Towards Semantic Annotation Supported by Dependency Linguistics and ILP

Jan Dědek and Peter Vojtáš

Department of Software Engineering, Charles University,
Prague, Czech Republic
{dedek,vojtas}@ksi.mff.cuni.cz

Abstract. The abstract should summarize the contents of the paper and should contain at least 70 and at most 150 words. It should be written using the *abstract* environment.

Keywords: Semantic Annotation, Dependency Linguistics, Inductive Logic Programming, Information Extraction, Machine Learning

1 Introduction

a. The Semantic Web and Information Extraction
b. Information Extraction and Machine Learning Problem of feature selection, deep linguistic analysis is not used very often or has to be at lo level.
c. Linguistics: Phrase structure and dependency structure
d. Similarity of RDF and dependency linguistics
e. Our contributions and benefits of using the combination of tools

2 Related work

2.1 ILP users

There are many users of ILP in the linguistic an information extraction area. For example in [13] ILP was used for shallow parsing and phonotactics. In [18] ILP was used to construct good features for propositional learners like SVM to do information extraction. This approach seems to be very powerful but also much more complicated then ours. Authors of [12] summarized some basic principles of using ILP for learning from text without any linguistic preprocessing. One of the most related approaches to ours can be found in [1]. The authors use ILP for extraction of information about chemical compounds and other concepts related to global warming and they try to express the extracted information in terms of ontology. They use only part of speech analysis and named entity recognition in the preprocessing step. But their inductive procedure uses also additional domain knowledge for the extraction. This could be also employed in our approach.

2.2 Deep parsing users

As stated in [4] deep syntactic information provided by current parsers is not always reliable (e.g. for biomedical texts). But in our case deep linguistic parsing plays an essential role. There are other approaches that use deep parsing, but they often use the syntactic structure only for relation extraction and either do not use machine learning at all (the extraction rules have to be handcrafted) [20], [9], [5] or do some kind of similarity search based on the syntactic structure [8], [19] or the syntactic structure plays only very limited role in the process of feature selection for propositional learners [3].

2.3 Classical propositional learning (Information extraction - GATE)

There is also a long row of information extraction approaches that use classical propositional learners like SVN on a set of features selected form the text. We do not cite them here with one exception, which is [14] - using machine learning facilities in GATE. This is the software component (Machine Learning PR) that we have extended in our solution.

2.4 Semantic annotation - GATE

Last category of related works goes in the direction of semantics and ontologies. As far as we do not develop this topic in this paper, we just refer to the ontology features in GATE [2], which can be easily used to populate an ontology with the extracted data.

3 Exploited methods - ILP and linguistics

Exploited methods - ILP and linguistics In our solution we have exploited several tools and formalisms. These can be divided into two groups: linguistics and (inductive) logic programming. First we describe the linguistic tools and formalisms the rest will follow.

3.1 GATE

GATE[1] is probably the most spread tool for text processing. Document and annotation management, utility processing resources, JAPE grammar rules, machine learning facilities and performance evaluation tools are the most helpful features of GATE that we have used.

3.2 PDT and TectoMT

As we have started with our native language - Czech, we had to make tolls for processing Czech available in GATE. We have implemented a wrapper for the TectoMT system[2] [21] to GATE. TectoMT contains many useful tools. We

[1] `http://gate.ac.uk/`

[2] `http://ufal.mff.cuni.cz/tectomt/`

have used mainly the morphological analyzers (including POS tagger), syntactic parser and deep syntactic (tectogrammatical) parser. All the tools are based on the dependency based linguistic theory and formalism of the Prague Dependency Treebank project [10]. So far our solution does not include any coreference and discourse analysis.

Rozepsat PDT

3.3 ILP

Inductive Logic Programming (ILP) [17] is a machine learning technique based on logic programming. Given an encoding of the known background knowledge (in our case linguistic structure of all sentences) and a set of examples represented as a logical database of facts (in our case tokens annotated with the target annotation type are positive examples and the remaining tokens negative ones), an ILP system will derive a hypothesised logic program (in our case extraction rules) which entails all the positive and none of the negative examples.

From our experiments (Section 5) can be seen that ILP is capable to find complex and meaningful rules that cover the intended information.

?? large amount of training data ??

As we do not have large amount of training data, there is no problem with excessive time demands during learning and the application of the learned rules is simple and quick.

4 Implementation

Here we just briefly describe implementation of our system. The system consists of several modules, all integrated in GATE as processing resources.

4.1 TectoMT Wrapper

First is the TectoMT wrapper, which takes the text of a GATE document, sends it to TectoMT linguistic analyzers, parses the results and converts the results to the form of GATE annotations.

4.2 ILP Wrapper

After an annotator have annotated several documents with desired target annotations, machine learning takes place. This consists of two steps:

1. learning of extraction rules form the target annotations and
2. application of the extraction rules on new documents.

In both steps the linguistic analysis has to be done before and in both steps a background knowledge (a logical database of facts) is constructed form linguistic structures of documents that are to be processed. We call the process of background knowledge construction "ILP serialization"; more details are presented below **(??Sect. 1??)**. Then in the first step positive and negative examples are constructed form target annotations and the machine learning ILP inductive procedure is executed to obtain extraction rules. In the second step a Prolog system is used to check if the extraction rules entail any of target annotation candidates.

The learning examples and annotation candidates are usually constructed from all document tokens (and we did so in the present solution), but it can be optionally changed to any other textual unit, for example only numerals or tectogrammatical nodes (words with lexical meaning) can be selected. This can be done easily with help of "Machine Learning PR" (LM PR) from GATE[3]

ML PR provides an interface for exchange of features (including target class) between annotated texts and propositional learners in both directions - during learning as well as during application. We have used ML PR and developed our "ILP Wrapper" for it. The implementation was a little complicated because complex linguistic structures cannot be easily passed as propositional features, so we used the ML PR interface just for exchange the class attribute and annotation id and we access the linguistic structures directly in a document.

4.3 ILP tool

As an ILP tool we have used "A Learning Engine for Proposing Hypotheses" (Aleph v5)[4], which we consider very practical. It uses quite effective method of inverse entailment [16] and keeps all handy features of a Prolog system (we have used YAP Prolog[5]) in its background.

4.4 Root/subtree preprocessing/postprocessing, robustness

Sometimes annotations cover more then one token. This situation complicates the process of machine learning and is often called as "chunk learning". Eider we have to split the single annotation to multiple learning instances and after application we have to merge them back together, or we can change the learning task from learning annotated tokens to learning borders of annotation (start tokens and end tokens). The later approach is implemented in GATE in Batch Learning PR in the "SURROUND" mode.

We have used another approach to solve this issue. Our approach is based on syntactic structure of a sentence and we call it "root/subtree preprocessing/postprocessing". The idea is based on the observation that tokens of a multi-token annotation usually have a common parent node in the syntactic tree. So we can

---

[3] Machine Learning PR is an old GATE interface for ML and it is almost obsolete but in contrast to the new "Batch Learning PR" the LM PR is easy to extend for a new ML engine.

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

[5] `http://www.dcc.fc.up.pt/~vsc/Yap/`

1. extract the parent nodes (in dependency linguistics this node is also a token and it is usually one of the tokens inside the annotation),
2. learn extraction rules for parent nodes only and
3. put annotations to the whole subtrees of root tokens found during the application of extraction rules.

We call the first point as "root preprocessing" and the last point as "subtree postprocessing". We have successfully used this approach for the "damage" task of our evaluation corpus (See Section 5) for details.

4.5 Semantic interpretation

Information extraction can solve the task "how to get documents annotated", but as we aim on the semantic annotation, there is a second step of "semantic interpretation" that has to be done. In this step we have to interpret the annotations in terms of a standard ontology. On a very gross level this can be done easily. Thanks to GATE ontology tools we can convert all the annotations to ontology instances with a quite simple JAPE rule [6], which takes the content of an annotation and saves it as a label of the new instance or as a value of some property of a shared instance. For example in our case of fire accidents, there will be a new instance of an accident class for each document and the annotations would be attached to this instance as values of its properties. So from all annotations of the same type would be constructed instances of the same ontology class or values of the same property. This is very gross level but it still can be useful. It is similar to the GoodRelation [11] design principle of "incremental enrichment"[6]:

> ...you can still publish the data, even if not yet perfect. The Web will do the rest - new tools and people.

But of course we are not satisfied with this "gross level" of semantics and we plan to develop the semantic interpretation step further as proposed in one of our pervious works [7].

4.6 How to download

So far we do not provide our solution as a ready-made installable tool. But a middle experienced Java programmer can build it form source codes in our SVN repository[7].

[6] `http://www.ebusiness-unibw.org/wiki/Modeling_Product_Models#Recipe:_.22Incremental_Enrichment.22`
[7] Follow the instructions at `http://czsem.berlios.de/`

5 Evaluation

5.1 Dataset

We have evaluated our state of the art solution on a small dataset that we use for development. It is a collection of 50 Czech texts that are reporting on some accidents (car accidents and other actions of fire rescue services). These reports come from the web of Fire rescue service of Czech Republic[8]. The labeled corpus is publically available on the web of our project[9]. The corpus is structured such that each document represents one event (accident) and several attributes of the accident are marked in text. For the evaluation we selected two attributes of different kind. The first one is "damage" - the amount (in CZK - Czech Crowns) of the summarized damage arisen during a reported accident. The second one is "injuries", which marks mentions of people injured during the accident. These two attributes differ in two directions:

1. Injuries annotations always cover only a single token while damage usually consists of two or three tokens - one or two numerals express the amount and one extra token is for currency.
2. The complexity of the marked information (and the difficulty of the corresponding extraction task) differs slightly. While labeling of all money amounts in the corpus will result in 75% accuracy for damage annotations, in the case of injured persons mentions there are much more possibilities and indications are more spread in context.

5.2 Paum classifier as a baseline

To compare our solution with other alternatives we took a Paum propositional learner from GATE [15]. The quality of propositional learning from texts is strongly dependent on selection of right features. We obtained quite good results with features of a window of two preceding and following token lemmas and morphological tags. The precision was farther improved by adding the feature of "analytical function" from syntactic parser.

5.3 Cross validation

We used the 10-fold cross validation in the evaluation. Thanks to this technique the evaluation is very simple. After processing all the folds each document is processed with some of the ten learned models such that the particular document was not used in learning of that model, so all documents are unseen by the model applied on them. At the end we just compare the gold standard annotations with the learned ones in all documents.

[8] `http://www.hzscr.cz/hasicien/`
[9] `http://czsem.berlios.de/`

task/method	matching	missing	excessive	overlap	prec.	recall	F1.0
damage/ILP	14	0	7	6	51.85%	70.00%	59.57%
damage/ILP - the same results, lenient measures					74.07%	100.00%	85.11%
dam./ILP-roots	16	4	2	0	88.89%	80.00%	84.21%
damage/Paum	20	0	6	0	76.92%	100.00%	86.96%
injuries/ILP	15	18	11	0	57.69%	45.45%	50.85%
injuries/Paum	25	8	54	0	31.65%	75.76%	44.64%
inj./Paum-afun	24	9	38	0	38.71%	72.73%	50.53%

Table 1. Evaluation results

5.4 Results

Results of the evaluation are summarized in Table 1. We used standard information retrieval performance measures precision, recall and F_1 measure and also theirs lenient variants (overlapping annotations are counted as correct or matching).

In the first task ("damage") the methods obtained much higher scores then in the second ("injuries") as the second task was more difficult. In the first task also the root/subtree preprocessing/postprocessing improved results of ILP such that afterwards annotation borders were all placed precisely. The ILP method had better precision and worse recall but the F_1 score was very similar in both cases. Examples of learned rules In Figure 1 we present rules learned form the whole dataset.

```
[Rule 1] [Pos cover = 14 Neg cover = 0]
damage_root(A) :-
   lex_rf(B,A), has_sempos(B,'n.quant.def'), tDependency(C,B), tDependency(C,D),
   has_t_lemma(D,vy353et345ovatel).

[Rule 2] [Pos cover = 13 Neg cover = 0]
damage_root(A) :-
   lex_rf(B,A), has_functor(B,'TOWH'), tDependency(C,B), tDependency(C,D),
   has_t_lemma(D,'353koda').


[Rule 1] [Pos cover = 7 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), has_functor(B,'PAT'), has_gender(B,anim), tDependency(B,C),
   has_t_lemma(C,zran283n253).

[Rule 2] [Pos cover = 4 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), has_functor(B,'PAT'), has_gender(B,fem), tDependency(B,C),
   has_t_lemma(C,zran283n253).
```

```
[Rule 8] [Pos cover = 6 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), has_gender(B,anim), tDependency(C,B), has_t_lemma(C,zranit),
   has_negation(C,neg0).
[Rule 7] [Pos cover = 3 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), has_functor(B,'ACT'), tDependency(B,C), has_t_lemma(C,zran283n253).

[Rule 3] [Pos cover = 2 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), has_t_lemma(B,spolujezdec), tDependency(B,C), has_negation(C,neg0).

[Rule 4] [Pos cover = 2 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), has_functor(B,'ACT'), tDependency(C,B), has_t_lemma(C,utrp283t).

[Rule 5] [Pos cover = 2 Neg cover = 0]
injuries(A) :-
   lex_rf(B,A), tDependency(B,C), has_t_lemma(C,'51let253').
```

6 Conclusion and future work

a. Application? - accident seriousness ranking, environment protection, economy
b. From our experiments can be seen that ILP is capable to find complex and meaningful rules that cover the intended information. But in terms of the performance measures the results are not better than those from propositional learner.

References

1. Aitken, S.: Learning information extraction rules: An inductive logic programming approach. In: van Harmelen, F. (ed.) Proceedings of the 15th European Conference on Artificial Intelligence. IOS Press, Amsterdam (2002), http://www.aiai.ed.ac.uk/%7Estuart/AKT/ilp-ie.html
2. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to Meet New Challenges in Language Engineering. Natural Language Engineering 10(3/4), 349—373 (2004)
3. Bunescu, R., Mooney, R.: Extracting relations from text: From word sequences to dependency paths. In: Kao, A., Poteet, S.R. (eds.) Natural Language Processing and Text Mining, chap. 3, pp. 29–44. Springer London, London (2007), http://dx.doi.org/10.1007/978-1-84628-754-1_3
4. Bunescu, R.C.: Learning for Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction. Ph.D. thesis, Department of Computer Sciences, University of Texas at Austin, Artificial Intelligence Lab (2007), http://www.cs.utexas.edu/users/ml/papers/razvan-dissertation.pdf

5. Buyko, E., Faessler, E., Wermter, J., Hahn, U.: Event extraction from trimmed dependency graphs. In: BioNLP '09: Proceedings of the Workshop on BioNLP. pp. 19–27. Association for Computational Linguistics, Morristown, NJ, USA (2009)
6. Cunningham, H., Maynard, D., Tablan, V.: JAPE: a Java Annotation Patterns Engine. Tech. rep., Department of Computer Science, The University of Sheffield (2000), http://www.dcs.shef.ac.uk/intranet/research/resmes/CS0010.pdf
7. Dědek, J., Vojtáš, P.: 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. pp. 7–12. IEEE Computer Society (2008), http://www2.computer.org/portal/web/csdl/doi/10.1109/ADVCOMP.2008.17
8. Etzioni, O., Banko, M., Soderland, S., Weld, D.S.: Open information extraction from the web. Commun. ACM 51(12), 68–74 (2008), http://portal.acm.org/citation.cfm?id=1409378
9. Fundel, K., Küffner, R., Zimmer, R.: Relex—relation extraction using dependency parse trees. Bioinformatics 23(3), 365–371 (y 07)
10. Hajič, J., Hajičová, E., Hlaváčová, J., Klimeš, V., Mírovský, J., Pajas, P., Štěpánek, J., Vidová-Hladká, B., Žabokrtský, Z.: Prague dependency treebank 2.0 cd–rom. Linguistic Data Consortium LDC2006T01, Philadelphia 2006 (2006)
11. Hepp, M.: Goodrelations: An ontology for describing products and services offers on the web. In: Gangemi, A., Euzenat, J. (eds.) EKAW. Lecture Notes in Computer Science, vol. 5268, pp. 329–346. Springer (2008)
12. Junker, M., Sintek, M., Sintek, M., Rinck, M.: Learning for text categorization and information extraction with ilp. In: In Proc. Workshop on Learning Language in Logic. pp. 84–93. Springer, LNCS (1999)
13. Konstantopoulos, S.T.: Using ILP to Learn Local Linguistic Structures. Ph.D. thesis, Faculteit der Wiskunde en Natuurwetenschappen, Groningen (2003), http://dissertations.ub.rug.nl/faculties/science/2003/s.t.konstantopoulos/
14. Li, Y., Bontcheva, K., Cunningham, H.: Adapting SVM for Data Sparseness and Imbalance: A Case Study on Information Extraction. Natural Language Engineering 15(02), 241–271 (2009), http://journals.cambridge.org/repo_A45LfkBD
15. Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., Kandola, J.S.: The perceptron algorithm with uneven margins. In: ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning. pp. 379–386. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2002)
16. Muggleton, S.: Inverse entailment and progol. New Generation Computing, Special issue on Inductive Logic Programming 13(3-4), 245–286 (1995), http://citeseer.ist.psu.edu/muggleton95inverse.html
17. Muggleton, S.: Inductive logic programming. New Generation Comput. 8(4), 295– (1991)
18. Ramakrishnan, G., Joshi, S., Balakrishnan, S., Srinivasan, A.: Using ilp to construct features for information extraction from semi-structured text. In: ILP'07: Proceedings of the 17th international conference on Inductive logic programming. pp. 211–224. Springer-Verlag, Berlin, Heidelberg (2008)
19. Wang, R., Neumann, G.: Recognizing textual entailment using sentence similarity based on dependency tree skeletons. In: RTE '07: Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing. pp. 36–41. Association for Computational Linguistics, Morristown, NJ, USA (2007)
20. Yakushiji, A., Tateisi, Y., Miyao, Y., Tsujii, J.: Event extraction from biomedical papers using a full parser. Pac Symp Biocomput pp. 408–419 (2001), http://view.ncbi.nlm.nih.gov/pubmed/11262959

21. Žabokrtský, Z., Ptáček, J., Pajas, P.: TectoMT: Highly modular MT system with tectogrammatics used as transfer layer. In: ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation. pp. 167–170. Association for Computational Linguistics, Columbus, OH, USA (2008)