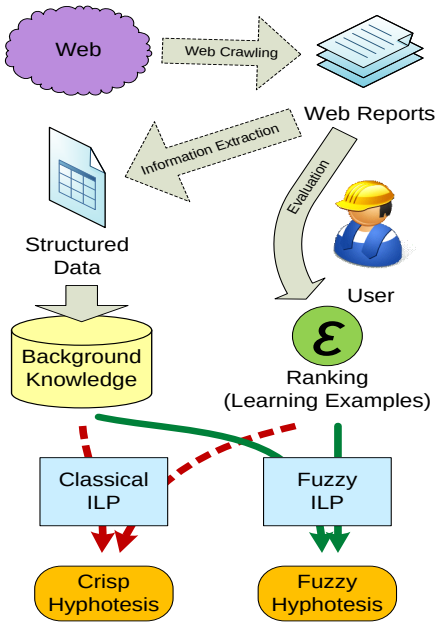

Figure 1. Schema of the whole system.

(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

$(\forall e, f \in E)(\forall \mathcal{M})(\mathcal{M} \models_f \mathcal{B} \cup \mathcal{H})$ we have $\mathcal{E}(e) > \mathcal{E}(f) \Rightarrow \|e\|_{\mathcal{M}} \geq \|f\|_{\mathcal{M}}$. That is, it cannot happen that $\mathcal{E}(e) > \mathcal{E}(f) \wedge \|e\|_{\mathcal{M}} < \|f\|_{\mathcal{M}}$, or rephrased, if $\mathcal{E}$ is rating e higher than f, then it can not happen in a model of $\mathcal{B} \cup \mathcal{H}$ 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. $\mathcal{B}$ 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 of injury, degree of damage, ... Hypothesis $\mathcal{H}$ typically consists of a fuzzy logic program, which when added to $\mathcal{B}$, 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 of fuzzy ILP we refer to the paper [4])). Moreover, in practice, we use GAP - Generalized Annotated Programs, so graded formulas will be sometimes understood as annotated (with crisp connectives and more complex annotation of head of rules). This is possible, because in [5] we have shown that (some extension of) fuzzy logic programming is equivalent to (some restriction of) generalized annotated programs.

2.2 Translation of fuzzy ILP 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 two crisp background knowledge sets $\mathcal{B}_T^{raw}$ and $\mathcal{B}_T^{mon}$ as follows:

First is a direct coding of the fuzzy value by an additional attribute:

If $\mathcal{B}(p(x)) = t \in T$, then for we add $p(x,t') \in B_T^{raw}$.

Second is obtained by a process called monotonization:

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

Also example sets are constructed in two ways.

For all $t \in T$ we create a crisp example set $E_t = P_t \cup N_t$, where $e \in P_t\ iff\ \mathcal{E}(e) = t$ and N_t is the rest of E.

For all $t \in T$ we create a crisp example set $E_{\geq t} = P_{\geq t} \cup N_{<t}$, where $e \in P_{\geq t}\ iff\ \mathcal{E}(e) \geq t$ and N_t is the rest of E.

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

Figure 2. Accident attributes.

These two translation create two ILP tasks, first is purely crisp and second can be understood (and translated back to) fILP.

First *raw ILP task* is for each $t \in T$ given by E_t and B_T^{raw}, as a result we get a set of hypothesis H_t.

Second, for each $t \in T$ we create a crisp ILP task $E_{\geq t}, B_T^{mon}$ and get a hypothesis $H_{\geq t}$ guaranteeing examples of degree at least t. Note that variable boundings in B have no boundings on truth value attribute, which was added to each predicate, and hence there are no variable boundings in $H_{\geq t}$ on truth value attribute. To predicates in E we did not add the additional truth value attribute

Now we sketch the translation of second ILP task to GAP (fILP) rules. 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 enumeration 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 between fuzzy and GAP programs, which are know to be equivalent, see [5].

3 The system prototype and our experiment

The main experiment presented in this paper leads to the seriousness classification of an accident presented on a web report, which is one of possible utilizations of the extracted semantic information. We use web reports of fire departments of several regions of the Czech Republic. These reports are written in Czech language and can be accessed through the web of General Directorate of the Fire and Rescue Service of the Czech Republic[1].

For the present experiment we have selected a collection of 50 web reports. We have identified several features presented in these reports and manually extracted corresponding values. This will be described in more detail in section 3.1. To each report we have also assigned a value of overall ranking of seriousness of presented accident, which is the target of the classification.

For the seriousness classification we have used two inductive logic approaches – Classical ILP and Fuzzy ILP (as described above). Technically the difference between the approaches consist in different setting of *ILP task*. Both can be done with a classical ILP tool. We have used "*A Learning Engine for Proposing Hypotheses*" (Aleph v5[2]), which seems very practical to us. It use quite effective method of *inverse entailment* [6] and keeps all handy features of Prolog system (supports YAP and SWI) in its background.

We have compared results of the two approaches (fuzzy and classical) and we could see that the fuzzy approach made better results than the classical one. See section 4 for details of the results.

[1] http://www.hzscr.cz

[2] http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/

Crisp learning examples

```
serious_2(id_47443). %positive

serious_0(id_47443). %negative
serious_1(id_47443). %negative
serious_3(id_47443). %negative
```

Monotonized learning examples

```
serious_atl_0(id_47443). %positive
serious_atl_1(id_47443). %positive
serious_atl_2(id_47443). %positive

serious_atl_3(id_47443). %negative
```

Figure 3. Learning examples.

damage → damage_atl

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

Figure 4. Monotonization of attributes.

3.1 Features of accidents

Figure 2 summarizes all features (or attributes) that we have obtained from accident reports. Except the attribute `type` (type of an accident – `fire`, `car_accident` and `other`) all the attributes are numerical and so monotonizable. In some cases value of some attribute is unknown. We have decided to make evidence of this and keep the values `unknown` in a knowledge base. Short explanation of each attribute follows.
`size` is a file size of text part of a report.
`damage` is an amount (in CZK – Czech Crowns) of summarized damage arisen during an accident.
`dur_minutes` is time (minutes) taken to handle an accident.
`fatalities` and `injuries` are numbers of fatalities (and injuries) taken by an accident.
`cars` is number of cars damaged during an accident (especially during car accidents).
`professional_units` and `amateur_units` are numbers of fireman units sent for an accident.
`pipes` is number of used fire pipes.
`lather`, `aqualung` and `fan` (ventilator) indicates weather these devices were used.

3.2 Seriousness ranking

Values of overall seriousness ranking attribute were stated from "general impression" from report's texts with respect to the particular attributes. Values of seriousness ranking have evolved to 14 distinct values in range form 0.5 to 8. We have divided the values into four approximately equipotent groups and learned logic rules for each group separately.

3.3 Data transformation

As already described in previous section, we have two possibilities to organize crisp ILP tasks, one with raw data and second with monotonized data. For this we have to translate extracted data.

For the construction of the E_t example set in our application we encode it in the predicate `serious_t`, For the construction of the $E_{\geq t}$ example set in our application we encode it in the predicate `serious_atl_t`, see Fig 3.

For the construction of monotonized set of background knowledge B_T^{mon} we use rules, here illustrated on predicates `damage` and `damage_atl`, see Fig 4. Here, first rules deals with `unknown` values and the second constructs the translation.

```
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),prof_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),prof_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).

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),prof_units_atl(A,2).
serious_atl_2(A):-injuries_atl(A,1),prof_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).
```

Figure 5. Crisp & monotonized hypothesis

crisp → monotone

```
serious_atl_0(ID) :- serious_2(ID).
serious_atl_1(ID) :- serious_2(ID).
serious_atl_2(ID) :- serious_2(ID).
```

monotone → crisp

```
serious_2(ID) :- serious_atl_2(ID),
                 not(serious_atl_3(ID)).
```

Figure 6. Conversion of results

4 Results

The Fig 5 summarizes sets of obtained rules from two experiments:
(1) experiments with E_t and B_T^{raw}
(2) experiments with $E_{\geq t}$, B_T^{mon}.

Evaluation of learning is depicted in the Fig 7.

We wanted to compare raw and monotonized learning tasks. As they run on different example sets we had translate results of one learning (rules with head `serious_t`) to results of second learning (rules with head `serious_atl_t`). Logic programming translation rules are depicted on Fig 6.

Then the comparison of both learnings is possible, see white areas of Fig 7.

5 Conclusion

In this paper we have presented a fuzzy system which provides a fuzzy classification of textual web reports. Our approach was based on usage of third party linguistic analyzer, our previous work on web information extraction and fuzzy inductive logic programming. Main contributions are formal models and prototype implementation of the system and evaluation experiments with the Fuzzy ILP classification method.

		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

Figure 7. Evaluation results

Experiments have shown better results of fuzzy approach. We see the difference in the fact that monotonization leads to the extension of the learning domain.

Acknowledgment

This work was partially supported by Czech projects IS-1ET100300517, GACR-201/09/H057 and MSM-0021620838.

References

[1] J. Dědek, A. Eckhardt, and P. Vojtáš. Experiments with czech linguistic data and ILP. In F. Železný and N. Lavrač, editors, *ILP 2008 - Inductive Logic Programming (Late Breaking Papers)*, pages 20–25, Prague, Czech Republic, 2008. Action M.

[2] S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer, Berlin, 2001.

[3] P. Hajek. *Metamathematics of Fuzzy Logic*. Kluwer, 1998.

[4] 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*, pages 260–274, 2007.

[5] S. Krajci, R. Lencses, and P. Vojtas. A comparison of fuzzy and annotated logic programming. *Fuzzy Sets and Systems*, 144:173–192, 2004.

[6] S. Muggleton. Inverse entailment and progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

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

[8] J. Pavelka. On fuzzy logic i, ii, iii. *Zeitschr. Math. Logik und Grundl. Math.*, 25:45–52, 119–134, 447–464, 1979.