Natural Language Understanding

- We want to communicate with computers using natural language (spoken and written).
  - unstructured natural language — allow any statements, but make mistakes or failure.
  - controlled natural language — only allow unambiguous statements that can be interpreted (e.g., in supermarkets or for doctors).
- There is a vast amount of information in natural language.
- Understanding language to extract information or answering questions is more difficult than getting extracting gestalt properties such as topic, or choosing a help page.
- Many of the problems of AI are explicit in natural language understanding. "AI complete".

Syntax, Semantics, Pragmatics

- Syntax describes the form of language (using a grammar).
- Semantics provides the meaning of language.
- Pragmatics explains the purpose or the use of language (how utterances relate to the world).

Examples:

- *This lecture is about natural language.*
- *The green frogs sleep soundly.*
- *Colorless green ideas sleep furiously.*
- *Furiously sleep ideas green colorless.*

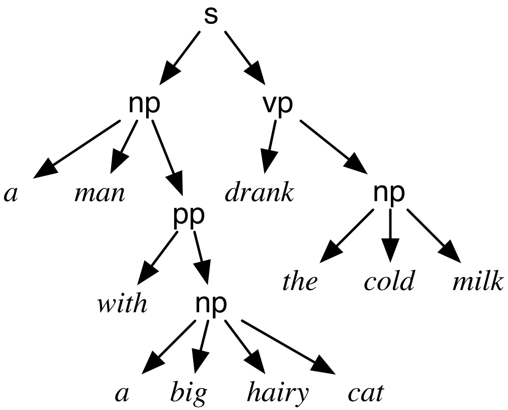
Beyond N-grams

- *A man with a big hairy cat drank the cold milk.*
- Who or what drank the milk?

Beyond N-grams

- *A man with a big hairy cat drank the cold milk.*
- Who or what drank the milk?

Simple syntax diagram:



Context-free grammar

- A terminal symbol is a word (perhaps including punctuation).
- A non-terminal symbol can be rewritten as a sequence of terminal and non-terminal symbols, e.g.,

$$sentence \longmapsto noun\_phrase, verb\_phrase$$

$$verb\_phrase \longmapsto verb, noun\_phrase$$

$$verb \longmapsto [drank]$$

- Can be written as a logic program, where a sentence is a sequence of words:

$$sentence(S) \leftarrow noun\_phrase(N), verb\_phrase(V), append(N, V, S).$$

To say word "drank" is a verb:

$$verb([drank]).$$

Difference Lists

- Non-terminal symbol s becomes a predicate with two arguments, $s(T_1, T_2)$, meaning:
  - T_2 is an ending of the list T_1
  - all of the words in T_1 before T_2 form a sequence of words of the category s.
- Lists T_1 and T_2 together form a difference list.
- "the student" is a noun phrase:

$$noun\_phrase([the, student, passed, the, course], \\ [passed, the, course])$$

Difference Lists

- Non-terminal symbol s becomes a predicate with two arguments, $s(T_1, T_2)$, meaning:
  - T_2 is an ending of the list T_1
  - all of the words in T_1 before T_2 form a sequence of words of the category s.
- Lists T_1 and T_2 together form a difference list.
- "the student" is a noun phrase:

$$noun\_phrase([the, student, passed, the, course], [passed, the, course])$$

- The word "drank" is a verb:

$$verb([drank|W], W).$$

Definite clause grammar

The grammar rule

$$sentence \longmapsto noun\_phrase, verb\_phrase$$

means that there is a sentence between T_0 and T_2 if there is a noun phrase between T_0 and T_1 and a verb phrase between T_1 and T_2:

$$\begin{aligned} & sentence(T_0, T_2) \leftarrow \\ & \quad noun\_phrase(T_0, T_1) \land \\ & \quad verb\_phrase(T_1, T_2). \end{aligned}$$

$$\overbrace{\underbrace{T_0 \qquad\qquad\qquad}_{noun\_phrase} \underbrace{T_1 \qquad\qquad\qquad}_{verb\_phrase}}^{sentence} T_2$$

Definite clause grammar rules

The rewriting rule

$$h \longmapsto b_1, b_2, \ldots, b_n$$

says that h is b_1 then b_2, ..., then b_n:

$$\begin{aligned}
&h(T_0, T_n) \leftarrow \\
&\qquad b_1(T_0, T_1) \wedge \\
&\qquad b_2(T_1, T_2) \wedge \\
&\qquad \vdots \\
&\qquad b_n(T_{n-1}, T_n).
\end{aligned}$$

using the interpretation

$$\overbrace{\underbrace{T_0 \qquad}_{b_1} \underbrace{T_1 \qquad}_{b_2} T_2 \cdots \underbrace{T_{n-1} \qquad}_{b_n}}^{h} T_n$$

Terminal Symbols

Non-terminal h gets mapped to the terminal symbols, $t_1, ..., t_n$:

$$h([t_1, \cdots, t_n | T], T)$$

using the interpretation

$$\overbrace{t_1, \cdots, t_n}^{h} T$$

Thus, $h(T_1, T_2)$ is true if $T_1 = [t_1, ..., t_n | T_2]$.

Complete Context Free Grammar Example

see
`http://artint.info/code/Prolog/ch12/cfg_simple.pl`

What will the following query return?

$noun\_phrase([the, student, passed, the, course, with, a, computer], R).$

Complete Context Free Grammar Example

see
`http://artint.info/code/Prolog/ch12/cfg_simple.pl`

What will the following query return?

$noun\_phrase([the, student, passed, the, course, with, a, computer], R).$

How many answers does the following query have?

$sentence([the, student, passed, the, course, with, a, computer], R).$

Augmenting the Grammar

Two mechanisms can make the grammar more expressive:

- extra arguments to the non-terminal symbols
- arbitrary conditions on the rules.

We have a Turing-complete programming language at our disposal!

Building Structures for Non-terminals

Add an extra argument representing a parse tree:

$$
\begin{aligned}
& \mathit{sentence}(T_0, T_2, s(NP, VP)) \leftarrow \\
& \qquad \mathit{noun\_phrase}(T_0, T_1, NP) \wedge \\
& \qquad \mathit{verb\_phrase}(T_1, T_2, VP).
\end{aligned}
$$

Enforcing Constraints

Add an argument representing the number (singular or plural), as well as the parse tree:

$$
\begin{aligned}
& sentence(T_0, T_2, Num, s(NP, VP)) \leftarrow \\
& \qquad noun\_phrase(T_0, T_1, Num, NP) \wedge \\
& \qquad verb\_phrase(T_1, T_2, Num, VP).
\end{aligned}
$$

The parse tree can return the determiner (definite or indefinite), number, modifiers (adjectives) and any prepositional phrase:

$$
\begin{aligned}
& noun\_phrase(T, T, Num, no\_np). \\
& noun\_phrase(T_0, T_4, Num, np(Det, Num, Mods, Noun, PP)) \leftarrow \\
& \qquad det(T_0, T_1, Num, Det) \wedge \\
& \qquad modifiers(T_1, T_2, Mods) \wedge \\
& \qquad noun(T_2, T_3, Num, Noun) \wedge \\
& \qquad pp(T_3, T_4, PP).
\end{aligned}
$$

Complete Example

see
`http://artint.info/code/Prolog/ch12/nl_numbera.pl`

Question-answering

- How can we get from natural language to a query or to logical statements?
- Goal: map natural language to a query that can be asked of a knowledge base.
- Add arguments representing the individual and the relations about that individual. E.g.,

  $$noun\_phrase(T_0, T_1, O, C_0, C_1)$$

  means
  - $T_0 - T_1$ is a difference list forming a noun phrase.
  - The noun phrase refers to the individual O.
  - C_0 is list of previous relations.
  - C_1 is C_0 together with the relations on individual O given by the noun phrase.

Example natural language to query

see
http://artint.info/code/Prolog/ch12/nl_interface.pl

Context and world knowledge

The student took many courses. Two computer science courses and one mathematics course were particularly difficult. The mathematics course...

Context and world knowledge

The student took many courses. Two computer science courses and one mathematics course were particularly difficult. The mathematics course...

Who was the captain of the Titanic?
Was she tall?