

# Getting Computers to Process Language I

Human Communication 1  
Lecture 14

16/02/09

Susen Rabold

1

## Getting computers to process language

Last time we saw

- how pervasive ambiguity is in natural language

Today we will see

- how computers can accomplish some tasks in processing language
- difficulties that arise in doing this
- some applications for language technology
- the debate about statistics and symbols.

16/02/09

Susen Rabold

2

## Mechanical processing

- How have you constructed DRSs so far?
  - A Produce a syntactic tree
  - B Apply the semantic rules
- If we're more specific about the details of A and B, we're close to a computer program for performing these tasks.
- How could one get a computer program to do A?
  - We'll focus on quite simple models

16/02/09

Susen Rabold

3

## Grammar used in parsing example (a)

PN → John	S → NP VP
PN → Mary	
PN → this	NP → PN
DET → a	NP → Det N
Det → his	NP → Det AP N
Det → the	
Det → my	AP → AP A
Det → her	AP → A

16/02/09

Susen Rabold

4

## Grammar used in parsing example (b)

N → dog  
N → boy            VP → V0  
N → girl           VP → V1 NP  
N → home  
N → garden  
V1 → is  
V1 → loves  
V0 → walks  
A → angry  
A → beautiful

Susen Rabold

5

## Building syntactic trees - I

- The INPUT is the sequence of words to be analysed.
- Start off with the current symbol = S
- Repeat the following until you've run out of words:
  - α Choose a rule whose LHS is the current symbol. Remember the other choices (we may have to come back to these alternatives).

16/02/09

Susen Rabold

6

## Building syntactic trees - II

- Draw in the tree
- Work from left to right through the daughters
  - If the daughter is a word, check it's the same as the first word in the input, erase the latter, and move on to the next daughter.
  - If it's not a word, set the current symbol to be the one you're looking at and continue at α.

16/02/09

Susen Rabold

7

## Building syntactic trees - III

- On getting stuck:
  - go back to the last point where you had a choice and try a different one, erasing any tree you've drawn in, and putting any words you erased from the input back again.
- On running out of rules to try:
  - there is no analysis of the sentence (or no other analysis if you've already found one).

16/02/09

Susen Rabold

8

## Building syntactic trees - IV

- On running out of words, the tree you have drawn so far is a possible analysis of the input, if there are no dangling daughters/children.
- This is an ALGORITHM; an explicit set of instructions for solving some problem.
  - The process of building a tree is called PARSING.  
Note that we first only work on a single tree at once. In other words, this is a SERIAL (rather than PARALLEL) parser.
  - Parsing removes categorial ambiguity.

16/02/09

Susen Rabold

9

## Ambiguities multiply (a)

I 1  
saw x2  
a star x2  
with a telescope x2  
with a large x2  
lens x1  
with my friends x2

- So there are (at least)  $2^5 (= 32)$  analyses of this sentence.

16/02/09

Susen Rabold

10

## Ambiguities multiply (b)

- This is an example of exponential growth, a big problem.
- If it takes one unit of time (say 1/100 s) to find one analysis, it will take us at least 0.32s to find all of these analyses.
- If we added another ambiguity, it would take us at least 0.64s. If each word in a sentence were two-ways ambiguous, it would take us about 2 1/2 hours to process a 20-word sentence and more than 300 years to process a 40 word sentence.

16/02/09

Susen Rabold

11

## A different model . . . (a)

- Process in PARALLEL:
  - every time you have a choice of rule investigate the consequences of all possible rules at once.
- But we run into a similar problem as with the serial model:
  - we will need to be able to store a large number of analyses, probably a number exponentially related to the length of a sentence.

16/02/09

Susen Rabold

12

## A different model . . . (b)

- So, it seems that the simple model is going to get us into difficulties: both the parallel and serial models seem to require too much time or storage.

16/02/09

Susen Rabold

13

## A theoretical ideal: modularity (a)

- The model presented above is too simple, but has one theoretically very appealing property. Because we do syntactic processing followed by semantic processing in that model: *semantic information to do with a sentence can't affect our use of syntactic rules.*
- In other words, syntactic processing is isolated from semantic processing. This is highly MODULAR.

16/02/09

Susen Rabold

14

## A theoretical ideal: modularity (b)

- As a consequence, it's easy
  - to write a computer program to use as a model
  - and to predict what the behaviour of the program will be.
- On the other hand, it doesn't seem feasible to use this kind of organization in a computer model of language processing.

16/02/09

Susen Rabold

15

## Getting Computers to Process Language II

Human Communication 1

Lecture 14

16/02/09

Susen Rabold

16

## Computer applications of language technology (a)

- How can we apply models of the kind shown so far in automatically processing language?
- How is that related to current engineering practice?
- What can we learn from this about humans ?

16/02/09

Susen Rabold

17

## Computer applications of language technology (b)

Language-based computer applications are of growing importance both for

- improving the effective use of information
- broadening the base of computer literacy

Major problems will arise in exploiting background knowledge in the same way as humans do.

16/02/09

Susen Rabold

18