Steps towards Semantically Annotated Language Resources

Manfred Klenner, Fabio Rinaldi, Michael Hess

Institute of Computational Linguistics, University of Zürich,
Winterthurerstrasse 190, CH-8057 Zürich, Switzerland
{klenner,rinaldi,hess}@cl.unizh.ch

Abstract

The use of textual resources such as text corpora, tree banks, large-scale lexica etc., has become a widely accepted commitment in the field of computational linguistics. However the scope of the annotations proposed has been unbalanced towards the 'surface' level. Only recently corpora with a deeper level of annotations have started to emerge. In this paper we describe a machine learning approach aimed at learning transformational rules that would allow the (partial) generation of semantic annotations starting from syntactic annotations generated by a parser (or created manually).

1. Introduction

The use of textual resources such as text corpora, tree banks, large-scale lexica etc., has become a widely accepted commitment in the field of computational linguistics. These resources can be used for many applications, such as rapid deployment of prototype NLP systems, training of statistical NLP components, evaluation of NLP applications.

However, the availability of annotated data is not well balanced within the various subareas. Part of speech tagged corpora and tree banks are available in various languages, but semantically annotated databanks are not. There a few exceptions and of course some ongoing efforts to overcome this situation. The developers of FrameNet are augmenting the knowledge representation structures with annotated textual data, e.g. they annotate FrameNet relevant thematic roles over a corpus of real-world sentences. Consequently, the FrameNet resource is used by (Gildea and Jurafsky, 2001) to train a classifier that - given a sentence - assigns thematic roles.

We are concerned with a complementary project for German, focusing on a particular application domain: Answer Extraction (Abney et al., 2000). The format used for the semantic representation is based on a version of Minimal Recursion Semantics (Copestake et al., 1999; Hobbs, 1985) called Minimal Logical Form (MLF) (Hess, 1997). Additionally, our system is supposed to learn a larger set of semantic interpretation rules instead of assigning thematic roles only.

We are pursuing two goals: building up a large corpus of real-world sentences with their respective semantic representation and find those machine learning approach that produces the best interpretation rules for the pairings of sentences and MLFs. This way, semantic interpretation becomes independent of the syntactic representation structures produced by a particular parser. Any tree format is allowed - the various variants of phrase structure trees but also dependency trees.

2. Minimal Logical Forms

We use sentences from the Negra corpus (Skut et al., 1997), a German tree bank comprising 20.000 sentences. Fig. 1 shows an example of a Negra tree, here the subordinate clause *"der analoge Kassetten abspielen kann"* (literally translated as *"which analogue tapes play can"*). Each node has a *node number* (terminals range from 1 up to 499, nonterminals start with 500), a categorical label (e.g. VMFIN denotes a finite modal verb) and a grammatical function (e.g. SB denotes a subject, NK means noun kernel).

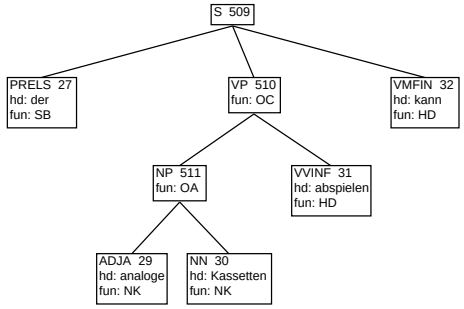

Figure 1: Negra Parse Tree of a Subordinate Clause

Up to now, about 1000 Negra sentences have been translated semi-automatically into MLFs (see section 3.). MLF are flat, existentially closed conjunctions of atomic formulae where some of the relationships are supplied with "handles", namely eventualities, objects and properties. Fig. 2 shows a graphical representation of the minimal logical form of the subordinate clause from Fig. 1.

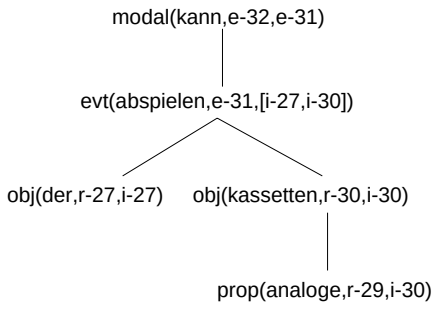

Figure 2: Minimal Logical Form of the Subordinate Clause

Here, e-32 represents an instance of a modality, namely the ability of playing some sort of tapes. The playing event is represented by e-31, its arguments are the objects i-27 and i-30. Objects and properties have attached two kind of

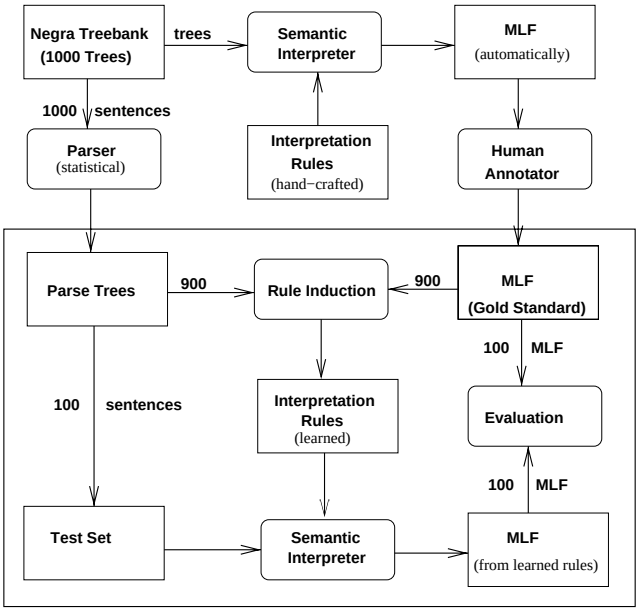

Figure 3: Annotation and Learning Design

constants: a reifier (prefix r) and an identifier (e or i). The identifier represents an instance of the denoted object, while the reifier refers to the 'property of being such an object'.

Such a flat format has been shown to be well suited for some NLP applications, like question answering (Rinaldi et al., 2002), as it allows for underspecification (which could arise as a result of partial parsing) and definition of a similarity measure (partial matching of user queries and MLF formulas).

The algorithm described in this paper is being applied in the ExtrAns system, a Question Answering system specifically targeted at technical domains (Rinaldi et al., 2004). While the original ExtrAns system has been developed for English, we are currently considering its application to German. The work described in this paper is an essential step in that direction, as it provides a way to generate in a rapid manner an initial set of semantic interpretation rules, which later can be refined manually.

3. The Semantic Annotation Process

Minimal logical forms are generated semi-automatically from Negra sentences using a (robust, rule-based) semantic interpreter. In a post processing state, the resulting minimal logical forms are scrutinized by a human annotator, and - where necessary - corrected and completed (see the upper part of Fig. 3). This way, a gold standard of minimal logical forms is produced.

```
noun = [cat:  'NounClass'] &
adj = [cat:  'ADJA'] &
lp(adj,noun)
⟹
prop(adj,noun).
```

Figure 4: Adjective Interpretation Rule

Interpretation rules consist of a condition and an action part (separated by $\Longrightarrow$, see the adjective interpretation rule from Fig. 4). Conditions are made out of *node descriptions* (e.g. `[cat:NounClass]`, where cat refers to the syntactic label of a node and 'NounClass' is a *label class* which subsumes e.g. 'NN') and two-placed predicates that serve as restrictions upon the tree structure (e.g. lp for linear precedence[1]). Additionally, there are rule variables (e.g. $noun$, adj). Every rule variable functions as a local variable within a rule, such that subsequent parts of the rule can refer with it to whatever (node number) it might evaluate. Action parts contain one or more semantic predicates (here `prop(adj,noun)`).

The adjective interpretation rule applied to the tree given in Fig. 1 would result in the binding of $noun$ to node 30 and adj to node 29. The expansion of the semantic constructor predicate `prop(adj,noun)` yields the MLF 'prop(analoge,r-29,i-30)'.

For semi-automatic annotation, there are currently about 60 (hand-crafted) rules, (partially) covering phenomena such as verb interpretation (finite, modal and infinitive constructions), noun interpretation (genitive NPs, adjective phrases, attributive PPs) coordination at the AP, NP and PP level, some rules for compound nouns and appositive constructions. No rules for negation or the relations between main and subordinate clauses exist, to name but two phenomena not captured by the current rule set. Moreover, the existing rules are neither perfect nor complete. The hand-crafted rules are meant to ease the process of semantic annotation, not to replace it. Note that we do not want to spend to much effort in writing rules for a syntactic format that is not produced by any parser: there is no parser that produces Negra trees. But even if, manual rule engineering is not our goal. We want to get independent from any particular parsing scheme by a machine learning approach that adapts to any kind of parse trees. To this end, a corpus of semantically annotated sentences is needed. Currently, 1000 sentences have been translated into MLF. They serve as a basis for our experiments with a rule learning component.

4. Rule Learning

Given a set of pairs consisting of the minimal logical form (the gold standard version) and the syntax tree of a sentence, semantic interpretation rules can be derived automatically. Note that learning *interpretation rules* instead of a full-fledged *semantic interpretation function* (mapping parse trees directly to MLFs) reduces the complexity of the learning problem: no (global) control structures need to be learned. This kind of knowledge comes with our semantic interpreter (which defines a learning bias, thus).

Before we specify our learning experiments, let's introduce one technical detail of our semantic interpreter. Identifiers and reifiers (our semantic objects) are constructed from the (syntactic) node numbers they stem from. E.g., *e-31* denotes the identifier of the verb $abspeichern$, which has node number 31 (cf. Fig. 1). Thus, every identifier in the resulting MLF can be easily traced back to its corresponding syntactic node. We call such a node the *host of the identifier*. The algorithm for rule construction vastly utilize

[1] lp is restricted to syntax nodes under the same mother

Starting from the rule anchor downwards:

- for each level of immediate dominance:
  - generate a node description for all nodes that either are hosts, or (itself) are dominating a host
  - generate a rule restriction predicate that captures this immediate dominance restriction
  - generate a rule restriction predicate that captures the linear precedence restriction
- finally, add the MLF-predicate with the generated node variables as the semantic constructor of the interpretation rule

Figure 5: Basic Rule Learner

this correspondence. The general idea is: Construct one rule for each single MLF predicate (e.g. prop(analoge,r-29,i-30)). That means: find hosts for all the identifiers of the MLF, find a least general syntactic node that dominates them all (we call such a node a *rule anchor*) and fix the structural restrictions originating from the rule anchor leading to the hosts. Structural restrictions are immediate dominance, linear precedence and the categorical label of the nodes (see Fig. 5 for a description of the algorithm).

Consider the MLF *evt(abspielen,e-31,[i-27,i-30])*. *VVINF-31*, *PRELS-27* and *NN-30* are the identifier hosts of *e-31*, *i-27* and *i-30*. *S-500* is found to be the rule anchor, it dominates all identifier hosts.

In the first iteration, the node descriptions *x1=[cat:'S']*, *x2=[cat:'PRELS]* and *x3=[cat:'VP']* are generated, as well as the rule restrictions *idom(x1,[x2,x3])* and *lp(x2,x3)*. *PRELS-27* is a identifier host, but *VP-510* must be expanded. Two node descriptions are generated: *x4=[cat:'NP']* and *x5=[cat:'VVINF']*. The rule restrictions are *idom(x3,[x4,x5])* and *lp(x4,x5)*. *VVINF-31* is a identifier host. The expansion of *NP-511* yields: *x6=[cat:'NN']* and *idom(x4,[x6])*. Finally *evt(x5,[x2,x6])* is asserted. The resulting rule is given in Fig. 6.

The question then was, how well does this basic algorithm behave: does the learned rules generalize well over unseen parse trees?

5. Experiments

We evaluated the basic construction algorithm with respect to the Negra tree bank and the LoPar parser (Schmid, 2000). The LoPar parser is a statistical parser that is based on a large, manually constructed context free grammar that was automatically converted into a probabilistic context free grammar using expectation maximization tuning over a large text corpus (22 million words). LoPar relies heavily on the X-bar schema, thus it is producing very deep syntactic structures. It uses a tag set similar to the Negra tag set, but the syntactic labels are further augmented with subcategorization information. As a result, a large number of different node labels exist, e.g.,'VPA1-1-2.nap.na' or 'VPP-past1-1-2.n.'. Both decisions, the use of X-bar and the huge tag set, pose some problems to the rule construction algorithm (see below).

```
x1=[cat:'S'] & x2=[cat:'PRELS'] &
x3=[cat:'VP'] &
idom(x1,[x2,x3]) & lp(x2,x3) &
x4=[cat:'NP'] & x5=[cat:'VVINF']
idom(x3,[x4,x5]) & lp(x4,x5) &
x6=[cat:'NN'] & idom(x4,[x6])
==>
evt(x5,[x2,x6]).
```

Figure 6: Learned Interpretation Rule

The learning algorithm does not utilize the advanced features of the semantic interpreter (e.g., cat classes like 'NounClass'), nor does it try to combine similar rules via generalization[2]. As a result, a lot of (specific) rules are generated and the (empirical) question is whether they are of any generative power. In a first experimental setting (see Fig. 3 for the general setting), the whole set of minimal logical forms generated by the interpreter were compared to the gold standard [3]. The automatic construction of interpretation rules for Negra trees achieved a precision of 0.80 and recall of 0.90 (see Fig. 7) for unseen parse trees. In the LoPar setting a similar value for precision, 0.79, but a worse value for recall, 0.70, resulted. This is a reflex of the aforementioned syntactic scheme of LoPar trees (deep trees, a lot of different syntactic label): The learned rules are to specific (recall).

	Negra	LoPar
precision	0.80	0.79
recall	0.90	0.70

Figure 7: Evaluation Results on the Whole Set of MLF

We didn't expect such a good result, given such a simple learning algorithm as presented above. So we did a closer inspection of the results and found that different semantic predicates has got different precision and recall values. Verb interpretation seemed to be the critical point, which we evaluated separately (see Fig. 8).

	Negra	LoPar
precision	0.68	0.81
recall	0.69	0.22

Figure 8: Evaluation Results for Verb Interpretation

For the Negra setting, precision is 0.68 % (instead of 80%) and recall 0.69 % (instead of 90%). This seem more realistic to us, and showed us that learning of verb interpretation rules are to be tuned. They should (in part) be more specific (since they trigger on negative examples - lowering precision). But they should also be more general (since 31% of the positive examples are not captured - worsening recall) [4] At least for recall, the situation is even more drastic

[2] However, the generated rule groups are ordered: most specific rules first

[3] We did 10-fold cross-validation

[4] One could equally well argue that our MLF-Corpus is yet to

	#Rules	Most Frequent Rules	Single Rules
Negra	132	28,25,15,14,12	90
Lopar	282	4,3,3,3,3	265

Figure 9: Rule Statistics

for the LoPar parse trees. Here a recall value of 0.22% was achieved. But this is reasonable, given the huge syntatic labels used by LoPar (remember that interpretation rules are formulated wrt. syntactic labels). This observation is exemplified by the distribution of rule frequencies as given in Fig. 9.

In the case of Negra, 132 rules were learned, where the most frequent rule was counted 28 times (i.e. there are 28 positive examples). The table also reveals the sparse data problem that is responsible for the low recall: 90 out of 132 rules are counted 1 time (i.e. they are derived from one positive example). With LoPar, 282 rules were learned, but the most frequent rule was learned from only 4 positive examples. Because of the vast syntactic labels used by LoPar, 94% (265 out of 282) of the rules stem from a single positive example.

We realized that much more effort must be spent in building up a larger semantically annotated training corpus. But also a more sophisticated learning approach must come into play. Currently, we are experimenting with a inductive logic programming approach. However, to keep the hypothesis space at a tractable size, we let the basic learning algorithm specified in Fig. 5 generate seed rules. That is, inductive learning does not start from scratch but with a skeleton of constraints that any verb interpretation rule necessarily must comprise.

6. Related Work

Learning for semantic interpretation was first done by (Zelle and Mooney, 1993). Their system, CHILL, is an approach to induce a natural language parser. The goal is to learn the actions of a shift-reduce parser expressed as a Prolog program. Their technique seems to be well suited for data base queries posed in natural language. Here the input is more or less simple, compared to the unrestricted texts that we must account for in an answer extraction scenario.

FrameNet (Baker et al., 1998) is the most prominent semantic annotation project. Among other things, a corpus of annotated sentences with frame semantics is being developed. Annotation results in the attachment of semantic (case) roles (e.g. agent, patient, ..) to substrings of a sentence. However, the arguments of the verb are not identified, since the substrings are not further segmented. Thus, no logical form is derived. Although, in principle, this is possible, it would have it's costs. Moreover, Gildea and Jurasfky (2001) report problems with the phrase shaping done by the annotators. Consider the substring segmentation [a horse who] in the FrameNet example: *I had [$_{<Theme>}$a horse who] loved going [$_{<Path>}$ on the beach and in the sea at Weston-Super-Mare] but hated puddles and tiny streams.*

small to to provide a representative training set given our basic rule constructor algorithm

7. Conclusion

Although our direct goal is to improve the answer extraction process and port our question answering system to German, the potential of the methodology proposed in this paper (to learn semantic interpretation rules from a semantically annotated corpus) are wider than that. The problems with hand crafted rules are well known. They are expensive to create and hard to maintain. But even worse, if the syntactic input structures are modified (e.g. upgrading to a new version of a statistical parser or the use of a new and better one), the whole set of rules needs to be reengineered. These limitations can be removed by a component that learns interpretation rules from a large (invariant) semantic corpus and syntactic input structures that are not restricted to a particular tree format.

There is a large number of real-world systems that could potentially benefit from large-scale semantically annotated resources (e.g. question answering systems, semantic web applications, etc.), however the costs of producing such resources (once formats for representing them have been agreed) are bound to be huge. We think that the machine learning approach presented in this paper (possibly followed by manual verification) might provide a major support in this enterprise.

8. References

Abney, Steven, Michael Collins, and Amit Singhal, 2000. Answer extraction. In Sergei Nirenburg (ed.), *Proc. 6th Applied Natural Language Processing Conference*. Seattle, WA: Morgan Kaufmann.

Baker, C. F., C. J. Fillmore, and J. B. Lowe, 1998. The berkeley framenet project. In *Proceedings of the COLING-ACL*. Montreal, Canada.

Copestake, A., D. Flickinger, I. Sag, and C. Pollard, 1999. Minimal recursion semantics. an introduction. Technical report, Stanford University.

Gildea, D. J. and D. Jurafsky, 2001. Automatic labelling of semantic role. *Computational Linguistics*, 28:245–288.

Hess, Michael, 1997. Mixed-level knowledge representations and variable-depth inference in natural language processing. *International Journal on Artificial Intelligence Tools*, 6(4):481–509.

Hobbs, Jerry R., 1985. Ontological promiscuity. In *Proc. ACL'85*. University of Chicago, Association for Computational Linguistics.

Rinaldi, Fabio, James Dowdall, Michael Hess, Diego Mollá, and Rolf Schwitter, 2002. Towards Answer Extraction: an application to Technical Domains. In *ECAI2002, European Conference on Artificial Intelligence, Lyon*.

Rinaldi, Fabio, James Dowdall, Michael Hess, Diego Mollá, and Rolf Schwitter, 2004. Question answering in terminology-rich technical domains. In Mark Maybury (ed.), *New Directions in Question Answering*. AAAI Press.

Schmid, H., 2000. Lopar: Design and implementation. Technical Report Technical Report, No 149, University of Stuttgart.

Skut, W., B. Krenn, T. Brants, and H. Uszkoreit, 1997. An annotation scheme for free worder order languages. In *5th International Conference of Applied Natural Language, Washington, USA*.

Zelle, M.J. and R.j Mooney, 1993. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence*.