

Fuzzy Classification of Web Reports with Mining from Texts with Linguistic Annotations

Jan Dědek^a, Peter Vojtáš^a, Marta Vomlelová^b

^a*Department of Software Engineering, Charles University,
Prague, Czech Republic*

^b*Department of Theoretical Computer Science and Mathematical Logic,
Charles University, Prague, Czech Republic*

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, publicly available prototype implementation, extensive evaluation experiments and comparison of our classifier with other alternatives like decision trees, support vector machines, neural networks etc.

Keywords: fuzzy, inductive logic programming

1. Introduction

Big amount of information on the web increases the need of automated processing. Especially machine processing and machine understanding of textual information is very difficult. Crisp methods have their limitations. In this paper we present a fuzzy system, which provides a fuzzy classification of textual web reports.

Our motivating examples are messages of accident reports on the web (Fig. 1). We would like to have a tool, which is able to classify such message by a degree of being it a serious accident.

Our solution is based on information extraction (see the emphasized pieces of information that could be extracted from a report in the Fig. 1) and on a machine learning procedure that provides rules for fuzzy classification of the reports. Our experiments deal with texts in the Czech language but our

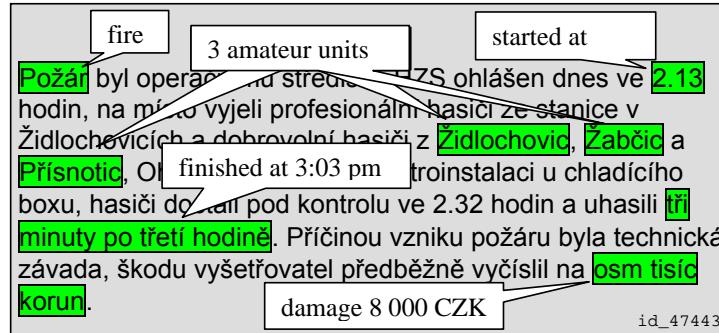


Figure 1: Example of analyzed web report.

method is general and can be used with any structured linguistic representation. In this paper we do not provide many details about the information extraction part of the solution. We concentrate on the classification part.

The main contributions of this paper can be stated as follows:

- formal models for fuzzy classification of information from web reports
- prototype implementation of a fuzzy classification system
- experimental evaluation of the fuzzy classification system.

Our paper is organized as follows: In the section 2 we introduce some works that are closely related to ours. In section 3 we develop our models and methods and the design of our system, focusing on the description of linguistic analyzers, inductive logic programming (ILP), fuzzy inductive logic programming (fILP) and several translations of a fuzzy ILP task to crisp ILP. In the section 4 we describe the system prototype implementation, the data and our experiments. In the section 5 we describe the results of the experiments and compare our methods with other well-known classifiers from machine learning. The section 6 concludes the paper.

2. Related Work

There are plenty of systems dealing with text mining and text classification. In (Reformat et al., 2008) the authors use ontology modeling to enhance text identification. The authors of (Chong et al., 2005) use preprocessed data from National Automotive Sampling System and test various soft computing

methods to modeling severity of injuries (some hybrid methods showed best performance). Methods of Information Retrieval (IR) are very numerous, with extraction mainly based on key word search and similarities. The Connection of IR and text mining techniques with web information retrieval can be found in Chapter Opinion mining in the book of Bing Liu (Liu, 2007).

3. Models, methods, design of the system

The general schema of our system is in Fig. 2. 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 is annotated by a human user.

3.1. Linguistic Analysis

In this section we will briefly describe the linguistic tools that we have used to produce linguistic annotations of texts. These tools are being developed in the Institute of Formal and Applied Linguistics in Prague, Czech Republic. They are publicly available – they have been published on a CDROM under the title PDT 2.0 (Hajič et al. (2006) – first five tools) and in (Klimeš (2006) – Tectogrammatical analysis). These tools are used as a processing chain. At the end of the chain they produce tectogrammatical (Mikulová et al., 2006) dependency trees built up from the text.

Tool 1. Segmentation and tokenization consists of tokenization (dividing the input text into words and punctuation) and segmentation (dividing a sequences of tokens into sentences).

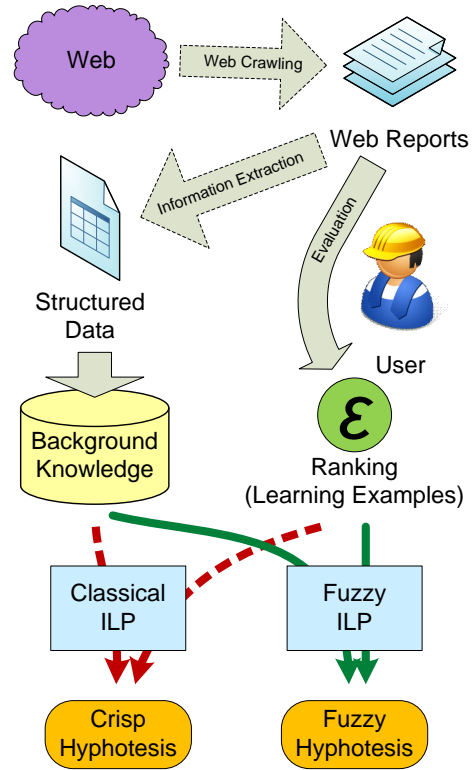


Figure 2: Schema of our system.

- Tool 2.** Morphological analysis assigns all possible lemmas and morphological tags to particular word forms (word occurrences) in the text.
- Tool 3.** Morphological tagging consists in selecting a single pair lemma-tag from all possible alternatives assigned by the morphological analyzer.
- Tool 4.** Collins’ parser – Czech adaptation. Unlike the usual approaches to the description of English syntax, the Czech syntactic descriptions are dependency-based, which means, that every edge of a syntactic tree captures the relation of dependency between a governor and its dependent node. Collins’ parser gives the most probable parse of a given input sentence.
- Tool 5.** Analytical function assignment assigns a description (analytical function – in linguistic sense) to every edge in the syntactic (dependency) tree.
- Tool 6.** Tectogrammatical analysis produces linguistic annotation at the tectogrammatical level, sometimes called “layer of deep syntax”. An example of a tectogrammatical tree can be seen on the Fig. 3. Annotation of a sentence at this layer is closer to the meaning of the sentence than its syntactic annotation and thus information captured at the tectogrammatical layer is crucial for machine understanding of natural language (Klimeš, 2006).

3.2. *Web Information Extraction*

After the content of a web resource is analyzed by the above mentioned linguistic tools, the output linguistic data is stored in the form of tectogrammatical trees. To achieve our objectives we have to extract information from this representation. Here we refer to our previous work (Dědek and Vojtáš, 2008a,b; Dědek et al., 2008). A long path of tools starting with web crawling and resulting with the extracted structured information has been developed in our previous works. In the Fig. 3 nodes of the tectogrammatical tree are decorated. A piece of information about the damage of 8000 CZK can be found there (the three nodes on the right). We have used ILP to learn rules, which are able to detect these nodes. The extraction process requires a human assistance when annotating the training data.

Note that our method is general and is not limited to Czech and can be used with any structured linguistic representation.

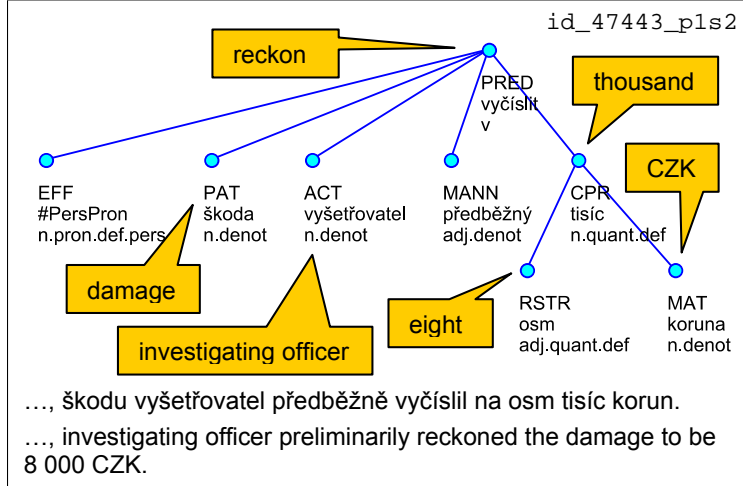


Figure 3: Example of a linguistic tree of one of analyzed sentences.

3.3. Classical ILP

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

Second (the subject of this paper) we need an inductive procedure when trying to explain the degree of seriousness of an accident by the attributes of this accident (the attributes form our background knowledge and we will often refer to them as a background knowledge). The places where induction has to be used have following requirements:

- the data and the classification is/can be fuzzy,
- in the case of tectogrammatical trees the background knowledge is multi-relational.

These requirements and our previous experience with fILP led us to chose the fILP as the machine learning inductive technique in our system. To make the paper readable we present a short description of the ILP techniques bellow.

A set of examples $E = P \cup N$ is given, where P contains positive and N negative examples, and a background knowledge B . The task of ILP 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 is to be classified - in our case accidents. B typically consists of several predicates (relational tables), which describe properties of an object, which is to be classified - in our case properties of an accident. The background knowledge can contain also some rules. A hypothesis H typically consists of logic programming rules. H added to B entails all positive examples and no negative examples. Main advantage of ILP is its multirelational character, namely B can reside in several relational tables.

3.4. Fuzzy and GAP induction

In our presentation of ILP we follow Dzeroski and Lavrac (2001) and Muggleton and de Raedt (1994), for fILP we follow the paper of T. Horváth and P. Vojtáš (Horváth and Vojtáš, 2007) about fuzzy inductive logic programming.

We use the approach of the fuzzy logic in narrow sense developed by J. Pavelka (Pavelka, 1979) and P. Hájek (Hájek, 1998). Formulas are of the form φ, x (φ is syntactically the same as in the crisp case), they are graded by a truth value $x \in [0, 1]$. A structure \mathcal{M} consist of a domain M and relations are interpreted fuzzy (we do not consider function symbols here). The evaluation $\|\varphi\|_{\mathcal{M}}$ of a formula φ uses truth functions of many valued connectives (our logic is extensional and/or truth functional). The satisfaction \models_f is defined by

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

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

$$(\forall e_1, e_2 \in E)(\forall \mathcal{M})(\mathcal{M} \models_f \mathcal{B} \cup \mathcal{H})$$

we have

$$\mathcal{E}(e_1) > \mathcal{E}(e_2) \Rightarrow \|e_1\|_{\mathcal{M}} \geq \|e_2\|_{\mathcal{M}}.$$

That is, it cannot happen that

$$\mathcal{E}(e_1) > \mathcal{E}(e_2) \wedge \|e_1\|_{\mathcal{M}} < \|e_2\|_{\mathcal{M}},$$

or rephrased: if \mathcal{E} is rating e_1 higher than e_2 , then it cannot happen that e_1 is rated worse than e_2 in a model of $\mathcal{B} \cup \mathcal{H}$.

Typically, \mathcal{E} consists of ground instances of the target predicate, which are classified in truth degrees – in our case a degree of seriousness of an accident. \mathcal{B} typically consists of several fuzzy predicates (fuzzy relational tables), which describe properties of an object, which is to be classified – in our case fuzzy properties of an accident – a degree of injury, a degree of damage, etc. A 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 the same degree — for more detailed discussion on this definition of fILP we refer to the paper (Horváth and Vojtáš, 2007)). Moreover, in practice, we use GAP – Generalized Annotated Programs, so graded formulas will be sometimes understood as annotated (with crisp connectives and with a more complex annotation of the heads of rules). This is possible, because in (Krajci et al., 2004) we have shown that (some extension of) fuzzy logic programming is equivalent to (some restriction of) generalized annotated programs.

3.5. Translation of fuzzy ILP task to several crisp ILP tasks

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

Assume that our fuzzy sets take values from a finite set of truth values $T : \{0, 1\} \subseteq T \subseteq [0, 1]$. For each predicate $p(x)$ in B we add an additional attribute $p(x, t)$ to express the truth value. We construct two crisp background knowledge sets \mathcal{B}_T^{raw} and \mathcal{B}_T^{mon} as follows:

- The first (\mathcal{B}_T^{raw}) is a direct coding of the fuzzy value by an additional attribute:
If $\mathcal{B}(p(x)) = t$, $t \in T$, then we add $p(x, t)$ to \mathcal{B}_T^{raw} .
- The second (\mathcal{B}_T^{mon}) is obtained by a process called monotonization:
If $\mathcal{B}(p(x)) = t$, $t \in T$, then for all $t' \in T$, $t' \leq t$ we add $p(x, t')$ to \mathcal{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

$$e \in P_t \text{ iff } \mathcal{E}(e) = t$$

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

$$e \in P_{\geq t} \text{ iff } \mathcal{E}(e) \geq t$$

and N_t is the rest of E .

These two translation create two ILP tasks, the first one (*raw*) is purely crisp and the second one (*mon*) can be understood as (and translated back to) fILP.

- The *raw ILP task* is given by B_T^{raw} and E_t for each $t \in T$. As a result it produces a set of hypotheses H_t .
- The *mon ILP task* is given by B_T^{mon} and $E_{\geq t}$ for each $t \in T$. As a result it produces a set of hypotheses $H_{\geq t}$ guaranteeing examples of degree of at least t .

Note that variable bindings in B have no bindings on the truth value attribute, which was added to each predicate, and hence there are no variable bindings on the truth value attribute in $H_{\geq t}$. We did not add the additional truth value attribute to the predicates in E

Now we sketch the translation of the *mon ILP task* to GAP (fILP) rules. Let us assume that C is the target predicate in the domain of \mathcal{E} . We define \mathcal{H} with a domain consisting of one GAP rule

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

here $B_1 : x_1 \& \dots \& B_n : x_m$ is the enumeration of all predicates in B .

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

$$R = C(y) \leftarrow B_1(y_1, t_1), \dots, B_n(y_n, t_n)$$

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

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

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

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

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

is a correct solution of the fILP task given by \mathcal{E} and \mathcal{B} , for all $R \in H_t$ and $t \in T$. Our presentation is here a little bit simplified and we freely switch between fuzzy and GAP programs, which are known to be equivalent, see in (Krajci et al., 2004).

4. 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 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 Republic¹.

For our 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 4.2. To each report we have also assigned a value of overall ranking of seriousness of presented accident, which is the target of the classification. The whole dataset can be downloaded from our Fuzzy ILP classifier's web page².

4.1. Experiment description

For the seriousness classification we have used two inductive logic approaches – Crisp 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 v5³), which seems very practical to

¹<http://www.hzscr.cz>

²<http://www.ksi.mff.cuni.cz/~dedek/fuzzyILP/>

³<http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/>

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

Table 1: Accident attributes.

us. It uses quite effective method of *inverse entailment* (Muggleton, 1995) and keeps all handy features of a Prolog system (supports YAP and SWI) in its background.

We have compared results of the Crisp and Fuzzy ILP approaches with other classification methods and we could see that the fuzzy approach made better results than the crisp one and also than many other methods. See section 5 for details of the results.

4.2. Features of accidents

Table 1 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 monotonic. There are cases when the value of an attribute is unknown. We have decided to make evidence of this and keep the values **unknown** in our knowledge base. A brief 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 taken to handle an accident.

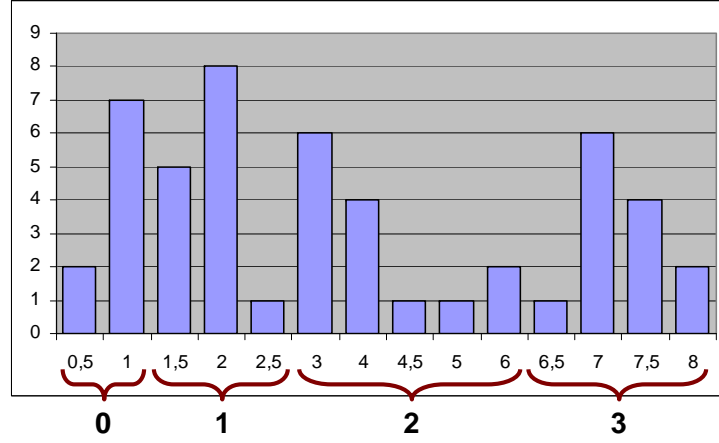


Figure 4: Frequencies of accident ranking.

- **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 whether these devices were used.

4.3. 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 from 0.5 to 8. Histogram with frequencies of all the values is on the Figure 4. We have divided the values into four approximately equipotent groups (see in the Fig. 4) and these groups determine the target class attribute in our classification task.

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
```

```
size(id_47443, 427).
type(id_47443, fire).
damage(id_47443, 8000).
dur_minutes(id_47443, 50).
fatalities(id_47443, 0).
injuries(id_47443, 0).
cars(id_47443, 0).
amateur_units(id_47443, 3).
profesional_units(id_47443, 1).
pipes(id_47443, unknown).
lather(id_47443, 0).
aqualung(id_47443, 0).
fan(id_47443, 0).
```

Figure 5: Learning examples.

Figure 6: B_T^{raw} – crisp attributes.

```
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 7: Monotonization of attributes (damage_atl \leftarrow damage).

4.4. Data transformation

To use ILP for the classification task we have to translate the input data to the Prolog-like logic representation. As already described in previous sections, the transformations for the crisp and for the fuzzy approach are different. Here we will describe the implementation details about the construction of crisp and fuzzy knowledge bases and example sets.

For the construction of the crisp example set E_t in our application we encode it in the predicate **serious_t**, for the construction of the fuzzy (or

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

serious_1(ID) :- serious_atl_1(ID),
    not(serious_atl_2(ID)),
    not(serious_atl_3(ID)).

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

serious_3(ID) :- serious_atl_3(ID).
```

Figure 8: Conversion rules for monotonized hypotheses (serious_t \leftarrow serious_atl.t).

monotonized) example set $E_{\geq t}$ we encode it in the predicate `serious_atl_t` in our application, see Fig. 5.

For the construction of crisp background knowledge B_T^{raw} we use a simple translation of the data to the predicates (illustrated on the Fig. 6).

For the construction of monotonized background knowledge B_T^{mon} we use rules, here illustrated on predicates `damage` and `damage_atl`, see Fig. 7. Here, the first rule deals with **unknown** values and the second constructs the translation. All the negations we use (Fig. 7 and Fig. 8) are the standard Prolog *negations as failure*.

```

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 9: Crisp and monotonized hypotheses.

Once we have the learning examples and the background knowledge, we can run the ILP inductive learning procedure and obtain learned rules (also called a hypothesis). According to the kind of the ILP task (crisp or monotonized) we obtain corresponding kind (crisp or monotonized) of rules (see e.g. Fig. 9). But these rules cannot be used directly to solve the classification task. There are common cases when more then one rule is applicable to a single instance, which is to be classified. So we have to select which one to use. For monotonized hypotheses we select the maximum from all the

possibilities. It is illustrated on the Fig. 8. For crisp hypotheses there is not such clear criterion, so we simply use the first applicable rule.

On the other hand in the crisp case there are many instances, which cannot be classified because there is no applicable rule. (In our experiment it was about 51% of unclassified instances, see in the next section.) It could be caused by the lack of training data, but the monotonized case does not suffer from this shortage. We can always select the bottom class and – what is probably more important – the set of positive training examples is extended by the monotonization.

5. Results

The Fig. 9 summarizes sets of obtained rules learned from our data:

- Crisp hypothesis learned from E_t and B_T^{raw} and
- Monotonized hypothesis learned from $E_{\geq t}$ and B_T^{mon} .

In both cases learning and testing examples and the background knowledge had their origin in the same data (the same accidents) but they differ in the form of the ILP task (crisp and monotonized).

We have evaluated these two methods and compared them with other learning tools used in data mining. To make the comparison clear and easy to perform, we have implemented an interface between the ILP methods and the Weka data mining software⁴. The interface makes it possible to use the ILP methods as an ordinary Weka classifier for any⁵ classification task inside the Weka software. Our implementation is publicly available. The data, source codes and an platform independent installer of the Crisp and Fuzzy ILP classifier for Weka can be downloaded from our Fuzzy ILP classifier’s web page⁶. This makes our experiment repeatable according to the SIGMOD Experimental Repeatability Requirements (Manolescu et al., 2008).

For our experiment we used the Weka Experimenter and performed an experiment in which the Crisp and Fuzzy ILP classifiers were compared with additional classifiers:

⁴<http://www.cs.waikato.ac.nz/ml/weka/>

⁵For the fuzzy ILP method, there is a requirement on the target (class) attribute: it has to be monotonizable (e.g. numeric).

⁶<http://www.ksi.mff.cuni.cz/~dedek/fuzzyILP/>

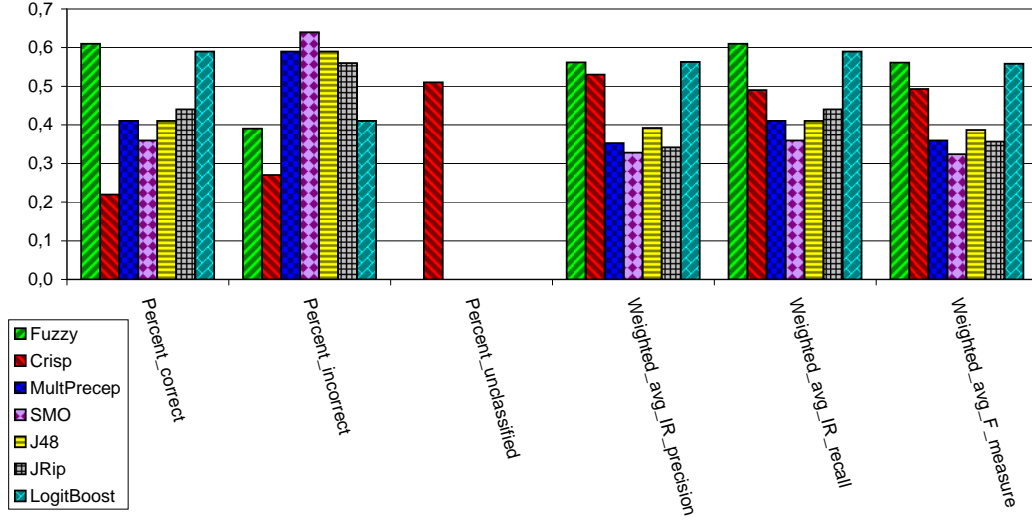


Figure 10: Evaluation of the methods – average values.

	Fuzzy	Crisp	MultPerc	SMO	J48	JRip	LBoost
Corr	0.61±.19	.22±.17●	.41±.19●	.36±.24●	.41±.22●	.44±.17●	.59±.26
Incor	.39±.19	.27±.24	.59±.19○	.64±.24○	.59±.22○	.56±.17○	.41±.26
Uncl	.00±.00	.51±.29○	.00±.00	.00±.00	.00±.00	.00±.00	.00±.00
Prec	.56±.24	.53±.37	.35±.20●	.33±.26	.39±.22	.34±.21●	.56±.28
Rec	.61±.19	.49±.32	.41±.19●	.36±.24●	.41±.22●	.44±.17●	.59±.26
F	.56±.20	.49±.33	.36±.19●	.32±.24●	.39±.21	.36±.19●	.56±.27

○, ● statistically significant improvement or degradation

Legend:

```

Fuzzy ..... czsem.ILP.FuzzyILPClassifier "
Crisp ..... czsem.ILP.CrispILPClassifier "
MultPerc .... functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a'
SMO ..... functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K
          \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"
J48 ..... trees.J48 '-C 0.25 -M 2'
JRip ..... rules.JRip '-F 3 -N 2.0 -O 2 -S 1'
LBoost ..... meta.LogitBoost '-P 100 -F 0 -R 1 -L -1.7976931348623157E308 -H 0.1 -S 1 -I 10
          -W trees.DecisionStump'

Corr ..... Percent correct
Inor ..... Percent incorrect
Uncl ..... Percent unclassified
Prec ..... Weighted avg IR precision
Rec ..... Weighted avg IR recall
F ..... Weighted avg F measure

```

Table 2: Evaluation of the methods in 2 times 10-fold cross validation.

- Multilayer Perceptron (Bishop, 1996),
- Support Vector Machine classifier SMO (Keerthi et al., 2001),
- J48 decision tree (Quinlan, 1993),
- JRip rules (Cohen, 1995) and
- Additive logistic regression LogitBoost (Friedman et al., 2000).

We have evaluated all the methods two times by 10-fold cross validation on our data (section 4.1). The obtained results (average values) are described by the graph in the Fig. 10 and in the Table 2 (with standard deviations and decorated statistically significant values).

There is no clear winner in our experiment. But we can see that on our data the Fuzzy ILP classifier proved better results than majority of the methods and the results are statistically significant in many cases. Very good results obtained also the LogitBoost.

The LogitBoost algorithm is described in Friedman et al. (2000). The basic idea of the boosting algorithm is to learn several simple models on reweighted data and combine their opinions to get the resulting classification. In our case, the simple models were decision stamps, i.e. decision trees with only one nonleaf node. In each iteration, an optimal decision stamp for each target value is learned. Then, the training dataset is reweighted in a way that records badly classified by current model are given higher weight in the next iteration. A new record is classified as follows: for each target value, predictions of its decision stamps are summed up. The target value with the highest vote wins.

We consider boosting as an enhancement of decision trees. A deep decision tree is too dependent on the randomness in the data. It has a high variance. Boosting of (tiny) trees fits an additive logistic regression model on them, i.e. uses a statistically justifiable way of building a collection of tiny trees. This seems to have low variance and keeps high expressiveness (does not increase bias dramatically).

6. 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 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 with the Fuzzy ILP classification method.

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.

Acknowledgments

This work was partially supported by Czech projects: GACR P202/10/0761, GACR-201/09/H057, GAUK 31009 and MSM-0021620838.

References

- Bishop, C. M., January 1996. Neural Networks for Pattern Recognition, 1st Edition. Oxford University Press.
URL <http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0198538642>
- Chong, M., Abraham, A., Paprzycki, M., 2005. Traffic accident analysis using machine learning paradigms. *Informatica* 29, 89–98.
- Cohen, W. W., 1995. Fast effective rule induction. In: In Proceedings of the Twelfth International Conference on Machine Learning. pp. 115–123.
URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.8204>
- Dědek, J., Eckhardt, A., Vojtáš, P., 2008. Experiments with czech linguistic data and ILP. In: Železný, F., Lavrač, N. (Eds.), ILP 2008 (Late Breaking Papers). Action M, Prague, Czech Republic, pp. 20–25.
- Dědek, J., Vojtáš, P., 2008a. Computing aggregations from linguistic web resources: a case study in czech republic sector/traffic accidents. In: Dini, C. (Ed.), Second International Conference on Advanced Engineering Computing and Applications in Sciences. IEEE Computer Society, pp. 7–12.
URL <http://www2.computer.org/portal/web/csd1/doi/10.1109/ADVCOMP.2008.17>

- Dědek, J., Vojtáš, P., 2008b. Linguistic extraction for semantic annotation. In: Badica, C., Mangioni, G., Carchiolo, V., Burdescu, D. (Eds.), 2nd International Symposium on Intelligent Distributed Computing. Vol. 162 of Studies in Computational Intelligence. Springer-Verlag, Catania, Italy, pp. 85–94.
URL <http://www.springerlink.com/content/w7213j007t416132/>
- Dzeroski, S., Lavrac, N. (Eds.), 2001. Relational Data Mining. Springer, Berlin.
URL <http://www-ai.ijs.si/SasoDzeroski/RDMBook/>
- Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: a statistical view of boosting. *Annals of statistics* 28 (2), 337–374.
- Hájek, P., 1998. Metamathematics of Fuzzy Logic. Kluwer.
- Hajič, J., Hajičová, E., Hlaváčová, J., Klimeš, V., Mírovský, J., Pajas, P., Štěpánek, J., Vidová-Hladká, B., Žabokrtský, Z., 2006. Prague dependency treebank 2.0 cd-rom. Linguistic Data Consortium LDC2006T01, Philadelphia 2006.
- Horváth, T., Vojtáš, P., 2007. Induction of fuzzy and annotated logic programs. ILP: 16th International Conference, ILP 2006, Santiago de Compostela, Spain, August 24-27, 2006, Revised Selected Papers, 260–274.
- Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., Murthy, K. R. K., 2001. Improvements to platt’s smo algorithm for svm classifier design. *Neural Computation* 13 (3), 637–649.
- Klimeš, V., 2006. Transformation-based tectogrammatical analysis of czech. In: Proceedings of the 9th International Conference, TSD 2006. No. 4188 in Lecture Notes In Computer Science. Springer-Verlag Berlin Heidelberg, pp. 135–142.
- Krajci, S., Lencses, R., Vojtas, P., 2004. A comparison of fuzzy and annotated logic programming. *Fuzzy Sets and Systems* 144, 173–192.
- Liu, B., 2007. Web Data Mining. Springer-Verlag.
URL <http://dx.doi.org/10.1007/978-3-540-37882-2>

- Manolescu, I., Afanasiev, L., Arion, A., Dittrich, J., Manegold, S., Polyzotis, N., Schnaitter, K., Senellart, P., Zoupanos, S., Shasha, D., 2008. The repeatability experiment of sigmod 2008. *SIGMOD Rec.* 37 (1), 39–45.
- Mikulová, M., Bémová, A., Hajič, J., Hajičová, E., Havelka, J., Kolářová, V., Kučová, L., Lopatková, M., Pajas, P., Panevová, J., Razímová, M., Sgall, P., Štěpánek, J., Urešová, Z., Veselá, K., Žabokrtský, Z., 2006. Annotation on the tectogrammatical level in the prague dependency treebank. annotation manual. Tech. Rep. 30, ÚFAL MFF UK, Prague, Czech Rep.
- Muggleton, S., 1995. Inverse entailment and prolog. *New Generation Computing, Special issue on Inductive Logic Programming* 13 (3-4), 245–286. URL <http://citeseer.ist.psu.edu/muggleton95inverse.html>
- Muggleton, S., de Raedt, L., 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming* 19, 629–679.
- Pavelka, J., 1979. On fuzzy logic i, ii, iii. *Zeitschr. Math. Logik und Grundl. Math.* 25, 45–52, 119–134, 447–464.
- Quinlan, J. R., 1993. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- Reformat, M., Yager, R. R., Li, Z., 2008. Ontology enhanced concept hierarchies for text identification. *Journal Semantic Web Information Systems* 4 (3), 16–43.