T-(538|725)-MALV, Natural Language Processing
Partial parsing

Hrafn Loftsson[1] Hannes Högni Vilhjálmsson[1]

[1]School of Computer Science, Reykjavik University

October 2008

Outline

Outline

Full/deep parsing (í. full þáttun)

Aim

- To perform deep analysis.
- To construct "a complete" parse tree.
- Various linguistic theories have been used, e.g.:
  - CFG – **C**ontext **F**ree **G**rammar
  - PCFG – **P**robabilistic **C**ontext **F**ree **G**rammar (Collins 1996; Charniak 1997)
  - HPSG – **H**ead-Driven **P**hrase **S**tructure **G**rammar (Pollard and Sag 1994).
  - LFG – **L**exical **F**unctional **G**rammar (Kaplan 1989).
  - DG – **D**ependency **G**rammar – (Tesnière 1966)

Full parsing

The problem

- Difficult and time consuming to build a parser with large grammatical coverage.
- The solution set can grow exponentially.
  - Because often the parser tries to build all possible parse trees for a given sentence.
- The parser might also reject a correct analysis of a part of the sentence.
  - In the case when it does not fit into a global parse.

Outline

Partial/shallow parsing (í. hlutaþáttun)

Aim

- Analyse sentence parts (or chunks) without building a complete parse tree.
- "to recover syntactic information efficiently and reliably from unrestricted text, by sacrificing completeness and depth of analysis" (Abney 1996).

When is partial parsing suitable?

- When full analysis is not necessary.
  - For example, in information retrieval or information extraction.
- When efficiency is of prime importance.
- When the quality of the input is poor.

Partial/shallow parsing (í. hlutaþáttun)

Aim

- Analyse sentence parts (or chunks) without building a complete parse tree.
- "to recover syntactic information efficiently and reliably from unrestricted text, by sacrificing completeness and depth of analysis" (Abney 1996).

When is partial parsing suitable?

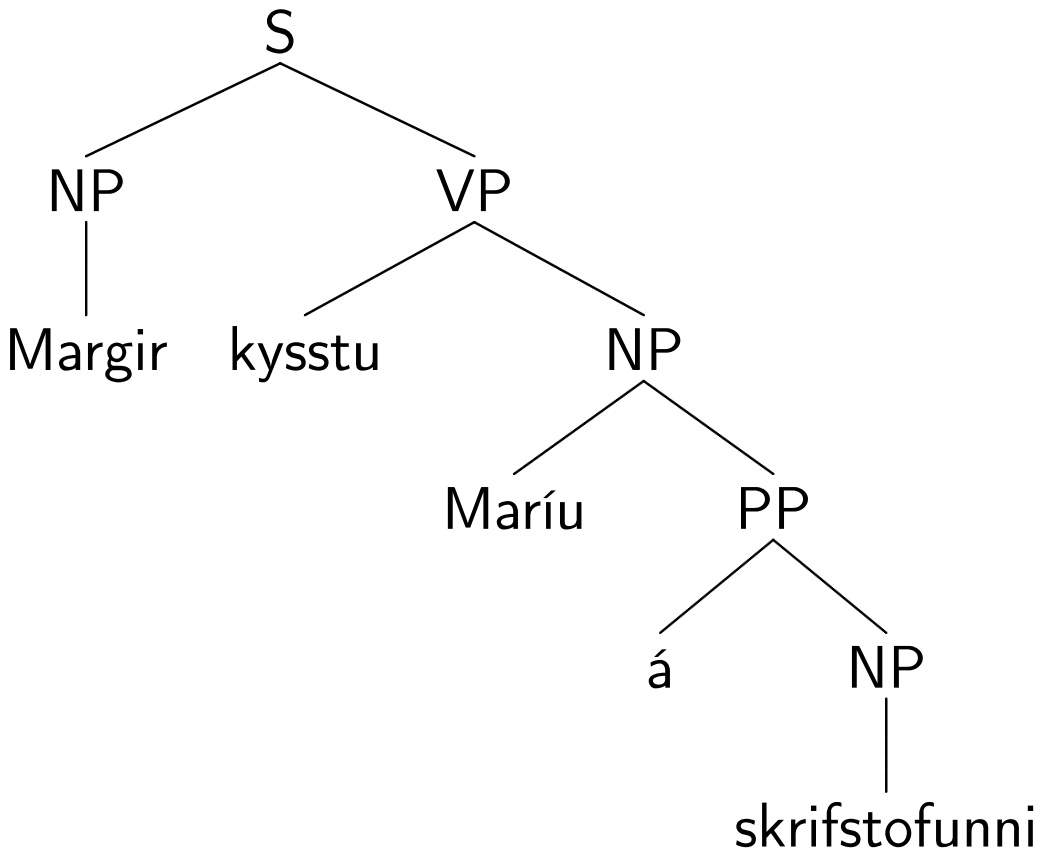
- When full analysis is not necessary.
  - For example, in information retrieval or information extraction.
- When efficiency is of prime importance.
- When the quality of the input is poor.

Full parsing vs. Partial parsing

- *Margir kysstu Maríu á skrifstofunni (Many kissed Maria at office-the)*
  - (Höskuldur Þráinsson (1999). Íslensk setningafræði)
- **Full parsing**:
  - [S [NP Margir] [VP kysstu [NP Maríu [PP á [NP skrifstofunni]]]]]
  - [S [NP Margir] [VP kysstu [NP Maríu]] [PP á [NP skrifstofunni]]]
- **Partial parsing**:
  - [NP Margir] [VP kysstu] [NP Maríu] [PP á [NP skrifstofunni]]

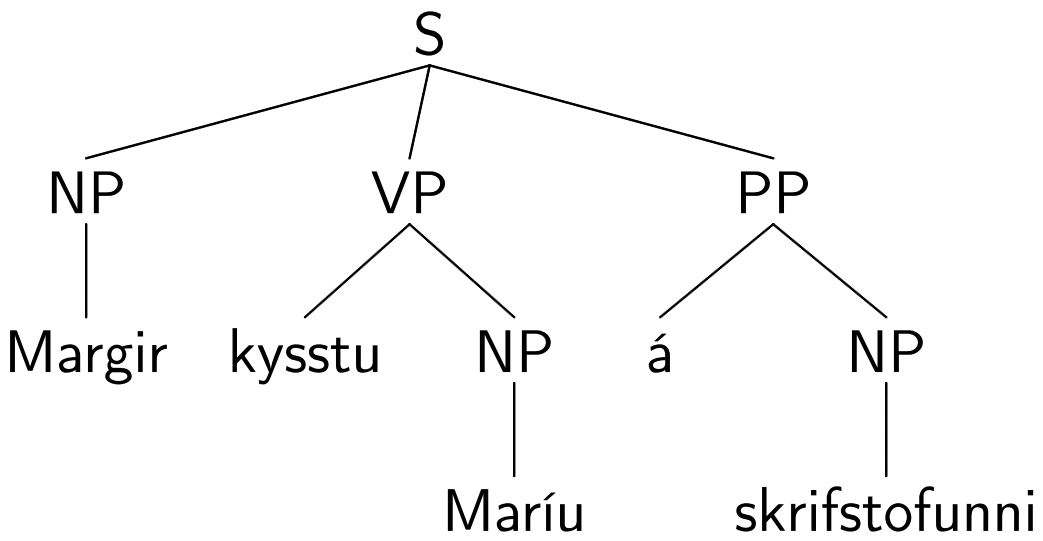
Full parsing vs. Partial parsing

- *Margir kysstu Maríu á skrifstofunni*



Full parsing vs. Partial parsing

- *Margir kysstu Maríu á skrifstofunni*



Outline

Multiword expressions (MWE) (í. fleiryrtar segðir)

Skilgreining

- Sequences of two or more words that act as a single lexical unit, for example:
  - Proper nouns (names): persons, companies, institutions
  - Temporal expressions: times, dates
  - Numerical expressions: numbers and amounts
  - Sequences of words that function like a conjunction, adverb, adjective, or a preposition.

Examples of MWEs

MWE_AdvP = MWE functioning as an **adverb**
MWE_PP = MWE functioning as a **preposition**
MWE_CP = MWE functioning as a **conjunction**
MWE_AP = MWE functioning as an **adjective**

```
[CP en c CP] [MWE_AdvP einhvern fokeo veginn nkeog MWE_AdvP]
[VP tengdist sfm3eþ VP] [NP það fphen NP]
[CP but CP] [MWE_AdvP somehow MWE_AdvP] ...

[PP [MWE_PP uppi aa á aþ MWE_PP] [NP bakkanum nkeþg NP] PP]
[PP [MWE_PP up on MWE_PP] [NP bank NP] PP]

[MWE_CP án ae þess fphee að cn MWE_CP] [VPi hafa sng VPi]
[NP nokkuð foheo NP] [PP fyrir aþ [NP stafni nkeþ NP] PP]
[MWE_CP without MWE_CP] [VPi having VPi] ...
```

Examples of MWEs

```
[MWE_AdvP við aa og c við aa MWE_AdvP] [VP gægðist sfm3eþ VP]
[NP hún fpven NP] [PP [MWE_PP yfir aa í ao MWE_PP] [NP bókina nveog NP] PP]
[MWE_AdvP now and then MWE_AdvP] [VP looked VP] ...

[CP og c CP] [NP [MWE_AP hvers fohee kyns nhee MWE_AP] líkamshirðingar nvee NP]
[CP and CP] [NP [MWE_AP various MWE_AP] ... NP]
```

Examples of MWEs

- The expressions above can be solved by using some kind of word lists.
- Proper nouns, temporal expressions and numerical expressions are a different story.
  - Nevertheless, wordlists, **gazetteers**, are sometimes used for proper nouns.
- But note that it may be necessary to recognise expressions that are not hard-coded in a list.
- $\Rightarrow$ Regular expressions!

Outline

Chunks (í. klumpar)

- A chunk = Group of words
- The difference between a chunk and a constituent is that the latter can contain nested constituents of the same type.
  - See for example slide no. 21 in the lecture "CFG and Prolog"
- A CFG is used to allow recursion in constituents
- Chunks do not contain nested chunks of the same type
- Regular expressions are thus sufficient to describe chunks.

A description of an Icelandic noun chunk

- Let us simplify the matter and only assume that a noun chunk can contain:
  - an adverb, an adjective, a noun
  - for example, “saga” (story)
  - or “skemmtileg saga” (a fun story)
  - or “mjög skemmtileg saga” (a very fun story)

In JFlex:

```
%% A finite-state automaton recognising simple noun chunks
%public
%class NounChunk
%standalone
%unicode

%{
    String Open="[NP " ;
    String Close="NP] ";
%}

WhiteSpace = [ \t\f]
WordChar = [^\r\n\t\f ]
Word = {WordChar}+
WordSpaces = {Word}{WhiteSpace}+

Gender = [kvhx] /* k=masculine, v=feminine, h=neuter, x=unspec */
Number = [ef]   /* e=singular, f=plural */
Case = [noþe]   /* n=nominative, o=accusative, þ=dative, e=genitive */
```

In JFlex (cont.):

```
AdverbTag = aa[me]?
AdjectiveTag = l{Gender}{Number}{Case}[sv][fme]
NounTag = n{Gender}{Number}{Case}[g\-]?[mös]?

Adverb = {WordSpaces}{AdverbTag}{WhiteSpace}+
Adjective = {WordSpaces}{AdjectiveTag}{WhiteSpace}+
Noun = {WordSpaces}{NounTag}{WhiteSpace}+

NounChunk = {Adverb}?{Adjective}?{Noun}

%%
{NounChunk} { System.out.print(Open + yytext() + Close);}
.           { System.out.print(yytext());}
```

Outline

Incremental finite-state parsers

(e. cascading/incremental partial/shallow/finite-state parsers)

- Based on a sequence of finite-state transducers (í. stöðuferjöldum)
- Each transducer has a specific role, for example to:
  - Mark MWEs
  - Mark verb phrases
  - Mark noun phrases
  - Mark preposition phrases
  - etc.
- The input is a tokenised and POS tagged text.
- The output of one transducer is used as the input to the next transducer in the sequence.
- The final output is the original text marked with syntactic information (e.g. chunks).

Incremental finite-state parsers

Exist for various languages

- Spanish (Molina et al. 1999)
- Swedish (Megyesi and Rydin 1999)
- German (Müller 2004)
- French (Aït-Mokhtar and Chanod 1997)
- Icelandic (Hrafn Loftsson and Eiríkur Rögnvaldsson 2007)

Efficient

- A sequence of finite-state transducers.

Incremental finite-state parsers

Exist for various languages

- Spanish (Molina et al. 1999)
- Swedish (Megyesi and Rydin 1999)
- German (Müller 2004)
- French (Aït-Mokhtar and Chanod 1997)
- Icelandic (Hrafn Loftsson and Eiríkur Rögnvaldsson 2007)

Efficient

- A sequence of finite-state transducers.

Annotation scheme

Purpose

- Description for annotation of constituent structure and syntactic functions.
- Grammar definition corpus (GDC) (í. málfræðiskilgreiningarmálheild).
  - A collection of typical sentences which have been annotated according to the annotation scheme.
  - Its purpose is to "provide an unambiguous answer to the question how to analyse any utterance in the object language" (Voutilainen, 1997).
  - Furthermore, a GDC can be used to develop a parser, because a parser should at least be able to analyse correctly the sentences in the GDC.

Outline

IceParser

- Hrafn Loftsson and Eiríkur Rögnvaldsson, 2007
- Based on an annotation scheme:
  `http://nlp.ru.is/pdf/shallowAnnotation.pdf`
- An incremental finite-state parser: `http://nlp.ru.is`
- Annotates constituents and syntactic functions.
  - Phrase/constituent structure module; 14 transducers.
  - Syntactic functions module; 8 transducers.

How is the accuracy estimated?

Constituents	Correct constituents in *gold standard*	Incorrect constituents
Generated by parser	A	B
Not generated by parser	C	

- **Precision**: $P = \frac{A}{A \cup B} = \frac{\#\text{ correct constituents in output of parser}}{\text{total }\#\text{ of constituents in output of parser}}$
- **Recall**: $R = \frac{A}{A \cup C} = \frac{\#\text{ correct constituents in output of parser}}{\text{total }\#\text{ of constituents in } \textit{gold standard}}$
- **F-measure** $= \frac{2 * P * R}{P + R}$ (e. harmonic mean)

IceParser: Evaluation

Experimental setup

- A *gold standard* was constructed:
  - About 500 sentences randomly selected from the POS tagged *IFD* corpus.
  - Manually annotated with constituent structure and syntactic functions using the annotation scheme.
- The *Evalb* (Sekine & Collins, 1997) bracket scoring program used for automatic evaluation.
- The parser evaluated using correct POS tags and tags generated by *IceTagger*.
  - POS tagging accuracy was 91.1% (unknown word ratio 7.8%).

Results for the various phrase types

Phrase type	F-measure using correct POS tags	F-measure using *IceTagger*	Freq. in test data
AdvP	91.8%	85.1%	8.2%
AP	95.1%	86.3%	8.1%
APs	87.0%	68.6%	0.5%
NP	96.8%	93.0%	37.6%
NPs	80.4%	74.3%	1.5%
PP	96.7%	91.3%	13.0%
VPx	99.2%	93.8%	19.3%
CP	100.0%	99.6%	5.7%
SCP	99.6%	97.6%	3.4%
InjP	100.0%	96.3%	0.2%
MWE	96.9%	92.6%	2.5%
All	96.7%	91.9%	100.0%

Results for the various syntactic functions

Function type	F-measure using correct POS tags	F-measure using *IceTagger*	Freq. in test data
SUBJ	68.2%	47.6%	4.7%
SUBJ>	92.7%	89.4%	30.3%
SUBJ<	83.7%	75.1%	12.3%
OBJ	0.0%	0.0%	0.2%
OBJ>	43.5%	20.0%	0.8%
OBJ<	90.2%	78.2%	19.7%
OBJAP>	71.4%	57.2%	0.2%
OBJAP<	75.0%	46.2%	0.4%
OBJNOM<	30.8%	16.7%	0.6%
...			
All	84.3%	75.3%	100.0%