Towards Semantic Annotation Supported by Dependency Linguistics and ILP

Jan Dědek

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

Abstract. In this paper we present a method for semantic annotation of texts, which is based on a deep linguistic analysis (DLA) and Inductive Logic Programming (ILP). The combination of DLA and ILP have following benefits: Manual selection of learning features is not needed. The learning procedure has full available linguistic information at its disposal and it is capable to select relevant parts itself. Learned extraction rules can be easily visualized, understood and adapted by human. A description, implementation and initial evaluation of the method are the main contributions of the paper.

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

1 Introduction

Automated semantic annotation (SA) is considered to be one of the most important elements in the evolution of the Semantic Web. Besides that, SA can provide great help in the process of data and information integration and it could also be a basis for intelligent search and navigation.

In this paper we present main results and reflections of our ongoing PhD project, a method for classical and semantic information extraction and annotation of texts, which is based on a deep linguistic analysis and Inductive Logic Programming (ILP). This approach is quite novel because it directly combines deep linguistic parsing with machine learning (ML). This combination and the use of ILP as a ML engine have following benefits: Manual selection of learning features is not needed. The learning procedure has full available linguistic information at its disposal and it is capable to select relevant parts itself. Extraction rules learned by ILP can be easily visualized, understood and adapted by human.

A description, implementation and initial evaluation of the method are the main contributions of the paper.

2 Related work

There are many users of ILP in the linguistic and information extraction area. 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 the part of speech analysis and named entity recognition in the preprocessing step. But their inductive procedure uses also additional domain knowledge for the extraction. In [17] ILP was used to construct good features for propositional learners like SVM to do information extraction. It was discovered that this approach is a little bit more successful than a direct use of ILP but it is also more complicated. The later two approaches could be also employed in our solution.

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 (extraction rules have to be handcrafted) [19], [9], [4] or do some kind of similarity search based on the syntactic structure [8], [18] or the syntactic structure plays only very specific role in the process of feature selection for propositional learners [3].

There is also a long row of information extraction approaches that use classical propositional learners like SVM on a set of features manually selected from input text. We do not cite them here. We just refer to [13] – using machine learning facilities in GATE. This is the software component (Machine Learning PR) to that we have compared our solution. Our solution is also based on GATE (See next sections.)

Last category of related works goes in the direction of semantics and ontologies. Because 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. We discus this topic later in Section 4.4.

3 Exploited methods – linguistics and ILP

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] [5] is probably the most widely used tool for text processing. In our solution the capabilities of document and annotation management, utility resources for annotation processing, JAPE grammar rules [6], 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 (a language with rich morphology and free word order), we had to make tools for processing Czech

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

available in GATE. We have implemented a wrapper for the TectoMT system[2] [20] to GATE. TectoMT is a Czech project that contains many linguistic analyzers for different languages including Czech and English. We have used a majority of applicable tools from TectoMT: a tokeniser, a sentence splitter, morphological analyzers (including POS tagger), a syntactic parser and the 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.

3.3 Inductive Logic Programming

Inductive Logic Programming (ILP) [16] 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.

As an ILP tool we have used "A Learning Engine for Proposing Hypotheses" (Aleph v5)[3], which we consider very practical. It uses quite effective method of inverse entailment [15] and keeps all handy features of a Prolog system (we have used YAP Prolog[4]) in its background.

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

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 (linguistic anlysis)

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 (machine learninig)

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

1. learning of extraction rules from the target annotations and

[2] `http://ufal.mff.cuni.cz/tectomt/`
[3] `http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/`
[4] `http://www.dcc.fc.up.pt/~vsc/Yap/`

2. application of the extraction rules on new documents.

In both steps the linguistic analysis has to be done before and in both steps background knowledge (a logical database of facts) is constructed from linguistic structures of documents that are being processed. We call the process of background knowledge construction as *ILP serialization*. Although this topic is quite interesting we do not present details here because of space limitations.

After the ILP serialization is done, in the learning case, positive and negative examples are constructed from target annotations and the machine learning ILP inductive procedure is executed to obtain extraction rules.

In the application case 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 the help of *Machine Learning PR* (LM PR) from GATE[5].

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 in our solution we use the ML PR interface only for exchange of the class attribute and annotation id and we access the linguistic structures directly in a document.

4.3 Root/subtree preprocessing/postprocessing

Sometimes annotations span over more than one token. This situation complicates the process of machine learning and this situation is often called as "chunk learning". Either we have to split a 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 annotations (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 a syntactic tree. So we can

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),

[5] *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.

2. learn extraction rules for parent nodes only and
3. span annotations over 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 post-processing*. We have successfully used this technique for the 'damage' task of our evaluation corpus (See Section 5 for details.)

4.4 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 coarse level this can be done easily. Thanks to GATE ontology tools [2] we can convert all the annotations to ontology instances with a quite simple JAPE [6] rule, which takes the content of an annotation and saves it as a label of a new instance or as a value of some property of a shared instance. For example in our case of traffic and 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. Thus from all annotations of the same type, instances of the same ontology class or values of the same property would be constructed. This is very inaccurate form of semantic interpretation but still it 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 fashion of semantic interpretation and we plan to further develop the semantic interpretation step as a sophisticated "annotation → ontology" transformation process that we have proposed in one of our previous works [7].

4.5 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 from source codes in our SVN repository[7].

5 Evaluation

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

[6] `http://www.ebusiness-unibw.org/wiki/Modeling_Product_Models#Recipe:_.22Incremental_Enrichment.22`

[7] Follow the instructions at `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 – 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

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' – an amount (in CZK - Czech Crowns) of summarized damage arisen during a reported accident. The second one is 'injuries', it marks mentions of people injured during an accident. These two attributes differ. Injuries annotations always cover only a single token, while damage annotations usually consist of two or three tokens – one or two numerals express the amount and one extra token is for currency.

To compare our solution with other alternatives we took the Paum propositional learner from GATE [14]. The quality of propositional learning from texts is strongly dependent on the selection of right features. We obtained quite good results with features of a window of two preceding and two following token lemmas and morphological tags. The precision was further improved by adding the feature of *analytical function* from the syntactic parser (see the last row of Table 1).

Results of a 10-fold cross validation 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 added to the correctly matching ones, the measures are the same if no overlapping annotations are present).

In the first task ('damage') the methods obtained much higher scores then in the second ('injuries') because the second task is 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 than the Paum learner but the F_1 score was very similar in both cases.

In Figure 1 we present some examples of the rules learned from the whole dataset. The rules demonstrate a connection of a target token with other parts of a sentence through linguistic syntax structures. For example the first rule connects a root numeral (*n.quant.def*) of 'damage' with a mention of 'investigator'

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

that stated the mount. In the last rule only a positive occurrence of the verb 'injure' is allowed.

```
[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,'investigator').
[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,'damage').

[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,'injured').
[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,'injure'), has_negation(C,neg0).
```

Fig. 1. Examples of learned rules, Czech words are translated.

6 Conclusion and future work

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 a propositional learner. This is quite surprising observation because Czech is a language with free word order and we would expect much better results of the dependency approach than those of the position based approach, which was used by the propositional learner.

Our method is still missing an intelligent semantic interpretation procedure and it should be evaluated on bigger datasets (e.g. MUC, ACE, TAC, CoNLL) and other languages. So far we also do not provide a method for classical relation extraction (like e.g. in [3]). In the present solution we deal with relations implicitly. The method has to be adapted for explicit learning of relations in the form of "subject predicate object".

Our method can also provide a comparison of linguistic formalisms and tools because on the same data we could run our method using different linguistic analyzers and compare the results.

Acknowledgments
This work was partially supported by Czech projects: GACR P202/10/0761, GACR-201/09/H057, GAUK 31009 and MSM-0021620838. The author would like to thank his supervisor Peter Vojtáš for the guidance of the PhD thesis.

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)

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 (2007)
4. 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)
5. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A framework and graphical development environment for robust NLP tools and applications. In: Proceedings of the 40th Anniversary Meeting of the ACL (2002)
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)
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. 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
14. 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)
15. Muggleton, S.: Inverse entailment and progol. New Generation Computing, Special issue on Inductive Logic Programming 13(3-4), 245–286 (1995)
16. Muggleton, S.: Inductive logic programming. New Generation Computing 8(4), 295–318 (1991), http://dx.doi.org/10.1007/BF03037089
17. 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)
18. 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)
19. Yakushiji, A., Tateisi, Y., Miyao, Y., Tsujii, J.: Event extraction from biomedical papers using a full parser. Pac Symp Biocomput pp. 408–419 (2001)
20. Žabokrtský, Z., Ptáček, J., Pajas, P.: TectoMT: Highly modular MT system with tectogrammatics used as transfer layer. In: Proceedings of the 3rd Workshop on Statistical Machine Translation. pp. 167–170. ACL, Columbus, OH, USA (2008)