Natural Language Processing

- NLP involves reasoning with the *meaning* of language, not just manipulating surface text.
- so, for instance, Eliza is NOT an NLP system (in emacs, try `M-X doctor`):

```
S: Hi. What brings you to see me?
U: I'm feeling a bit depressed.
S: Are you depressed often?
```

- this is obtained by simple string searching and replacing, but has no sophisticated reasoning:

```
U: I'm feeling a bit XyZZYhjla222 today.
S: Why do you say you are feeling a
    bit XyZZYhjla222 today?
```

- by reasoning about the meaning of the text, the applications can be made much more flexible:
  - questions can be expressed in unusual ways ("France has what city as its capital?")
  - cooperative answers can be given for implicit questions ("the next flight is completely booked, but there's still room on the 8:04 flight")
  - conciseness ("When is the next train to London?", "4.40", "which platform is it?")

NL Understanding

- to understand language it is necessary to determine how words are grouped together in meaningful ways.
- we need:
  1. rules concerning how the language is organized:
     lexicon: checking that the words in the sentence do exist
     syntax: checking that the sequence of words has a "valid" structure and extracting that structure to determine the meaning
     semantics: obtaining a representation of what the sentence means:

John gave the book to Mary.
The book was given to Mary by John.
	`give-action:`
	`agent: john`
	`object: book`
	`receiver: mary`

  2. an algorithm using these rule to parse a sentence
- it is a bit like parsing computer languages
- but more complex, mainly because natural language is ambiguous

For example

- You may have a simple grammar (syntax) like:

S = NP(number) + VP (number)	a sentence is formed by a nominal phrase and a verbal phrase, with accordance on number (both plural or both singular)
NP(number) = N(number)	a nominal phrase can be either a name...
NP(number) = P(number)	... or a pronoun
VP(number) = TV(number) + N(-)	a verbal phrase can be either a transitive verb, with an object, which is a name (no matter whether the object is plural or singular)..
VP(number) = IV(number)	or an intransitive verb

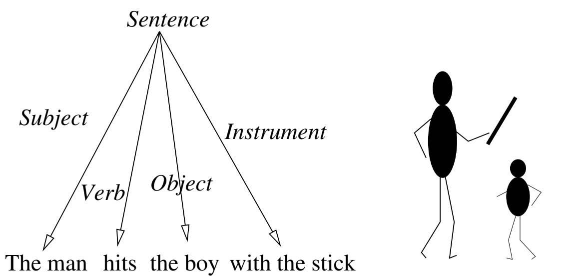
- A lexicon such as:

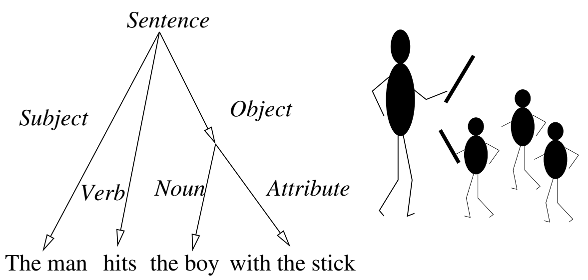
N(singular) = {Floriana, The cat}	N(plural) = {lecturers, cats}
P(singular) = {she}	P(plural) = {they}
TV(plural) = {love,hate}	TV(singular) = {loves,hates}
IV(plural) = {jump,travel}	IV(singular) = {jumps,travels}

- and a way to transform sentences into facts that can be managed automatically:

lecturers love cats $\rightarrow$ `love(lecturers,cats)`

Natural Language Ambiguity





In most of the cases, **context** helps disambiguating.

A database NL query system

Courses

Course	Semester	CourseDep.
COMP507	1	Comp. Science
COMP352	2	Comp. Science
COMP552	2	Comp. Science
COMP102	2	Comp. Science
MATH103	1	Mathematics

Lectures

Course	Day	Time	Theatre
COMP352	Monday	11am	RB6
COMP352	Tuesday	11am	RB8
COMP552	Thursday	10am	C
COMP102	Monday	12pm	B
MATH103	Tuesday	15pm	B

Lecturers

StaffNo	1stName	2ndName	Office	Department
5235234	Floriana	Grasso	2.08	Comp. Science
9759212	Paul	Dunne	6.03	Comp. Science
4626737	Dean	Jones	2.06	Comp. Science

TeachingDuties

StaffNo	Course
5235234	COMP507
5235234	COMP352
5235234	COMP552
5235234	COMP102
4626737	COMP507
9759212	COMP102
4626737	MATH103

Departments

Department	Address
Comp. Science	Peach Street
English	Chatham Street

- *Where is Floriana's office?*

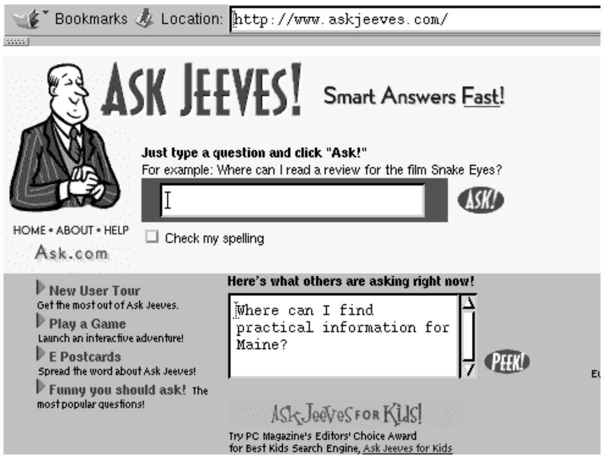
```
SELECT Office FROM LECTURERS
WHERE 1stName = ``Floriana'';
```

- *Which courses does Floriana teach?*

```
SELECT Course FROM TEACHINGDUTIES, LECTURERS
WHERE LECTURERS.1stName = ``Floriana''
AND LECTURERS.StaffNo = TEACHINGDUTIES.StaffNo;
```

- *Where is Floriana?* ????

A WWW Example



The system has four components:

1. Question Processing Engine: matches the input query to a "question template" involving both semantics and syntax
2. Answer Processing Engine
3. Knowledge Base, including a semantic and a syntactic network
4. MetaSearch (using 5 search engines)

Natural Language Generation

- NLP doesn't just involve language understanding, but also generating a text (to answer user's queries)
- natural language generation (NLG) systems generate texts in English or other human languages from computer-accessible data (e.g., data in database).
- e.g., from data:

```
isa(diabetes, disease)
symptom(diabetes, tiredness)
symptom(diabetes, nausea)
treatment(diabetes, insulin-injection)
```

  one can generate:

  Diabetes is a disease. Its symptoms include tiredness and nausea. It can be treated with insulin injections.

A Basic Architecture for NLG

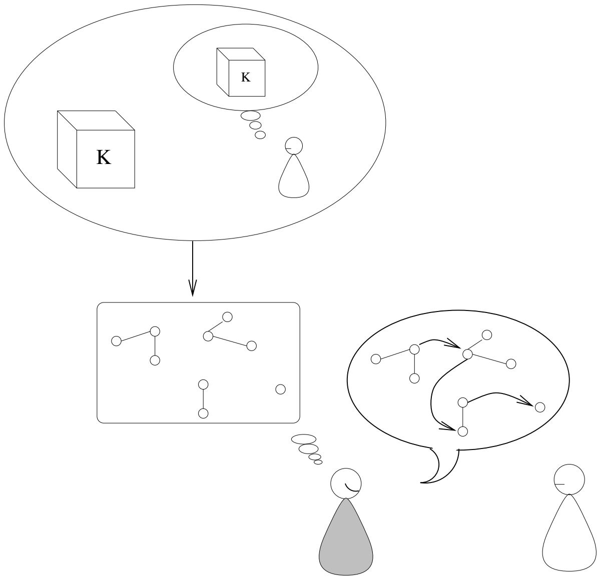
If we just consider the inputs and outputs..

Inputs: some of:

- A goal.. usually either expressed as:
  - the impact we want to have on the user e.g., "Persuade hearer to buy Percil".
  - or a description of the text required e.g., "Describe diabetes" or "Compare diabetes with heart disease".
- Characteristics of the user (e.g., "Intelligent user who likes whiter whites").
- Constraints on form of target text (e.g., short)

Output: A text, possibly marked up with formatting or hypertext instructions.

NLG requires lots of knowledge



- on the domain
- on the user
- on what the user may know/think about the domain
- on how things can be said in sentences

What does NLG involve?

Like NL Understanding, generation is typically split up into stages:

Content Determination and Text Planning: Deciding on what should be said and how it should be structured.

Sentence Planning: Deciding how information should be split into sentences, when to use pronouns and discourse markers, etc.

Realisation: Creating individual grammatically correct sentences.

Content Determination and Text Planning

- Issues
  - How, given a list of facts, can we select those that are relevant given a particular goal, user, and context?
  - How can we order these selected facts so that the text is coherent, each fact following from last in sensible order.
- Approaches
  1. Use general knowledge of how communicative goals relate to typical text structures (e.g., use of coherence relations), and use simple hierarchical planning techniques to plan text.
  2. Analyse many texts in target application area, abstract schemas that capture typical text structures, and use these to guide what to put in text. (e.g., recipe = title + ingredient list + sequence of steps).

  The former is more general, but latter simpler and OK for many applications.

Sentence Planning

- Issues
  - How should content be split into sentences?

    1cs82, the Human Computer Interaction Module for Computer Science, is taught by Floriana

    vs

    1cs82 is the Human Computer Interaction module. 1cs82 is a Computer Science module. 1cs82 is taught by Floriana
  - When should we use pronouns? (e.g., we may use “it” in the above example).
  - How do we obtain English terms given coded data in data/knowledge base? (e.g., arts-festival as “international arts festival”). How might these be varied according to the user and context?
  - When should we use discourse markers, e.g., “however”, “anyway”.

    I liked the HCI module. However, I would have liked to do some lab.

Realisation

- Issues
  - How can we construct grammatically correct sentences given some fact to convey?
  - How do we select between alternative ways of saying the same thing?
  - Which grammatical constructs should be used when?

    e.g.

    John saw Mary on the hill.

    vs

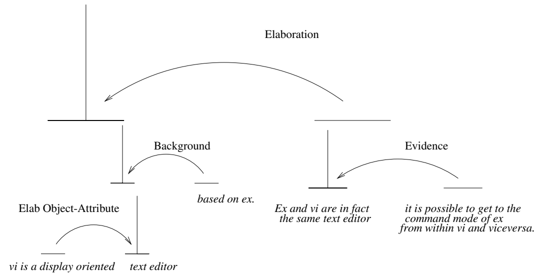
    Mary was seen by John while on the hill.
  - How do we ensure that correct tenses and agreement are used?

An example

Suppose we want to generate manual description automatically for Unix commands, such as:

> *vi is a display oriented text editor based on ex. ex and vi are, in fact, the same text editor; it is possible to get to the command mode of ex from within vi and vice-versa*

by using some linguistic knowledge such as:



We need to have some facts in our domain:

```
is-a (vi, text editor)
is-a (ex, text editor)
attribute (vi, display oriented)
based-on (vi, ex)
same (vi, ex)
get-from (vi, ex)
get-from (ex, vi)
```

Basic Action

Let us call our basic act *Inform*: when applied to an argument x it prints out the correspondent phrase (let us ignore for the moment how that phrase is constructed). So we will have for instance:

```
Inform (is-a (vi, text-editor))
```

→ *"vi is a text editor"*

```
Inform (get-from (vi, ex))
```

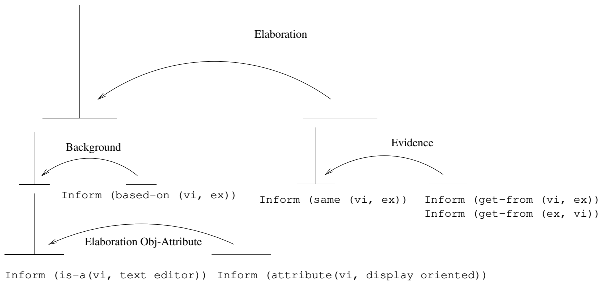
→ *"it is possible to get to the command mode of ex from within vi"*

Planning the text: a set of "operators"

```
GOAL:        Explain ?x
CONSTRAINTS: -
SUBGOALS:    Describe ?x
             DescribeAnalogies ?x
RHET-REL:    Elaboration

GOAL:        Describe ?x
CONSTRAINTS: Exists ?y based-on(?x,?y)
             Know(Hearer,?y)
SUBGOALS:    Define ?x
             Inform based-on(?x,?y)
RHET-REL:    Background

GOAL:        Define ?x
CONSTRAINTS: Exists ?y is-a(?x,?y)
SUBGOALS:    Inform is-a(?x,?y)
             Forall ?a attribute(?x,?a)
                Inform attribute(?x,?a)
RHET-REL:    Elab Object-Attribute

GOAL:        DescribeAnalogies ?x
CONSTRAINTS: Exists ?y same(?x,?y)
SUBGOALS:    Inform same(?x,?y)
             (Forall ?rel ?rel(?x,?y)
                Inform ?rel(?x,?y)
             Forall ?rel ?rel(?y,?x)
                Inform ?rel(?y,?x))
RHET-REL:    Evidence
```

Text produced

Input to the system is the goal: `(Explain vi)`.



vi is a text editor. vi is display oriented. vi is based on ex. ex and vi are the same. In fact, it is possible to get to the command mode of ex from within vi and it is possible to get to the command mode of vi from within ex

some sentence planning may be used to make the text smoother