Logic for Part-of-Speech Tagging and Shallow Parsing

Torbjörn Lager,
Department of Linguistics,
Uppsala University
Torbjorn.Lager@ling.uu.se

Abstract

In this paper, a purely logical approach to part-of-speech tagging and shallow parsing is explored. It has a lot in common with reductionist parsing strategies such as those employed in Constraint Grammar (Karlsson et al. 1994) and Finite-State Intersection Grammar (Koskenniemi 1990), but rules are formulated entirely in logic, and a model generation theorem prover is used for part-of-speech tagging and parsing.

1 Introduction

In this paper, a purely logical approach to part-of-speech tagging and shallow parsing is explored. It has a lot in common with *reductionist* parsing strategies such as those employed in Constraint Grammar (Karlsson et al. 1994) and Finite-State Intersection Grammar (Koskenniemi 1990), but rules are formulated entirely in first-order logic, and a model generation theorem prover is used for part-of-speech tagging and parsing. For the purpose of demonstrating the approach, a small WWW-based system has been implemented.

2 Logic

Recent years have seen several logic programming languages that extend pure Prolog to handle disjunction and various forms of negation. The particular kind of extension used in this paper have clauses of the following form:

(1) $A_1 ;...; A_k :- B_1,..., B_n, \sim D_1,..., \sim D_m$

That is, the consequent of a clause may consist of a disjunction of atomic formulas, and the antecedent may contain negated formulas. Negation here is not classical negation though, but non-monotonic, 'default' negation (like *not* or \+ in Prolog). For reasons we need not touch upon here, clauses must be *range-restricted*, which means that all variables occurring in a clause must occur in at least one of the positive body atoms $B_1,..., B_n$.

Explicit negative information can be given as follows:

(2) $:- B_1,..., B_n, \sim D_1,..., \sim D_m$

This logic has the expressive power of full first-order logic, and the non-monotonic operator only adds to that. A collection of such clauses is called a *disjunctive normal logic program*.

It is well-known that every *model* of a logic program can be represented by a set of ground atomic formulas. Furthermore, a model M is a *minimal model* of a theory T if there exists no other model of T which is included (in the set-theoretic sense) in M. Whereas a theory in the

language of pure Prolog always has exactly *one unique* minimal model, a disjunctive theory in general does not (cf. e.g. Fernández & Minker 1992).

The *minimal model state* of a theory *T* is the set of positive ground disjunctions all of whose minimal models satisfy *T*. Thus it provides a very compact representation of a set of minimal models.

3 Input

Let *yield(s)* denote the description of an input string of words *s*. For example, *yield("he can can a can")* consists of the following clauses:

(3) word(1-2,he). word(2-3,can). word(3-4,can). word(4-5,a). word(5-6,can).

Note that the forms of the words, as well as their relative positions, are encoded in these clauses.

4 Grammar

A grammar can be conceived of as a set of *constraints*, and constraints can be expressed by clauses. In the approach advocated here, *lexical entries* are encoded as clauses, often with disjunctions in their consequents, as in (4).

(4) cat(S,v(nonsg3,inf)) ; cat(S,n(sg)) ; cat(S,aux(fin)) :- word(S,can).

This clause can be paraphrased as "every instance of the word 'can' is a noun or a verb or an auxiliary".

Other constraints express negative information, and tend to look at the *local* context of the word occurrences to be disambiguated. Here's an example:

(5) :- cat(P0-P1,det), cat(P1-P2,v(_,_)).

This sentence can be paraphrased as "determiners and verbs, in that order, do not occur together".

Constraints can also be written that 'tag' occurrences of words with syntactic functions, and which perform disambiguation on that level too. Also, by means of a recursive definition of "zero or more words", words which are not strictly adjacent may be related, e.g. as follows:

(6) :- fun(P0-P1,subj), zero_or_more_words(P1-P2), fun(P2-P3,subj).

This clause expresses the constraint that a sentence may not contain more than one subject.

5 Part-of-Speech Tagging and Parsing

Let *T* be a grammar, correct but not necessarily complete, in the form of a set of constraints. Let *yield(s)* be a set of constraints deriving from the input string of words *s*. Then the notion of *grammaticality* can be defined as follows:

> **Def. 1**: An input string of words *s* is grammatical if the sentencehood of *s* is a logical consequence of *T È yield(s)*, else the grammaticality of *s* is *unknown*.

Notice that if the sentencehood of *s* cannot be demonstrated, we really don't know whether *s* is grammatical. Yet, this kind of strategy is often supplemented with the *closed world assumption*,

and with an *indirect* definition of ungrammaticality based on a negation by failure inference step, in which the negative is traded for the unknown:

> **Def. 2**: An input string of words *s* is ungrammatical if it cannot be shown to be grammatical in the sense of Def. 1.

Parsing by means of traditional generative grammars can be performed within such a framework (cf. Johnson 1994). In contrast, the *reductionist approach* can be seen as based on a notion of *ungrammaticality* defined as follows:

> **Def. 3**: An input string of words *s* is ungrammatical if *T È yield(s)* has no model, i.e. if *false* is a logical consequence of *T È yield(s)*, else the grammaticality of *s* is *unknown*.

This definition can be supplemented with the following indirect definition of *grammatical*:

> **Def. 4**: An input string of words *s* is grammatical if it cannot be shown to be ungrammatical in the sense of Def. 3.

In the traditional approach, represented by the definitions 1 and 2, everything not explicitly allowed (or 'licensed') is forbidden, whereas a reductionist grammar, based on the definitions 3 and 4, allows everything which it does not explicitly disallow.

The difference can be seen very clearly in the extreme case where we have *no grammar at all*. Then, in the traditional approach, since no input string of words can be parsed, *every* such string is deemed ungrammatical, whereas in the reductionist approach, *no* input string of words is deemed ungrammatical.

6 Output

If *false* cannot be demonstrated, then the grammar has at least one minimal model which shows that the sentence is allowed by the grammar and which encodes the linguistic properties of the input string of words. That is, a well-formed linguistic representation is any structure that satisfies the constraints.

There are two equivalent ways in which linguistic structures can be represented: as a set of minimal models, where each model represents a structure that 'survives' the constraints, or as a minimal model state (i.e. as a set of disjunctions of ground atomic sentences).

7 Examples

7.1 Part-of-Speech Tagging

The above ideas can be applied to part-of-speech tagging. A yield representing the sentence "he can can a can" as follows:

(7) `word(1-2,he). word(2-3,can). word(3-4,can). word(4-5,a). word(5-6,can).`

and the relevant lexicon entries as follows:

(8)
```
cat(S,pron(sg3)) :- word(S,he).
cat(S,det) :- word(S,a).
cat(S,v(nonsg3,inf)) ; cat(S,n(sg)) ; cat(S,aux(fin)) :- word(S,can).
```

has no less than twenty-seven minimal models. We then add the following constraints (which all seem reasonably true) to the theory:

(9)
```
:-cat(P0-P1,det), cat(P1-P2,v(_,_)).
:-cat(P0-P1,det), cat(P1-P2,aux(_)).
:-cat(P0-P1,pron(_)), cat(P1-P2,n(sg)).
:-cat(P0-P1,pron(sg3)), cat(P1-P2,v(nonsg3,_)).
:-cat(P0-P1,aux(_)), cat(P1-P2,n(sg)).
:- cat(P0-P1,aux(fin)), cat(P1-P2,aux(fin)).
```

Computing the set of minimal models, we find that only one minimal model remains.

(10)
```
{word(1-2,he). word(2-3,can). word(3-4,can). word(4-5,a). word(5-6,can).
cat(1-2,pron(sg3)). cat(2-3,aux(fin)). cat(3-4,v(nonsg3,inf)). cat(4-5,det).
cat(5-6,n(sg)). }
```

This model encodes the part-of-speech of the words in the input string. Thus the constraints serve well to discard certain minimal models and, by doing this, disambiguate the input string of words on the level of part-of-speech.

7.2 Shallow Parsing

Now, let us consider an example from syntax. Given the representation of the Swedish sentence "lisa älskar pelle" (English = "lisa loves pelle") in (11), the lexical entries in (12), the rules for assigning syntactical functions to part-of-speech in (13), and the negative information in (14), there are two minimal models, displayed in (15) and (16), respectively.

(11)
```
word(1-2,pelle). word(2-3,älskar). word(3-4,lisa).
```

(12)
```
cat(S,pn) :- word(S,pelle).
cat(S,pn) :- word(S,lisa).
cat(S,pron(nom)) :- word(S,hon).
cat(S,v) :- word(S,älskar).
```

(13)
```
fun(S,subj) ; fun(S,obj) :- cat(S,pn).
fun(S,subj) :- cat(S,pron(nom)).
fun(S,pred) :- cat(S,v).
```

(14)
```
:- fun(P0-P1,subj), zero_or_more_words(P1-P2), fun(P2-P3,subj).
:- fun(P0-P1,obj), zero_or_more_words(P1-P2), fun(P2-P3,obj).
```

(15)
```
{ word(1-2,lisa). word(2-3,älskar). word(3-4,pelle). cat(1-2,pn).
cat(2-3,v). cat(3-4,pn). fun(1-2,subj). fun(2-3,pred). fun(3-4,obj).}
```

(16)
```
{ word(1-2,lisa). word(2-3,älskar). word(3-4,pelle). cat(1-2,pn).
cat(2-3,v). cat(3-4,pn). fun(1-2,obj). fun(2-3,pred). fun(3-4,subj).}
```

In Swedish, this ambiguity is actual, since these two analyses are both possible. However, analyses assigning the same syntactic function to both names are ruled out by the description, as they should be.

These two analyses can be 'packed' into one minimal model state, as follows:

(17)
```
{ word(1-2,lisa). word(2-3,älskar). word(3-4,pelle). cat(1-2,pn).
cat(2-3,v). cat(3-4,pn). fun(1-2,obj);fun(1-2,subj). fun(1-2,obj);fun(3-4,obj).
fun(1-2,subj);fun(3-4,subj). fun(2-3,pred). fun(3-4,obj);fun(3-4,subj). }
```

Thus, a minimal model state can always provide a dense, 'underspecified' representation of analyses represented as a set of minimal models.

Finally, note that the sentence "hon älskar pelle" would only receive one analysis, since the nominative form of the personal pronoun "hon" (English = "she") disambiguates the sentence on the level of syntax.

7.3 More Complex Constraints

The constraints we have seen so far have been fairly simple, so let's consider a more complex one. In (Karlsson et al. 1994, p. 59) the following example of a constraint is given:

(18) ("<for>" =! CS (NOT -1 VFIN)(1 TO)(2 INF)(*3 VFIN))

The idea behind this constraint is that "the subjunction reading (CS) of the word-form "<for>" is correct if the preceding word is not a finite verb, if the base form of the next word is "to", if the word after this is an infinitive, and if there is a finite verb in or rightwards of positions 3."

Here's one way to write this in logic:

```
(19)  cat(P1-P2,cs) :-
          word(P1-P2,for),
          ~cat(P0-P1,v(fin)),
          cat(P2-P3,to),
          cat(P3-P4,v(inf)),
          zero_or_more_words(P4-P5),
          cat(P5-P6,v(fin)).
```

Note that this is a positive constraint. Note also the use of a negative condition, made possible by the inclusion of negation in the language, and the use of the zero-or-more-words predicate, introduced above.

Lets test this rule on a real-world example. In the *Brown corpus* (Francis & Kucera 1982), there is one (but only one!) example that satisfies the conditions of this rule.

> *Without saying so, she was really grateful; for to attend the dying was something she had never experienced ...*

Given a representation of the relevant part of this text as in (20), the lexicon entries in (21), and the constraints in (19) and (22), we are able to compute the minimal model state in (23).

```
(20)  word(1-2,for).  word(2-3,to).  word(3-4,attend).  word(4-5,the).  word(5-6,dying).
      word(6-7,was). word(7-8,something). ...

(21)  cat(S,cs) ; cat(S,prep) :- word(S,for).
      cat(S,to) ; cat(S,prep) :- word(S,to).
      cat(S,v(ing)) ; cat(S,n) :- word(S,dying).
      cat(S,det(def)) :- word(S,the).
      cat(S,v(fin)) :- word(S,was).
      cat(S,v(inf)) :- word(S,attend).
      cat(S,pron) :- word(S,something).

(22)  :- cat(P0-P1,prep), cat(P1-P2,v(inf)).
      :- cat(P0-P1,det(_)), cat(P1-P2,v(_)).
```

(23) {word(1-2,for). word(2-3,to). word(3-4,attend). word(4-5,the).
word(5-6,dying). word(6-7,was). word(7-8,something). cat(1-2,cs).
cat(2-3,to). cat(3-4,v(inf)). cat(4-5,det(def)). cat(5-6,n). cat(6-7,v(fin)). }

One final thing worth noting about this example is that there is a lot of interaction between constraints, but that the order in which they interact does not matter at all.

8 Implementation

On top of the model generation theorem prover *DisLog* (Seipel & Thöne 1994), which is capable of generating the set of minimal models or the minimal model state which satisfies a disjunctive normal logic program, I have built a simple but user-friendly World Wide Web application which tailors the DisLog prover to the task of tagging and parsing with grammars encoded in logic. The entry page is at *http://www.ling.gu.se/~lager/ICCG/iccg.html*.

9 Summary and Conclusion

In the *logic grammar* tradition, phrase structure grammars as well as unification-based extensions of phrase structure grammars are viewed in the light of the two equations "grammar = theory" and "parsing = theorem proving". This view is simple, natural and in many other ways attractive: Formal logic has an old tradition and is well-understood; logic is close to natural language, it comes with a clear notion of truth, and is declarative in the strongest possible sense. Also, properties like consistency and monotonicity have been extensively studied in logical frameworks. Indeed, if there is a *lingua franca* of knowledge representation, it ought to be formal logic.

Reductionist grammars – grammars that use mostly negative and usually local constraints to cut away ambiguities introduced by a lexical/morphological analysis phase – constitute an interesting alternative to traditional phrase structure grammars, and have been developed for several languages.

In this paper I have tried to take a logic grammar view on reductionist grammars. I hope to have shown that this view is conceptually simple and transparent, and well worth to explore. It remains to develop the basic ideas further, to scale up, to assess its practical merits and disadvantages, and to compare it with other approaches when tried on real corpora.

By trying to 'reconstruct' reductionist grammars in logic, we may begin to see why and how they work, if and when they deserve to be called 'declarative', what we could/should mean when we speak of them as 'inconsistent', 'redundant' or 'robust', etc. We may also be able to improve our understanding of how reductionist parsing relates to other kinds of methods such as parsing with phrase structure grammars, and tagging with statistical methods. For example, since phrase structure grammars can be expressed in logic, it would be interesting to elaborate on the relation of reductionist grammars to phrase structure grammars, and also, to try to combine the two.

The use of a *minimal model state* for representing ambiguity in a principled and very compact way appears to be related to work by Dymetman (1997), where an ambiguous representation of the grammatical properties of an input string of words is seen as a specialisation of the grammar

for that particular input string.[1] This indeed is exactly how a minimal model state is related to the kind of grammars explored in the present paper.

As far as scaling up is concerned, the logical approach is hopefully sufficiently similar to Constraint Grammar and Finite-state Intersection grammar to allow large portions of such grammars to be converted into logic. Moreover, since everything is expressed in logic, there may be off-the-shelf methods – such as these developed within the field of Inductive Logic Programming (Muggleton 1992) – for automatically acquiring constraints by learning from a corpus.

As regards efficiency, it remains to be shown that a logic-based reductionist grammar could ever become practical as a method for tagging or parsing. There is a well-known trade-off between expressive power and computational complexity that always threatens to make particular uses of such powerful formalisms intractable. On the other hand, there are, as far I know, no results that show that the specific kind of theories that I have in mind here actually exercise the worst case complexity of the general case. Be that as it may, since in any case, logic may still have a role to play as an analytical tool for investigating how different kinds of knowledge about language can be brought to bear on the problem of part-of-speech tagging and shallow parsing.

Acknowledgements

This work was conducted within the TagLog Project, supported by NUTEK and HSFR. I am grateful to my colleagues at Uppsala University and Göteborg University for useful discussions, and in particular to Joakim Nivre in Göteborg.

References

Dymetman, M. (1997) Charts, Interaction-free Grammars, and the Compact Representation of Ambiguity, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence NAGOYA, Aichi, Japan, August 23-29, 1997.*

Fernández, J. A. & Minker, J. (1992) Disjunctive Deductive Databases, *Proceedings of the Logic Programming and Automated Reasoning Conference.*

Francis, W. & Kucera, H. (1982) *Frequency Analysis of English Usage.* Houghton Mifflin.

Johnson, M. (1994) Two Ways of Formalizing Grammars. *Linguistics and Philosophy* 17(3).

Karlsson, F., Voutilainen, A., Heikkilä, J. & Antilla, A. (1995) *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text.* Mouton De Gruyter.

[1] Dymetman's approach is in turn related to methods proposed by researchers working in the LFG framework. The connection to LFG and thus indirectly to the work of Dymetman was kindly suggested to me by an anonymous reviewer of a draft version of the present paper.

Koskenniemi, K. (1990) Finite-state parsing and disambiguation. In Hans Karlgren (ed.) *COLING-90. Papers presented at the 13th International Conference on Computational Linguistics*, Vol. 2, Helsinki, Finland.

Muggleton, S. (Ed.). (1992) *Inductive Logic Programming*, Academic Press.

Seipel, D. & Thöne, H. (1994) DISLOG - A System for in Reasoning in Disjunctive Deductive Databases, In: Antoni Olivé (Ed.): *Proceedings of the Fifth International Workshop on the Deductive Approach to Information Systems and Databases*. September 19-21, 1994, Aiguablava, Costa Brava.