An Experiment in Shallow Robust Parsing

Krasimir Angelov

Chalmers University of Technology

March 9, 2011

Can we apply GF to open-domain text?

Current state

- Parsing for small controlled languages
- Language Generation from formal representation

Long-term goal

- Robustness for out of coverage content
- Statistical disambiguation

The Concrete Experiment

Resources

- English Resource Grammar
- Oxford Advanced Learner's Dictionary ($\sim$ 40000 words)
- Simplified Named Entities recognizer

Grammar Evaluation and **Probability Training**

- Sections 2–21 from PennTreebank

Note: In a previous work the parser was optimized to work efficiently for **wide coverage grammars** and **large lexicons**

English Resource Grammar

A distinguishing feature of GF is that grammars can be reused as software libraries.

English Resource Grammar

- Part of the resource library
- Originally not intended for parsing
- Wide coverage English grammar
- Still there are missing syntactic constructions
- Highly ambiguous

Named Entities Recognizer

The Named Entity recognizer uses this simple rules:

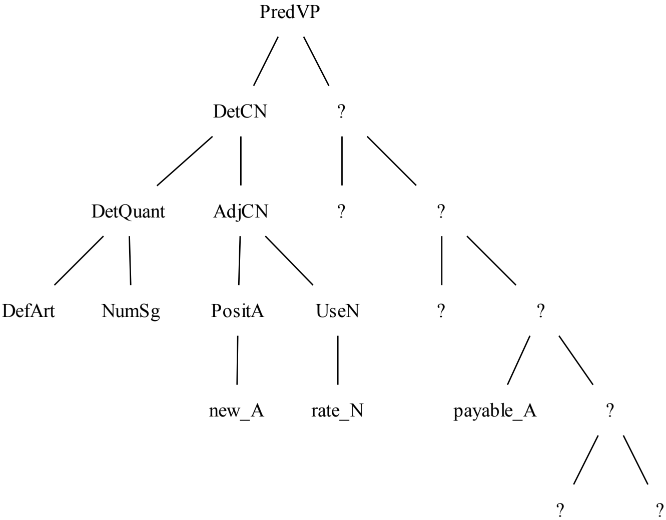
- A sequence of words starting with a capital letter is a name
- '-' and '&' are permitted between the words of a name

The recognizer cannot be implemented directly in GF:

- Some time ago I developed API which lets the user extend the parser with custom code
- For the experiment I implemented the NE recognizer as a Haskell procedure

Example Tree

For out of grammar sentences we want partial trees:



Robustness

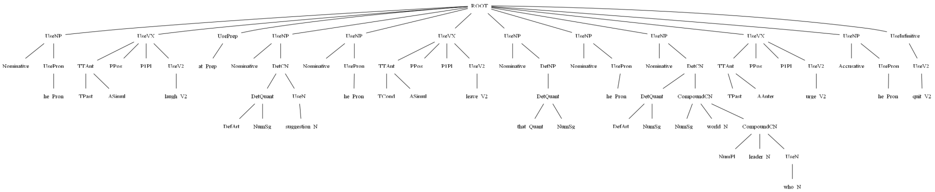
As a first approximation we use chunk parsing which is more robust than full parsing.

We scan the sentence for:

- basic **noun phrases** i.e. without PP atachement
- **verb phrases** without the object
- **prepositions** - mark the PP atachements

The high-level syntactic constructions are excluded to reduce the ambiguities and increase the robustness

Robustness


ROOT
UseNP UseVX UsePrep UseNP UseNP UseVX UseNP UseNP UseNP UseVX UseNP UseInfinitive
Nominative UsePron TTAnt PPos P1Pl UseV2 at_Prep Nominative DetCN Nominative UsePron TTAnt PPos P1Pl UseV2 Nominative DetNP Nominative UsePron Nominative DetCN TTAnt PPos P1Pl UseV2 Accusative UsePron UseV2
he_Pron TPast ASimul laugh_V2 DetQuant UseN he_Pron TCond ASimul leave_V2 DetQuant he_Pron DetQuant CompoundCN TPast AAnter urge_V2 he_Pron quit_V2
DefArt NumSg suggestion_N that_Quant NumSg DefArt NumSg NumSg world_N CompoundCN
NumPl leader_N UseN
who_N

Initial Evaluation

We collected all basic noun phrases from PennTreebank (2–21) and tried to parse them:

Success

- 75% of the phrases were parsed

Failure

- Incomplete patterns for Named Entities (ex: the United States)
- Syntax for dates?
- Missing words

The coverage of the verb phrases is not evaluated yet because in PennTreebank they include the object as well.

How to increase the coverage?

- Better coverage for the syntax of Named Entities. Perhaps something like **ANNIE in GATE**, or **NERC in KIM** can be reimplemented in GF.
- Someone have to do the **grammar for dates**.
- **Improve the lexicon** by collecting list of words from PennTreebank. The parser can just **guess the POS tag** for the unknown words (easy).

Observation

Close to the surface the disambiguation is easy:

- The parsed chunks are mostly unambiguous. There are at most 4-5 trees for one phrase.
- The ambiguities can be fixed by either:
  - assigning simple priorities (probabilities) to the different functions
  - constraining the part of speech tags

Example: “other corporate insider”

AdjCN (PositA other_A) (AdjCN (PositA corporate_A) (UseN insider_N))

CompoundCN NumSg other_N (AdjCN (PositA corporate_A) (UseN insider_N))

How to Convert Penn Treebank to GF?

We need a treebank consistent with GF for further experiments. Penn Treebank is not always consistent with GF but:

- we can use the GF parser to parse chunks of the sentence
- tags NN,NNS,VBN,VB,VBG,VBZ,VBD,VB and JJ match well with the corresponding categories in GF so we can use this for disambiguation.
- we can recover some parts of the high-level syntax by looking at the annotations

After some transformations we have **69%** of the treebank in GF abstract trees