Fuzzy Classification of Web Reports with Linguistic Text Mining

Jan Dědek Peter Vojtáš

***Abstract*— In this paper we present a fuzzy system which increases automation of fuzzy classification of textual web reports. Our approach is 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, prototype implementation and evaluation of experiments of the whole system.**

I. Introduction

Big amount of information on the web increases the need of automated precessing. Especially textual information are hard for machine processing and understanding. Crisp methods have their limitations. In this paper we present a fuzzy system which increases automation of fuzzy classification of textual web reports.

Our motivating example is an accident message Fig 1. 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.

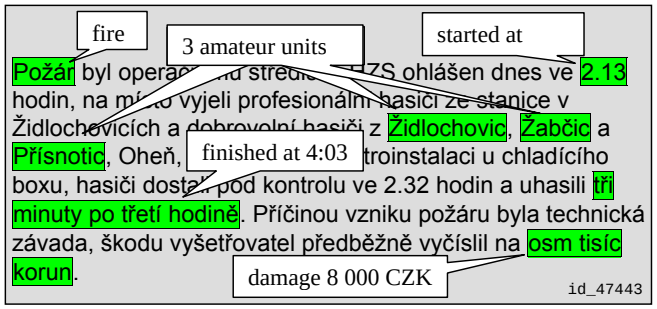

Fig. 1. Example of analyzed web report.

Main contributions of this paper can be stated as follows:

- formal models for fuzzy classification of linguistic web reports with linguistic text mining
- prototype implementation of an Internet application based on our fuzzy system
- evaluation of experiments of the whole system

Related work. There is a plenty of systems dealing with text mining and text classification, let us mention at least some. In [1] authors use ontology modeling to enhance text identification. In [2] authors 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. Connecting IR and text mining techniques with web information retrieval can be found in Chapter Opinion mining in the book of Bing Liu [3].

Our paper is organized as follows: In Chapter 2 we develop our models and methods and design the system, focusing on description of the linguistic analyzer, ILP, fuzzy ILP and several translations of fuzzy ILP to crisp ILP. In Chapter 3 we describe the system prototype and settings of our experiments, especially data preparation. In Chapter 4 we describe results of our experiments and provide comparison of used methods.

II. Models, methods, design of the system

General schema of our system is in Fig 2. We use our previously developed web information extraction tools based on third party linguistic analyser (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.

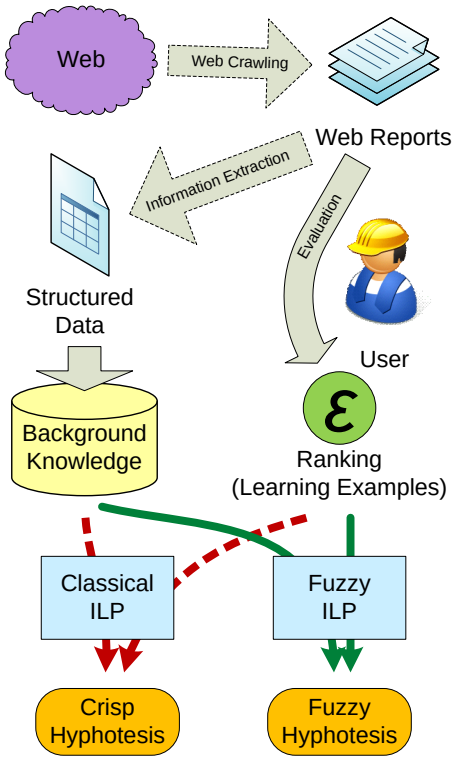

Fig. 2. Schema of presented work (text part).

Jan Dědek is a PhD student at 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, Czech Academy of Sciences and Department of Software Engineering, Charles University in Prague, Czech Republic (email: peter.vojtas@mff.cuni.cz).

A. Linguistic Analyzer

In this section we will briefly describe the linguistic tools that we have used to produce linguistic annotation 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 [4] (first five tools) and in [5] (Tectogrammatical analysis). These tools are used as a processing chain and at the end of the chain they produce tectogrammatical [6] dependency trees.

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).

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". Such a tree can be seen on the See Fig 3. Annotation of a sentence at this layer is closer to meaning of the sentence than its syntactic annotation and thus information captured at the tectogrammatical layer is crucial for machine understanding of a natural language [5].

B. Web Information Extraction

Having Web resource content analyzed by above linguistic tools, we have data stored in the form of tectogramatic trees. To achieve our objectives we have to extract information from this representation. Here we refer to our previous work [7], [8], [9]. In these papers a long path of tools starting with web crawling and resulting with the extracted structured information is described. In Fig 3 we can see nodes of tree were information about damage (8000 CZK) is located. We have used Inductive logic Programming to learn rules which are able to detect such nodes. In this paper we will concentrate on the usage of such extracted information to be able to classify content. Let us note, that this extraction process requires some human assistance when annotating data.

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

C. 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 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 (representation of tectogramatical trees)
- 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.

In our presentation of Inductive Logic Programming (ILP) we follow [10] and [11].

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.

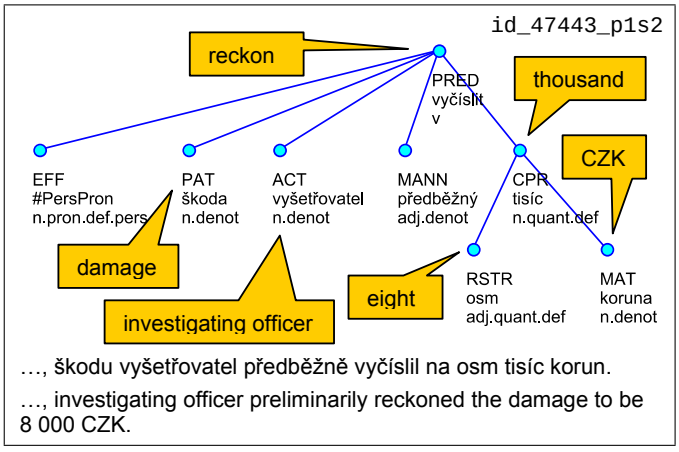

Fig. 3. Example of linguistic tree of one of analyzed sentences.

D. Fuzzy and GAP induction

In our presentation of fuzzy Inductive Logic Programming (fILP) we follow the paper of T. Horvath and P. Vojtas [12] about fuzzy inductive logic programming.

We use the approach of the fuzzy logic in narrow sense developed by J. Pavelka [13] and P. Hajek [14]. 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 (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, ... Background knowledge can contain also some rules, so far only crisp rules are used. 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 [12])). 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 [15] we have shown that (some extension of) fuzzy logic programming is equivalent to (some restriction of) generalized annotated programs.

E. 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.

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 [15].

III. The System Prototype and Setting of Our Experiment

The main experiment presented in this paper leads to the seriousness classification of an accident presented on a web report. Our long term goal is extraction of semantic information from web reports. And seriousness classification is one of possible utilization 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]. These reports are rich in information, e.g. where and when an traffic accident occurred, which units helped, how much time it took them to show up on the place of accident, how many people were injured, killed etc.

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 III-B. To each report we have also assigned a value of overall ranking of seriousness of presented accident, which is the target of the classification.

There are two objectives to do. Fist is the web information extraction, a long path starting with web crawling and resulting with the extracted structured information. Second is the seriousness classification, which utilizes the extracted information. We have made much work on the first (see e.g. [7], [8], [9]), in this paper we will concentrate on the second.

A. Experiment description

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* [16] 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 IV for details of the results.

B. Features of accidents

Figure 4 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.

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

[2] `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

Fig. 4. Characteristics of accident attributes.

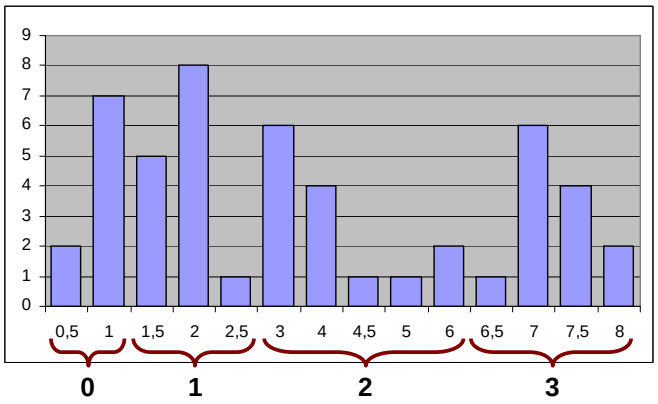

Fig. 5. Frequency histogram of accident ranking.

`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 (in 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.

C. Seriousness ranking

Values of overall seriousness ranking attribute were stated from "general impression" from report's texts with respect to the particular attributes (Fig 4). Values of seriousness ranking have evolved to 14 distinct values in range form 0.5 to 8. Histogram with frequencies of all the values is on the Figure 5.

We have divided the values into four approximately equipotent groups (shown on the Fig. 5) and learned logic rules for each group separately.

```
serious_2(id_47443). %positive

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

Fig. 6. Crisp 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
```

Fig. 7. Monotonised learning examples.

D. 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`, see Fig 6.

For the construction of the $E_{\geq t}$ example set in our application we encode it in the predicate `serious_atl_t`, see Fig 7.

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 8. Here, first rules deals with `NULL` values and the second constructs the translation.

IV. Results

A. Experiments with E_t and B_T^{raw}

Results of experiments with E_t and B_T^{raw} gave following rule set, see Fig 9.

Evaluation of learning is depicted in the red area of Fig 13.

B. Experiments $E_{\geq t}, B_T^{mon}$

Results of experiments with $E_{\geq t}, B_T^{mon}$ gave following rule set, see Fig 10.

Evaluation of learning is depicted in the green area of Fig 13.

C. Evaluation and comparison

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`).

```
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.
```

Fig. 8. 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. 9. Crisp hypothesis

```
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. 10. Monotonised hypothesis

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

Fig. 11. Conversion of results: crisp → monotone.

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

Fig. 12. Conversion of results: monotone → crisp.

Logic programming translation rules are depicted on Fig 11 and

Then the comparison of both learnings is possible, see white areas of Fig 13.

V. Conclusion

In this paper we have presented a fuzzy system which aimed to increase automation of 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, prototype implementation and evaluation of experiments of the whole system.

Experiments have shown better results of fuzzy approach.

Use of ILP was necessary for multirelational character of extracted data (relational representation of XML like form of PDT2.0 tectogramatical trees).

System need user assistance: of a skilled user in annotating data for extraction and an unskilled user for classification of a small training example set. This feature is especially important, because in Internet application, we cannot expect an unskilled user to classify a big number of examples

Acknowledgment

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

References

[1] M. Reformat, R. R. Yager, and Z. Li, "Ontology enhanced concept hierarchies for text identification," *Journal Semantic Web Information Systems*, vol. 4, no. 3, pp. 16–43, 2008.

[2] M. Chong, A. Abraham, and M. Paprzycki, "Traffic accident analysis using machine learning paradigms," *Informatica*, vol. 29, pp. 89–98, 2005.

		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. 13. Evaluation results

[3] B. Liu, *Web Data Mining*. Springer-Verlag, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-37882-2

[4] 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.

[5] V. Klimeš, "Transformation-based tectogrammatical analysis of czech," in *Proceedings of the 9th International Conference, TSD 2006*, ser. Lecture Notes In Computer Science, no. 4188. Springer-Verlag Berlin Heidelberg, 2006, pp. 135–142.

[6] 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.

[7] J. Dědek and P. Vojtáš, "Linguistic extraction for semantic annotation," in *2nd International Symposium on Intelligent Distributed Computing*, ser. Studies in Computational Intelligence, C. Badica, G. Mangioni, V. Carchiolo, and D. Burdescu, Eds., vol. 162. Catania, Italy: Springer-Verlag, 2008, pp. 85–94. [Online]. Available: http://www.springerlink.com/content/w7213j007t416132/

[8] ——, "Computing aggregations from linguistic web resources: a case study in czech republic sector/traffic accidents," in *Second International Conference on Advanced Engineering Computing and Applications in Sciences*, C. Dini, Ed. IEEE Computer Society, 2008, pp. 7–12. [Online]. Available: http://www2.computer.org/portal/web/csdl/doi/10.1109/ADVCOMP.2008.17

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

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

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

[12] 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.

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

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

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

[16] S. Muggleton, "Inverse entailment and progol," *New Generation Computing, Special issue on Inductive Logic Programming*, vol. 13, no. 3-4, pp. 245–286, 1995. [Online]. Available: http://citeseer.ist.psu.edu/muggleton95inverse.html