

# Enhancing deep parsing with input annotation from shallow tools

Rebecca Dridan

IRTG  
Universität des Saarlandes

DELPH-IN Summit  
Berlin, August 2007

# Outline

## Introduction

## Initial Experiments

- POS tags

- Named Entity tagging

## Restricting Lexical Items

- Method

- Results

## Issues



# Introduction

Research Focus Integrating deep and shallow processing tools.

# Introduction

**Research Focus** Integrating deep and shallow processing tools.

**Current Goal** To increase robustness and improve efficiency of the PET parser by annotating the input with information from shallow processing tools.

# Introduction

**Research Focus** Integrating deep and shallow processing tools.

**Current Goal** To increase robustness and improve efficiency of the PET parser by annotating the input with information from shallow processing tools.

## Questions

- What type of information is useful for input annotation?
- In what situations (languages, domains etc) is that information useful?
- How can the information be utilised?
- Can we measure the effects and tradeoffs we make?



# Introduction

## Known issues with deep parsing

- speed

# Introduction

## Known issues with deep parsing

- speed
- lack of robustness: no parses found
  - lack of lexical coverage
  - lack of constructional coverage

# Introduction

## Known issues with deep parsing

- speed
- lack of robustness: no parses found
  - lack of lexical coverage
  - lack of constructional coverage
- ambiguity: too many parses found
  - real vs spurious ambiguity
  - lexical ambiguity
  - syntactic ambiguity

# Introduction

## Known issues with deep parsing

- speed
- lack of robustness: no parses found
  - lack of lexical coverage
  - lack of constructional coverage
- ambiguity: too many parses found
  - real vs spurious ambiguity
  - lexical ambiguity
  - syntactic ambiguity

Can shallow annotation help?

# Outline

Introduction

**Initial Experiments**

POS tags

Named Entity tagging

Restricting Lexical Items

Method

Results

Issues

# Initial Experiments

PET XML input format allows:

- one or more POS tags (with probabilities) per token
- input of named entity elements that refer to one or more tokens
- modification of feature structure values for input items

## Initial Experiments

PET XML input format allows:

- one or more POS tags (with probabilities) per token
- input of named entity elements that refer to one or more tokens
- modification of feature structure values for input items

Experiments:

- added POS tags from TnT and TreeTagger POS taggers
- added Named Entity elements from SProUT



# POS tags

Adding POS tags could have two possible benefits:



# POS tags

Adding POS tags could have two possible benefits:

- Increase lexical coverage by adding generic lexical entries (`generic_noun`, `generic_verb`) where a word is not in the lexicon.



## POS tags

Adding POS tags could have two possible benefits:

- Increase lexical coverage by adding generic lexical entries (`generic_noun`, `generic_verb`) where a word is not in the lexicon.
- Reduce ambiguity (and improve parser efficiency) by filtering out lexical items with the wrong part of speech.



## POS tags

Adding POS tags could have two possible benefits:

- Increase lexical coverage by adding generic lexical entries (`generic_noun`, `generic_verb`) where a word is not in the lexicon.
- Reduce ambiguity (and improve parser efficiency) by filtering out lexical items with the wrong part of speech.

Results of adding pos elements to w tokens

- Increased lexical coverage
- *however* POS tags are completely ignored if the word is found in the lexicon, even with the wrong part of speech.

## POS tags (second attempt)

Add POS by setting the category to `noun` or `verb` where appropriate, using feature structure modification.

```
<w id="W3" cstart="16" cend="22">  
  <surface>lodges</surface>  
  <pos prio="0.980454" tag="NNS" />  
  <typeinfo id="TT3" baseform="yes">  
    <stem>lodge</stem>  
    <fsmod path="SYNSEM.LOCAL.CAT.HEAD" value="noun" />  
  </typeinfo>  
</w>
```

## POS tags (second attempt)

Add POS by setting the category to `noun` or `verb` where appropriate, using feature structure modification.

```
<w id="W3" cstart="16" cend="22">  
  <surface>lodges</surface>  
  <pos prio="0.980454" tag="NNS" />  
  <typeinfo id="TT3" baseform="yes">  
    <stem>lodge</stem>  
    <fsmode path="SYNSEM.LOCAL.CAT.HEAD" value="noun" />  
  </typeinfo>  
</w>
```

### Results

Parser warns about the conflicting feature structure attributes, *but proceeds to add the conflicting lexical item from the lexicon anyway.*



## Named Entity tagging

Named Entity tagging provides more semantic than syntactic information, but could still be useful for increasing lexical coverage.

## Named Entity tagging

Named Entity tagging provides more semantic than syntactic information, but could still be useful for increasing lexical coverage.

### Results

- Increased lexical coverage but not to the same extent as POS tagging.
- No improvement over POS results when combined with POS tagged input.
- When used with `constant=no` (used in parallel with the single tokens), increased ambiguity.
- When used with `constant=yes` (used instead of the single tokens) replaced correct parses with incorrect parses.

## Named Entity tagging

Named Entity tagging provides more semantic than syntactic information, but could still be useful for increasing lexical coverage.

### Results

- Increased lexical coverage but not to the same extent as POS tagging.
- No improvement over POS results when combined with POS tagged input.
- When used with `constant=no` (used in parallel with the single tokens), increased ambiguity.
- When used with `constant=yes` (used instead of the single tokens) replaced correct parses with incorrect parses.

Essentially the same as POS tagging only proper nouns, but errors have more effect.



# Outline

Introduction

Initial Experiments

POS tags

Named Entity tagging

**Restricting Lexical Items**

**Method**

**Results**

Issues

# Restricting Lexical Items

**Aim:** restrict the lexical entries the parser considers.

**Why?**

- speed
- use less resources - more complex sentences may now parse
- fewer (unlikely) parses produced - disambiguation easier?

## Restricting Lexical Items

**Aim:** restrict the lexical entries the parser considers.

**Why?**

- speed
- use less resources - more complex sentences may now parse
- fewer (unlikely) parses produced - disambiguation easier?

**How?**

```
<w id="W3" cstart="16" cend="22" >  
  <surface>lodges</surface>  
  <pos prio="0.980454" tag="NNS" />  
  <typeinfo id="TT3" baseform="yes" >  
    <stem>lodge</stem>  
    <fsmode path="SYNSEM.LOCAL.CAT.HEAD" value="noun" />  
  </typeinfo>  
</w>
```

## Restricting Lexical Items

**Aim:** restrict the lexical entries the parser considers.

**Why?**

- speed
- use less resources - more complex sentences may now parse
- fewer (unlikely) parses produced - disambiguation easier?

**How?**

```
<w id="W3" cstart="16" cend="22" >  
  <surface>lodges</surface>  
  <pos prio="0.980454" tag="NNS" />  
  <typeinfo id="TT3" baseform="yes" >  
    <stem>lodge</stem>  
    <fsmode path="SYNSEM.LOCAL.CAT.HEAD" value="noun" />  
  </typeinfo>  
</w>
```

# Method

## Input

- TreeTagger used to determine POS tags, probabilities and lemmas.
- typeinfo element created for verbs and nouns with probability greater than or equal to *threshold*.



# Method

## Input

- TreeTagger used to determine POS tags, probabilities and lemmas.
- typeinfo element created for verbs and nouns with probability greater than or equal to *threshold*.

## Changes to PET

Minor change to reject rather than warn about lexical entries that can not unify with the input typeinfo element.

Other unexpected changes required - stemming, morphological analysis . . .



## Results

Threshold	Coverage (%)	Average Readings	% of Sentences Affected
Unrestricted	70.9	287.96	-
0.60	65.1	247.85	31.9
0.90	67.9	230.52	23.7
0.98	68.7	238.45	17.7
1.00	71.1	282.79	6.0



## Results

Threshold	Coverage (%)	Accuracy (%)	Precision (%)
Unrestricted	70.9	61.6	86.9
0.60	65.1	53.4	82.1
0.90	67.9	56.9	83.8
0.98	68.7	58.2	84.6
1.00	71.1	62.1	87.3

# Outline

Introduction

Initial Experiments

POS tags

Named Entity tagging

Restricting Lexical Items

Method

Results

Issues



# Issues

Presently PET input format allows specification of “all or nothing”.



## Issues

Presently PET input format allows specification of “all or nothing”.

### Proposal

- Extension to the input formats (XML and/or SMAF) to allow specification of whatever information is available.

This could be:

- POS
- gender
- tense
- morphology
- ...
- PET still does morphological analysis (if required), and lexicon lookup.
- Underspecified lexical items are created if a token isn't in the lexicon.

# Issues

## Evaluation

### *Coverage*

- standard metric in the community and various profiling systems are designed to enable coverage evaluation
- by the nature of the processing, no result is much more likely than only getting incorrect parses

# Issues

## Evaluation

### *Coverage*

- standard metric in the community and various profiling systems are designed to enable coverage evaluation
- by the nature of the processing, no result is much more likely than only getting incorrect parses

*however*



# Issues

## Evaluation

### Coverage

- standard metric in the community and various profiling systems are designed to enable coverage evaluation
- by the nature of the processing, no result is much more likely than only getting incorrect parses

### *however*

- shallow tools have the potential to 'dilute' accuracy of deep parsers  $\Rightarrow$  accuracy measures become much more important
- accuracy information is available in `[incr tsdb()]` but not easily



# Issues

## Evaluation

### *Coverage*

- standard metric in the community and various profiling systems are designed to enable coverage evaluation
- by the nature of the processing, no result is much more likely than only getting incorrect parses

### *however*

- shallow tools have the potential to 'dilute' accuracy of deep parsers  $\Rightarrow$  accuracy measures become much more important
- accuracy information is available in `[incr tsdb()]` but not easily

**Proposal:** Extension to `[incr tsdb()]` so that parse accuracy can be easily compared after treebanking.



# Thank You!

## Current Flow

Tokens in input are either:

- a token with type info using a class as stem (NE style) and then PET makes an input item and does nothing to it  
OR
- a token with type info using an actual stem, and potentially other feature structure modifications and PET looks up the stem in the lexicon to get lex entries, doing no morphological analysis  
OR
- a straight word token, maybe with POS info and then PET will do its morphological analysis and then get lex entries out of the lexicon based on the morph base.  
In this case, POS information is ignored if a lex entry is found in the lexicon.



## Proposal

- Input tokens can be complex structures.
- Any feature structure value can be specified.
  - SYNSEM.LOCAL.CAT.HEAD
  - gender, number, case?
  - oscar+ ?
- Morphological analysis is (can be) still done on the word form.
- Lexicon lookup is restricted by the specified feature structure values.
- If no compatible entry can be found, create a generic item (ala generic\_noun) with all the specified features.