

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).

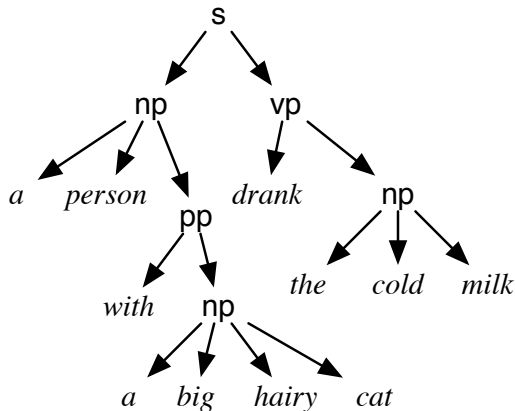
Examples:

- *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 person 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 string representing a word (perhaps including punctuation and composite words, such as “hot dog” or “Buenos Aires”).
- A **non-terminal symbol** can be rewritten as a sequence of terminal and non-terminal symbols, e.g.,

$sentence \mapsto noun_phrase, verb_phrase$

$verb_phrase \mapsto verb, noun_phrase$

$verb \mapsto [drank]$

- 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"])*

- The words “drank” and “passed” are verbs:

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

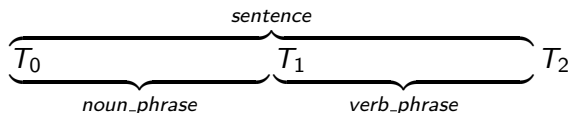
verb(["passed" | W], W).

Definite clause grammar

The grammar rule

$$\textit{sentence} \mapsto \textit{noun_phrase}, \textit{verb_phrase}$$

represented as: 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 :

$$\begin{aligned} \textit{sentence}(T_0, T_2) \leftarrow \\ \textit{noun_phrase}(T_0, T_1) \wedge \\ \textit{verb_phrase}(T_1, T_2). \end{aligned}$$


Definite clause grammar rules

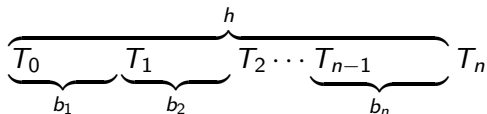
The rewriting rule

$$h \mapsto b_1, b_2, \dots, b_n$$

says that h is b_1 then b_2, \dots , then b_n :

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

using the interpretation



Terminal Symbols

Non-terminal h gets mapped to the terminal symbols, t_1, \dots, t_n :

$$h([t_1, \dots, t_n \mid T], T)$$

using the interpretation

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

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

Complete Context Free Grammar Example

see

https:

`//artint.info/3e/resources/ch15/geography_CFG.pl`

(also load https:

`//artint.info/3e/resources/ch15/geography_DB.pl`)

What will the following query return?

`noun_phrase([" a", " country", " that", " borders", " Chile"], L3).`

How many answers does the following query have?

`noun_phrase([" a", " Spanish", " speaking", " country", " that", " borders", "`

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!

- How can we get from natural language directly to the answer?
- Goal: map natural language to a query that is asked of a knowledge base.
- Add arguments representing the individual

noun_phrase(T_0, T_1, O)

means

- ▶ $T_0 - T_1$ is a difference list forming a noun phrase.
- ▶ The noun phrase refers to the individual O .
- Can be implemented by the parser directly calling the knowledge base.

Example natural language to query

see

`https://www.cs.ubc.ca/~poole/cs312/2023/prolog/
geography_string.pl`

% A noun phrase is a determiner followed by adjectives followed
% by a noun followed by an optional modifying phrase:

```
noun_phrase(L0, L4, Ind) :-  
    det(L0, L1, Ind),  
    adjectives(L1, L2, Ind),  
    noun(L2, L3, Ind),  
    omp(L3, L4, Ind).
```

Adjectives provide properties

```
% adj(T0,T1,Entity) is true if T0-T1
% is an adjective that is true of Entity
adj(["large" | L], L, Ind) :- large(Ind).
adj([LangName, "speaking" | L], L, Ind) :-
    language(Ind, Lang), name(Lang, LangName).

% adjectives(T0,T1,Entity) is true if
% T0-T1 is a sequence of adjectives that true of Entity
adjectives(T0,T2,Entity) :-
    adj(T0,T1,Entity),
    adjectives(T1,T2,Entity).
adjectives(T,T,_).
```

Verbs and prepositions provide relations

reln(T0, T1, Subject, Object)

- $T0 - T1$ is a verb or preposition that provides
- a relation that true between *Subject* and *Object*

```
reln(["borders" | L], L, Sub, Obj) :- borders(Sub, Obj).
```

```
reln(["bordering" | L], L, Sub, Obj) :- borders(Sub, Obj).
```

```
reln(["next", "to" | L], L, Sub, Obj) :- borders(Sub, Obj).
```

```
reln(["the", "capital", "of" | L], L, Sub, Obj) :-  
    capital(Obj, Sub).
```

```
reln(["the", "name", "of" | L], L, Sub, Obj) :-  
    name(Obj, Sub).
```

Verbs and prepositions provide relations

```
% A modifying phrase / relative clause is either  
% a relation (verb or preposition)  
% followed by a noun_phrase or  
% 'that' followed by a relation then a noun_phrase
```

```
mp(L0, L2, Subject) :-  
    reln(L0, L1, Subject, Object),  
    aphrase(L1, L2, Object).
```

```
mp(["that" | L0], L2, Subject) :-  
    reln(L0, L1, Subject, Object),  
    aphrase(L1, L2, Object).
```

```
% An optional modifying phrase is either a modifying phrase
```

```
omp(L0,L1,E) :-  
    mp(L0,L1,E).  
omp(L, L, _).
```


Building a list of constraints on the entity (geography_query_string.pl)

`noun_phrase(L0,L4,Entity,C0,C4)` is true if

- `L0` and `L4` are list of words, such that
 - ▶ `L4` is an ending of `L0`
 - ▶ the words in `L0` before `L4` (written `L0 – L4`) form a noun phrase
- `Entity` is an individual that the noun phrase is referring to
- `C0` is a list such that `C4` is an ending of `C0` and `C0 – C4` contains the constraints imposed by the noun phrase

```
noun_phrase(L0,L4,Entity,C0,C4) :-  
    det(L0,L1,Entity,C0,C1),  
    adjectives(L1,L2,Entity,C1,C2),  
    noun(L2,L3,Entity,C2,C3),  
    mp(L3,L4,Entity,C3,C4).
```

- 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

https://artint.info/3e/resources/ch15/geography_QA.pl

ALso load

https://artint.info/3e/resources/ch15/geography_DB.pl

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?