CS 5100: Foundations of Artificial Intelligence

Natural Language Processing

Prof. Amy Sliva
December 1, 2011

Outline

- Information theory (review)
- Decision tree ID3 algorithm
- Natural Language Processing
- Part of speech tagging

Information theory

- Information is about **categories** and **classification**
- We measure **quantity** of information by resources need to represent/store/transmit
- Messages are sequences of 0s and 1s (dots/dashes)—called "bits" (for binary digits)

- **Information quantity** of a message where the (uniform) probability of each value is p:

$$I = -\log p \text{ bits}$$

Entropy

- Entropy $H(E)$ is the **average information** (in bits) of events $e_1,\ldots,e_k$ in a long repeated sequence E

$$1/k * \sum_{j=1}^{k} I(e_j) =$$

$$1/k * \sum_{i=1}^{n} I(x_i)\ (k * p_i) \ = k/k * \sum_{i=1}^{n} I(x_i)\ (p_i)$$

$$\sum_{i=1}^{n} -log(p_i) * p_i \ = \mathbf{-\sum_{i=1}^{n} p_i * log(p_i)\ bits}$$

- For uniform distribution this is same as $-log\ P(x_i)$ (**information quantity**) since all $P(x_i)$ are equal

Entropy (cont.)

- Entropy sometimes called "**disorder**"
  - Represents lack of predictability as to the outcome for any element of a sequence (or set)

- If a set has just one outcome

  entropy = 1 * -*log*(1) = 0

- If there are 2 outcomes, then 50/50 probability gives the **maximum entropy**—complete unpredictability
  - Generalizes to any uniform distribution for *n* outcomes

  -(0.5 * *log*(0.5) + 0.5 * *log*(0.5)) = 1 bit

  NOTE: *log*(1/2) = -*log*(2) = -1

Calculating entropy

- Consider a biased coin: P(*heads*) = 0.75 and P(*tails*) = 0.25
- What is the entropy of a coin toss outcome?

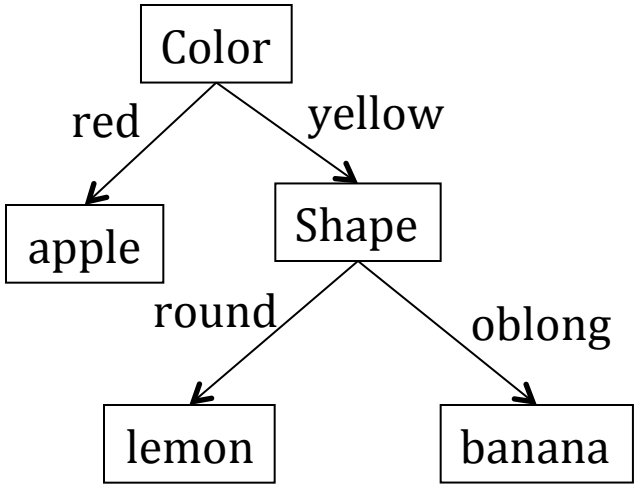
$H = 0.25 * -log(0.25) + 0.75 * -log(0.75) = 0.811$ bits

- A fair coin toss has more "information"
- **The more unbalanced the probabilities, the more predictable the outcome, the less you learn from each message**

Decision trees

- One possible representation of a **hypothesis** for learning a **target function**

Record#	Color	Shape	Fruit
1	red	round	apple
2	yellow	round	lemon
3	yellow	oblong	banana



- The goal is to create an "**efficient**" classification tree that always gives the same answer as the table

How do we learn a decision tree from data?

- Given a table with one **result attribute** and several **predictor attributes**, a **classification** (decision) **tree** is a tree s.t.
  1. Each leaf is labeled with a value of the result attribute
  2. Each non-leaf is labeled with the name of a predictor attribute
  3. Each link is labeled with one value of the parent's predictor

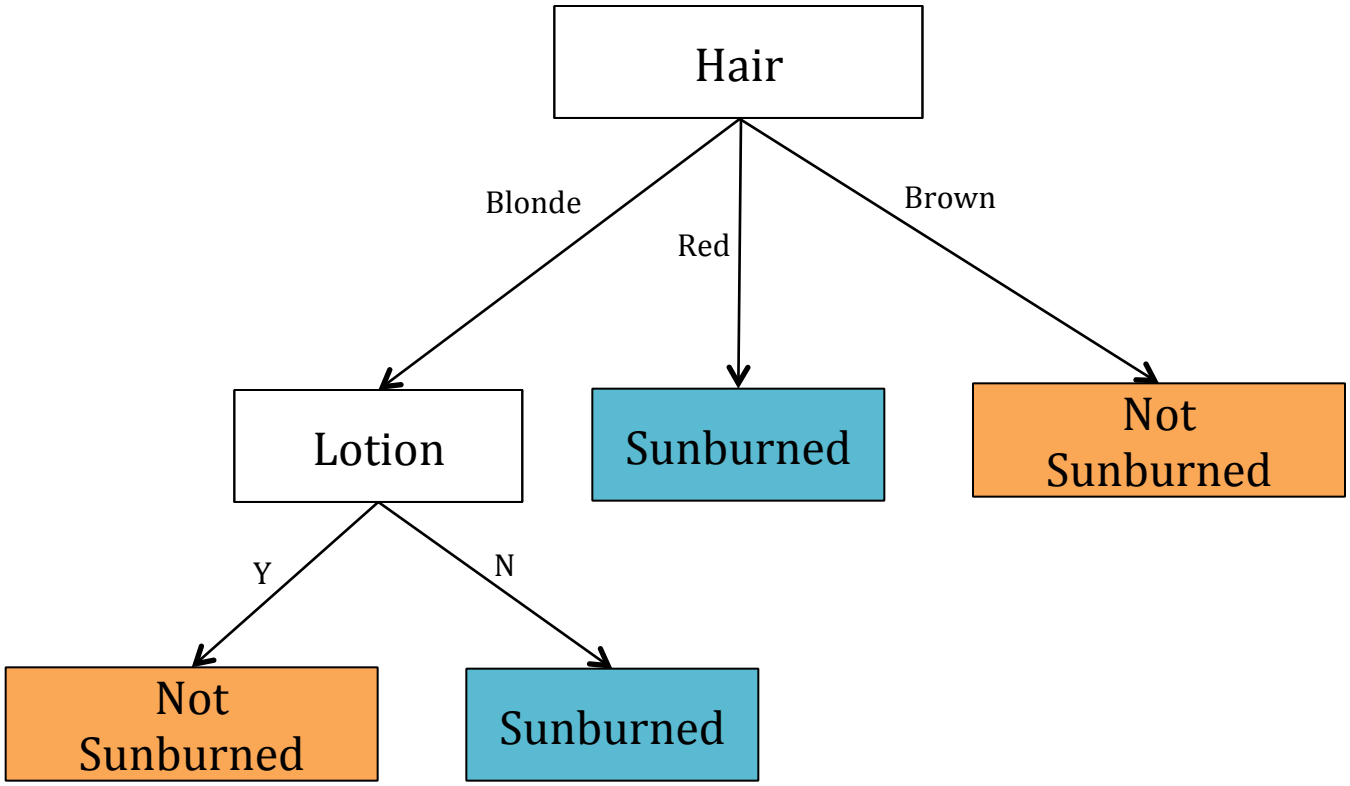
- **ID3 algorithm**
  - Takes table as input
  - "Learns" a classification tree that efficiently maps predictor value sets to their results from the table

A well-known toy example: sunburn data

Name	Hair	Height	Weight	Lotion	Sunburned
Sarah	Blonde	Average	Light	No	Yes
Dana	Blonde	Tall	Average	Yes	No
Alex	Brown	Short	Average	Yes	No
Annie	Blonde	Short	Average	No	Yes
Emily	Red	Average	Heavy	No	Yes
Pete	Brown	Tall	Heavy	No	No
John	Brown	Average	Heavy	No	No
Katie	Blonde	Short	Light	Yes	No

- Predictor attributes: Hair, Height, Weight, Lotion

Sunburn decision tree



Outline of the ID3 algorithm

- Create the root and make its COLLECTION the entire table
- Select any **non-singular** leaf node *N* to SPLIT
  - Choose the best attribute *A* for splitting *N* (**use info theory**)
  - For each value of *A* (a_1, a_2,…) create a child of *N*, N_{ai}
  - Label the links from *N* to its children "$A = a_i$"
  - SPLIT the collection of *N* among its children according to their values of *A*
- When no more non-singular nodes exist, the tree is finished

- **Singular node** is one whose COLLECTION includes just one value for the result attribute
  i.e., entropy = 0

Decision tree learning

- **Goal:** find a small tree consistent with the training examples
- **Intuition:** (recursively) choose "most significant" attribute as root of (sub)tree

```
function DECISION-TREE-LEARNING(examples, attributes, parent_examples) returns
tree

  if examples is empty then return PLURALITY-VALUE(parent_examples)
  else if all examples have the same classification then return the classification
  else if attributes is empty then return PLURALITY-VALUE(examples)
  else
      A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
      tree ← a new decision tree with root test A
      for each value v_k of A do
          exs ← {e : e ∈ examples and e.A = v_k}
          subtree ← DECISION-TREE-LEARNING(exs, attributes − A, examples)
          add a branch to tree with label (A = v_k) and subtree subtree
      return tree
```

- **How do we choose the most significant attribute?**

Choosing the best attribute to SPLIT

- Choose the attribute that is **most informative** and reduces entropy (disorder) the most
  - Attribute with highest **information gain**

- Assume there are k attributes we can choose
  - For each one, compute how much less entropy exists in the resulting children than we had in the parents
    $H(N)$ = weighted sum of H(children of N)

  - Each child's entropy is weighted by the "probability" of that child (estimated by the proportion of the parent's collection that would be transferred to the child in the split)

Constructing a decision tree with ID3

- Collection for split node 1 (root node) is full set of data
  $C(S_1)$ = {S, D, X, A, E, P, J, K}
  P(*Sunburn*) = {3/8, 5/8}
  $H(N_1)$ = -[3/8 *log*(3/8) + 5/8 *log*(5/8)]
  = 0.53 + 0.424 = **0.954**

- Find information gain (IG) for all four predictors:
  hair, height, weight, lotion

- Start with **lotion**: values {yes, no}
  Child 1 (yes) = {D, X, K}
  P(*Sunburn*) = {0/3, 3/3}
  H(Child 1) = 0

  Child 2 (no) = {S, A , E, P, J}
  P(*Sunburn*) = {3/5, 2/5}
  H(C2) = -[3/5 *log*(3/5) + 2/5 *log*(2/5)]
  = 0.971

  H({C1, C2}) = 3/8 *0 + 5/8 * 0.971 = 0.607
  IG(lotion) = 0.954 - 0.607 = 0.347

Constructing a decision tree with ID3

- Next try **hair color**: values {blonde, brown, red}
  Child 1 (blonde) = {S, D, A, K}
  P(*Sunburn*) = {2/4, 2/4}
  H(C1) = 1

  Child 2 (brown) = {X, P, J}
  P(*Sunburn*) = {0/3, 3/3}
  H(C2) = 0

  Child 3 (blonde) = {E}
  P(*Sunburn*) = {1/1, 0/1}
  H(C3) = 0

  H({C1, C2, C3}) = 4/8 *1 + 3/8 * 0 + 1/8 * 0 = 0.5
  IG(hair color) = 0.954 - 0.5 = 0.454

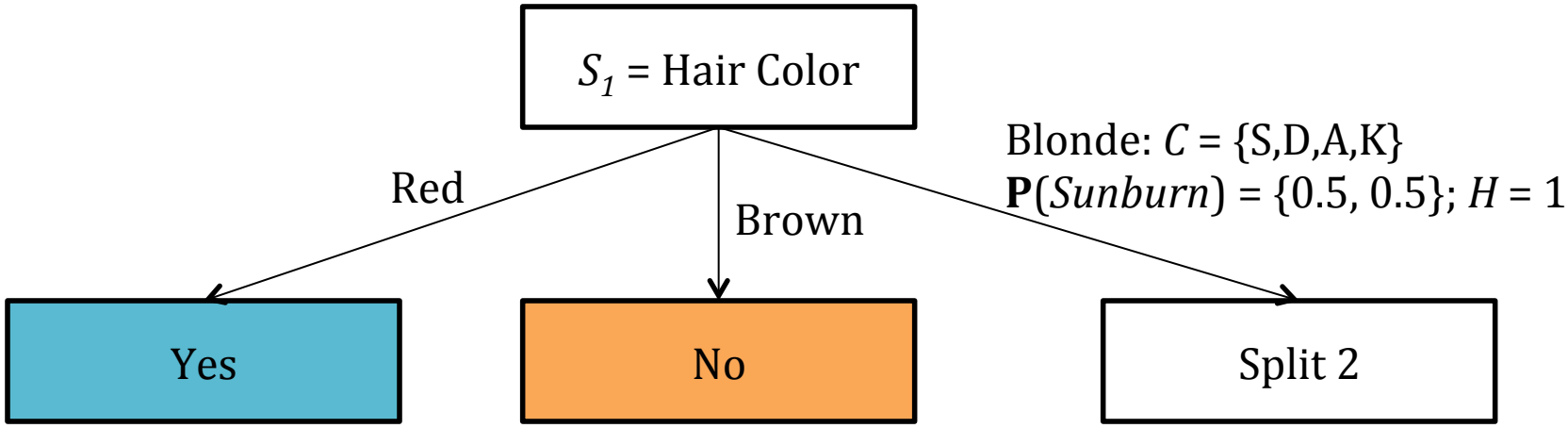
- Next try **height**…
  IG(height) = 0.954 - 0.69 = 0.26

- Next try **weight**…
  IG(weight) = 0.954 - 0.94 = 0.014

Constructing a decision tree with ID3

- Hair color wins!

- Draw the first split and assign the collections



Constructing a decision tree with ID3

- Collection for split node 2

  $C(S_2)$ = {S, D, A, K}

  P(*Sunburn*) = {0.5, 0.5}

  $H(S_1)$ = -[0.5 *log*(0.5) + 0.5 *log*(0.5)] = **1**

- Start with **lotion**: values {yes, no}

  Child 1 (yes) = {D, K}

  P(*Sunburn*) = {0/2, 2/2}

  H(C1) = 0

  Child 2 (no) = {S, A}

  P(*Sunburn*) = {2/2, 0/2}

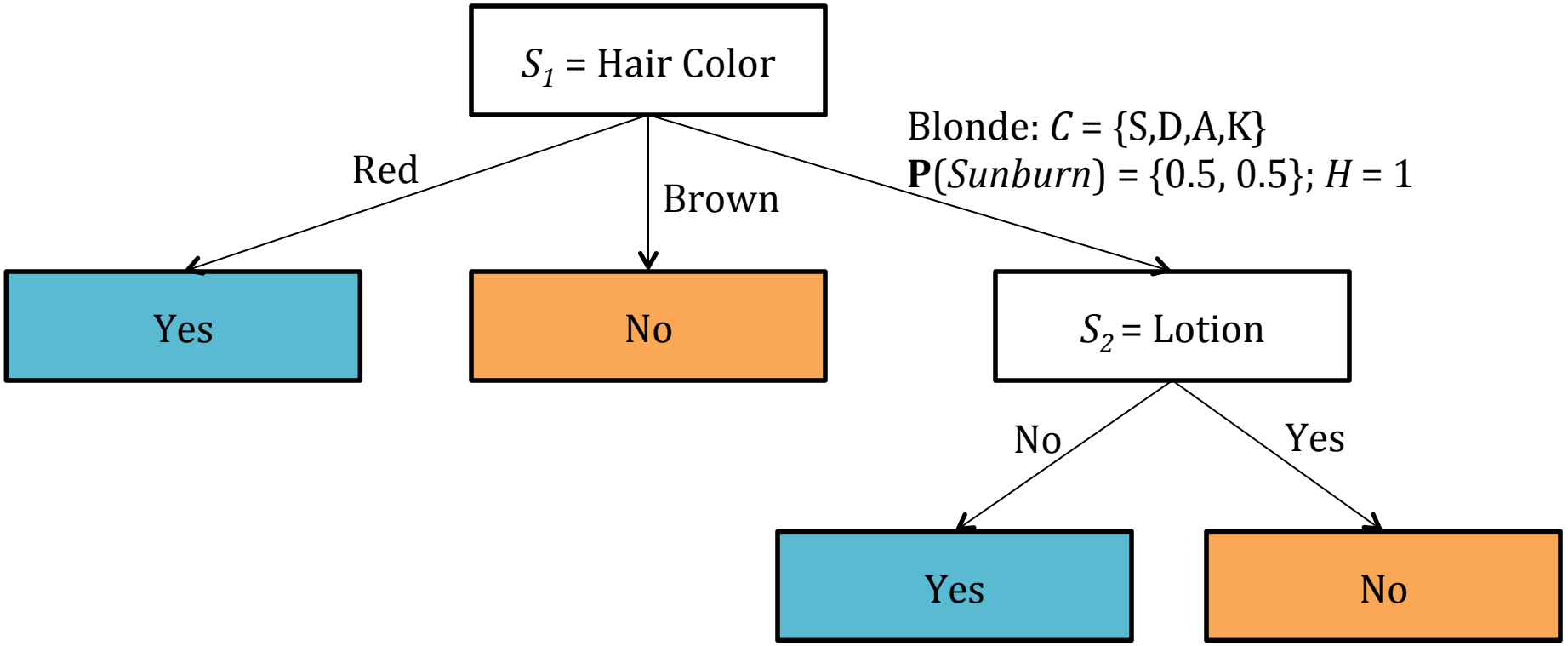
  H(C2) = 0

  H({C1, C2}) = 0

  IG(lotion) = 1- 0 = 1

- **No need to go any further!**

Constructing a decision tree with ID3



- Compact decision tree—only need two attributes to classify entire table!

Natural language processing

- Natural **language**
  - Language spoken by people
  - E.g., English, Japanese, Swahili, etc. as opposed to artificial languages like C++, Java, etc.

- Natural language **processing**
  - Applications that deal with natural language in one way or another

- Computational linguistics
  - Doing linguistics on computers
  - More on the linguistic side than NLP, but closely related

Major scientific challenges

- Lexical semantics/lexical ambiguity
- Syntax and Parsing
- Meaning (Semantics)
  - Reference resolution
- Pragmatics
- Dialog
  - Human-machine dialog

- Speech (a whole different set of problems!!)

Major application challenges

- Huge amounts of **data**
  - Internet = at least 20 billion pages
  - Intranets

- Applications for processing large amounts of text **require NLP expertise**
  - Classify text into categories
  - Index and search large texts
  - Automatic translation
  - Speech understanding (e.g., understand phone conversations)
  - Information extraction (e.g., extract useful information from resumes)
  - Automatic summarization (e.g., condense 1 book into 1 page)
  - Question answering
  - Knowledge acquisition
  - Text generation/dialogues

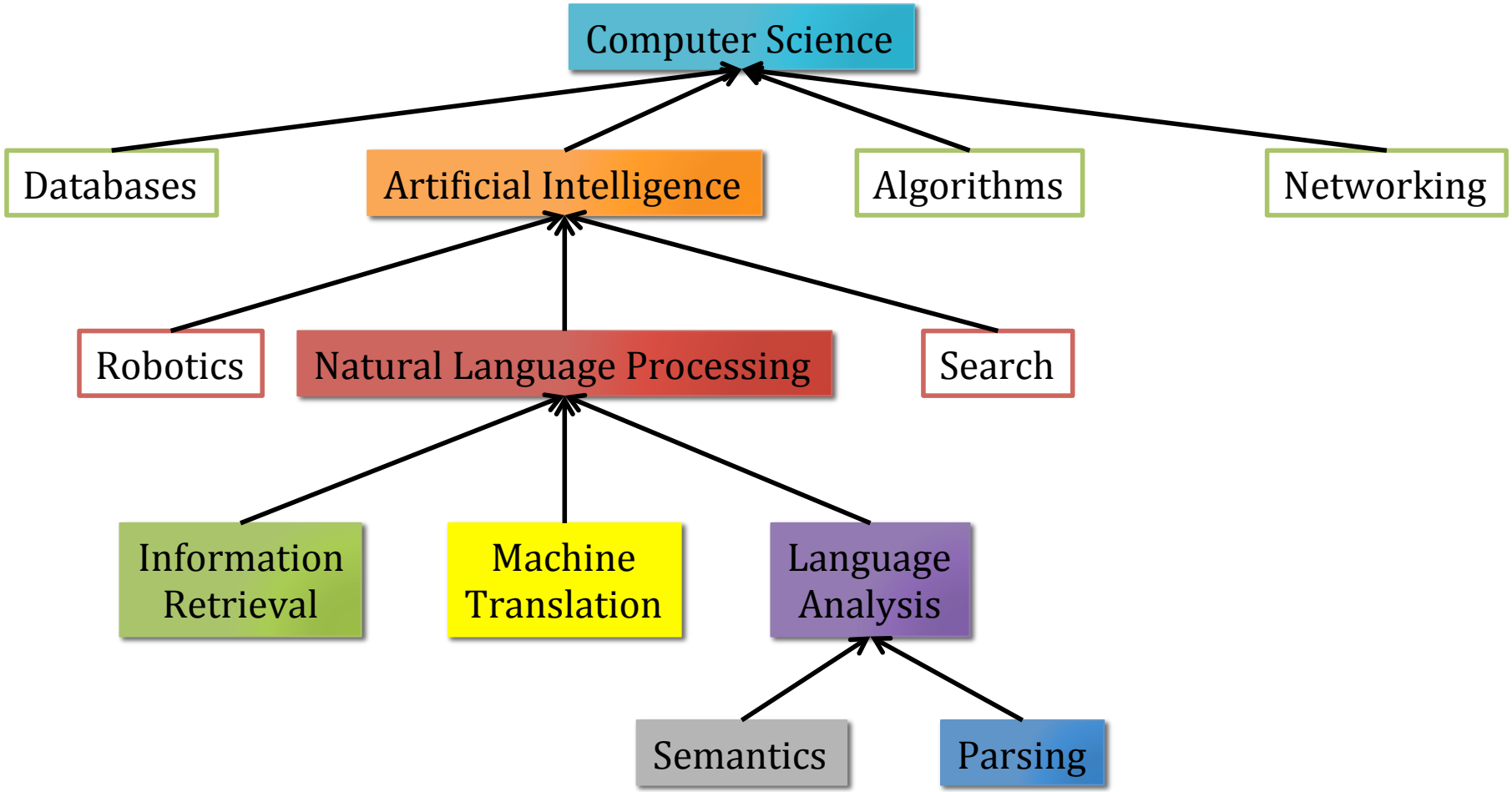
Why is natural language processing hard?

- kJfmmfj mmmvvv nnnffn333
- Uj iheale eleee mnster vensi credur
- Baboi oi cestnitze
- Coovoel2^ ekk; ldsllk lkdf vnnjfj?
- Fgmflmllk mlfm kfre **xnnn!**

Computers lack knowledge!

- Computers “see” text in English the same way you saw the previous text!
- People have no trouble understanding language
  - Common sense knowledge
  - Reasoning capacity
  - Experience
- Computers have
  - No common sense knowledge
  - No reasoning capacity

Where does NLP fit in the CS taxonomy?



- NLP uses: algorithms, search, databases

Issues in syntax

- Shallow parsing—identify **basic structures**
  - "the dog chased the bear"

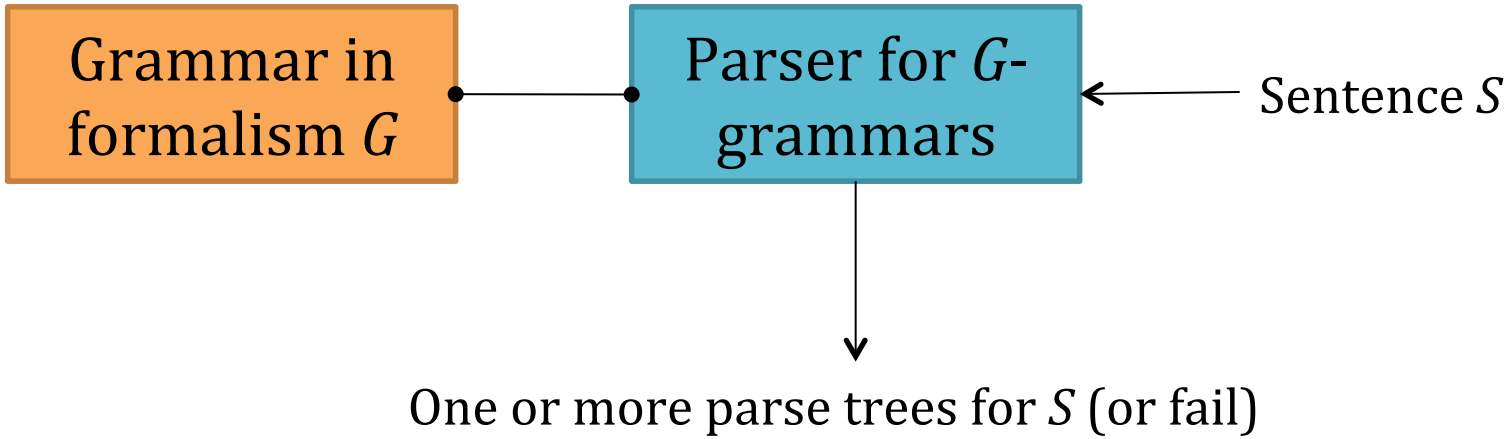
subject/noun phrase
"the dog"

predicate/verb phrase
"chased the bear"

- Deeper analysis
  - Who did what? (literal meaning)—**semantics**
  - The meaning in context—**pragmatics**
  - "the dog ate my homework"

Syntactic analysis (parsing)

- Uses formal grammar and parsing algorithm to find structure
  - Create parse trees from sentences



- Limitations
  - Explosion of number of parse trees
  - Inability to handle ungrammatical input

Grammars for parsing

- Grammar—specifies the **compositional structure** of complex messages
  - E.g., speech (linear), text (linear), music (two-dimensional)

- A **formal language** is a set of **strings** of **terminal** symbols (actual words)
- Each string in the language can be **analyzed/generated** by the grammar

- Grammar is a set of rewrite rules (productions)

$S \rightarrow NP\ VP$
$Article \rightarrow the \mid a \mid an \mid \ldots$
$NP \rightarrow \ldots$
$VP \rightarrow \ldots$
$\ldots$

Here ***S*** is the **sentence** symbol, ***NP*** and ***VP*** are **non-terminals**

Grammar formalisms

- Grammars are **generative**—categorized by **set of languages** they can generate/represent
  - Context free grammar
  - Unification grammar
  - Tree adjoining grammar
  - Dependency grammar
  - Combinatorial grammar

  - And more!

Natural language grammar elements

- A set of rules (or equivalent structures)
  - Expressed in terms of **lexical categories**
    e.g., noun, verb, etc.

- A **lexicon**
  - A list of allowable words with their parts of speech
  - Feature-based lexicons: noun [number singular]

Wumpus grammar

S → *NP VP*	[0.90]	I + feel a breeze
| *S Conj S*	[0.10]	I feel a breeze + and + it stinks
NP → *Pronoun*	[0.30]	I
| *Name*	[0.10]	John
| *Noun*	[0.10]	pits
| *Article Noun*	[0.25]	the + wumpus
| *Article Adjs Noun*	[0.05]	the + smelly dead + wumpus
| *Digit Digit*	[0.05]	3 4
| *NP PP*	[0.10]	the wumpus + in 1 3
| *NP RelClause*	[0.05]	the wumpus + that is smelly
VP → *Verb*	[0.40]	stinks
| *VP NP*	[0.35]	feel + a breeze
| *VP Adjective*	[0.05]	smells + dead
| *VP PP*	[0.10]	is + in 1 3
| *VP Adverb*	[0.10]	go + ahead
Adjs → *Adjective*	[0.80]	smelly
| *Adjective Adj*	[0.20]	smelly + dead
PP → *Prep NP*	[1.00]	to + the east
RelClause → *RelPro VP*	[1.00]	that + is smelly

Wumpus lexicon

Noun	→	**stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | …
Verb	→	**is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | …
Adjective	→	**right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] | …
Adverb	→	**here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | …
Pronoun	→	**me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | …
RelPro	→	**that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] | …
Name	→	**John** [0.01] | **Mary** [0.01] | **Boston** [0.01] | …
Article	→	**the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | …
Prep	→	**to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | …
Conj	→	**and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] | …
Digit	→	**0** [0.20] | **1** [0.10] | **2** [0.20] | **3** [0.20] | **4** [0.20] | …

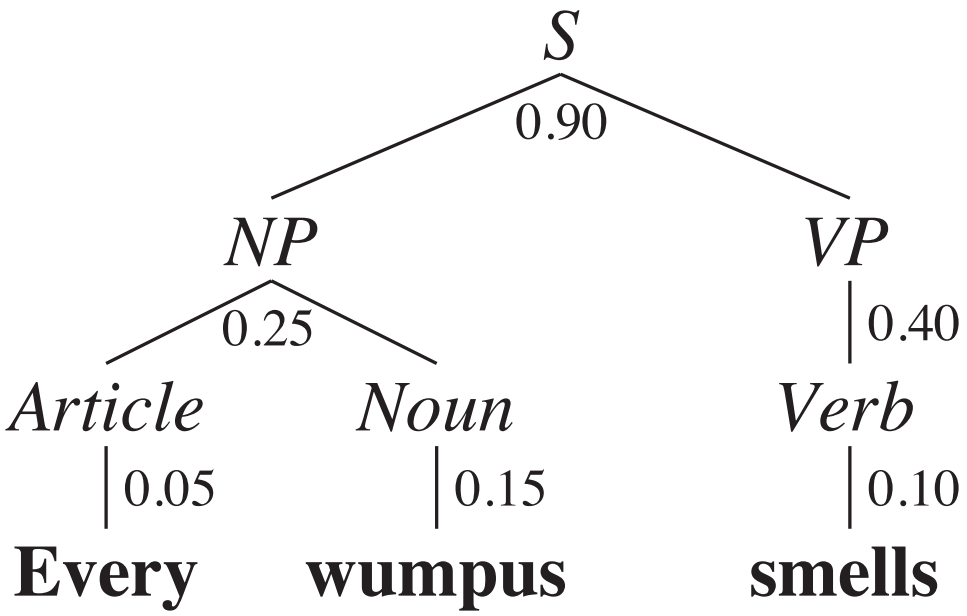
- About 40% of words in the NYTimes are not in a (large) dictionary—natural language is **productive**

Parsing with PCFGs

- Task of assigning correct trees to input strings
  - Tree covers **all and only the elements of the input** and has **an *S* a at the top**

- System may not be able to select the "correct" tree from among possible trees
  - Parsing involves search where choices must be made
  - Requires semantics to find the "right" tree!

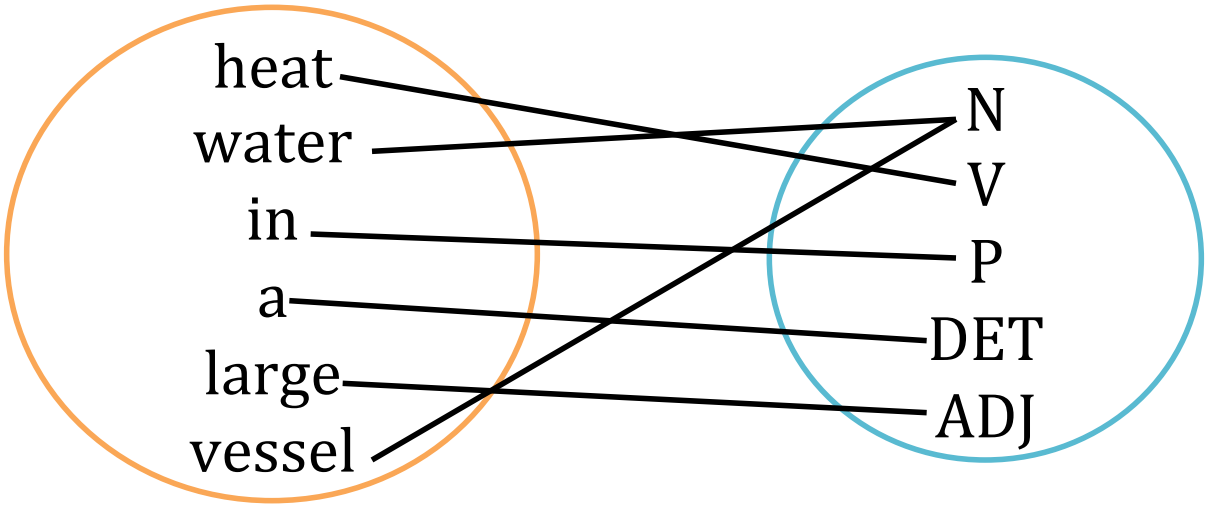
Wumpus parse tree



- Total probability of the tree
  $= 0.9 \times 0.25 \times 0.40 \times 0.05 \times 0.15 \times 0.10 = 0.0000675$

Part-of-speech tagging

- Process of assigning a part-of-speech (POS) to each word in a sentence



POS tagging example

Word	Tag
heat	verb (noun)
water	noun (verb)
in	prep (noun, adv)
a	det (noun)
large	adj (noun)
vessel	noun

What is POS tagging good for?

- Useful in
  - Information retrieval
  - Text to speech: **ob**ject (N) vs. ob**ject** (V);
    discount (N) vs. dis**count** (V)
  - Word sense disambiguation

- Useful as a preprocessing step of parsing
  - Unique tag to each word reduces the number of parses

Choosing a tagset

- Need to choose a **standard set of tags** to do POS tagging
  - One tag for each part of speech
- Could pick a very course tagset
  - N, V, Adj, Adv, Prep
- More commonly used set is **finer-grained**
  - E.g., the *Upenn TreeBank II* tagset has 36 word tags
    - PRP, PRP$, VBG, VBD, JJR, JJS…
    - (also has tags for phrases)
  - Even more finely-grained tagsets exist

Upenn TreeBank II word tags

- CC - Coordinating conjunction
- CD - Cardinal number
- **DT** - Determiner
- EX - Existential there
- FW - Foreign word
- **IN** - Preposition or subordinating conjunction
- **JJ** - Adjective
- JJR - Adjective, comparative
- JJS - Adjective, superlative
- LS - List item marker
- MD - Modal
- **NN** - Noun, singular or mass
- NNS - Noun, plural
- **NNP** - Proper noun, singular
- NNPS - Proper noun, plural
- PDT - Predeterminer
- POS - Possessive ending
- PRP - Personal pronoun
- PRP$ - Possessive pronoun
- RB - Adverb
- RBR - Adverb, comparative
- RBS - Adverb, superlative
- RP - Particle
- SYM - Symbol
- **TO** - to
- UH - Interjection
- VB - Verb, base form
- VBD - Verb, past tense
- VBG - Verb, gerund or present participle
- **VBN** - Verb, past participle
- VBP - Verb, non-3rd person singular present
- **VBZ** - Verb, 3rd person singular present
- WDT - Wh-determiner
- WP - Wh-pronoun
- WP$ - Possessive wh-pronoun
- WRB - Wh-adverb

Why is POS tagging hard?

- Ambiguity (more on this later…)
  - "**Plants**/**N** need light and water."
  - "Each one **plant**/**V** one."
  - "Flies like a flower"

    Flies: noun or verb?

    like: preposition, adverb, conjunction, noun, or verb?

    a: article, noun, or preposition?

    flower: noun or verb?

Methods for POS tagging

- **Rule-based** POS tagging
  - E.g., ENGTWOL (Voutilainen, 1995)—large collection (> 1000) of constraints on what sequences of tags are allowable
- Transformation-based tagging
  - E.g., Brill's tagger (Brill, 1995)—sorry, I don't know anything about this…
- **Stochastic** (Probabilistic) tagging
  - E.g., TNT (Brants, 2000)—we'll talk about this in more detail!

Stochastic POS tagging

- Based on probability of certain tag occurring, given various possibilities

- Necessitates a **training corpus**
  - Collection of sentences that have already been tagged
- Several such corpora exist
  - Brown University Standard Corpus of Present-Day American English (**Brown Corpus**)
    - About 1,000,000 words from a wide variety of sources
    - POS tags assigned to each

First approach

- Assign each word its **most likely** POS tag
- If w has tags $t_1,\ldots,t_k$ then use

$$P(t_i \mid w) = \frac{c(w, t_i)}{c(w, t_1) + \ldots + c(w, t_k)}$$

- where $c(w, t_i)$ = number of times w/t_i appears in the corpus
- Success: 91% for English!
- Example
  - heat::**noun/89**, **verb/5**

Second approach

- Given: sequence of words (i.e., a **sentence**) W s.t.

$$W = w_1, w_2, \ldots, w_n$$

  - E.g., W = heat water in a large vessel

- Assign sequence of tags T s.t.

$$T = t_1, t_2, \ldots, t_n$$

- Find T that **maximizes** $P(T \mid W)$

Practical Stochastic Tagger

- By Bayes Rule:

$$P(T \mid W) = \frac{P(W \mid T)\, P(T)}{P(W)} = \alpha\, P(W \mid T)\, P(T)$$

- So find T that maximizes $P(W \mid T)\, P(T)$
  - Chain rule:

    $P(T) = P(t_1)P(t_2 \mid t_1)\, P(t_3 \mid t_1, t_2)\, P(t_3 \mid t_1, t_2, t_3) \ldots P(t_n \mid t_1, \ldots, t_{n-1})$

  - **Markov assumption**: as an approximation, use:

    $P(T) \approx P(t_1)\, P(t_2 \mid t_1)\, P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})$

  - **Naïve Bayes assumption**: each word is dependent only on its own POS tag (given its POS tag, it is conditionally independent of the other words)

    $P(W \mid T) = P(w_1 \mid t_1)\, P(w_2 \mid t_2) \ldots P(w_n \mid t_n)$

- So

$$\mathbf{P(W \mid T)\, P(T) \approx P(w_1 \mid t_1)\, P(w_2 \mid t_2) \ldots P(w_n \mid t_n)\, P(t_1)\, P(t_2 \mid t_1)\, P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})}$$

Getting the conditional probabilities

- Want to compute

$$P(W \mid T)\, P(T) \approx P(w_1 \mid t_1)\, P(w_2 \mid t_2) \ldots P(w_n \mid t_n)\, P(t_1)\, P(t_2 \mid t_1)\, P(t_3 \mid t_2) \ldots P(t_n \mid t_{n-1})$$

- Let
  $c(t_i)$ = frequency of t_i in the corpus
  $c(w_i, t_i)$ = frequency of w_i / t_i in the corpus
  $c(t_{i-1}, t_i)$ = frequency of $t_{i-1}\ t_i$ in the corpus

- Then we can use

$$P(t_i \mid \text{ti-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})}$$

$$P(w_i \mid t_i) = \frac{c(w_i, t_i)}{c(t_i)}$$

POS tagging example

- Secretariat/NNP is/VBZ expected/VBN **to/TO race/VB** tomorrow/NN
  - to/TO race/???
- People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN **the/DT race/NN** for/IN outer/JJ space/NN
  - the/DT race/???
- For each word w_i, $t_i = \text{argmax}_t \, P(t \mid t_{i-1}) P(w_i \mid t)$
  - max(P(VB|TO) P(race | VB), P(NN|TO) P(race|NN))
- From the Brown corpus
  - P(NN | TO) = 0.021 P(race | NN) = 0.00041
  - P(VB | TO) = 0.34 P(race | VB) = 0.00003
- So
  - P(NN | TO) P(race | NN) = 0.021 × 0.00041 = 0.000007
  - **P(VB | TO) P(race | VB) = 0.34 × 0.00003 = 0.00001**

Semantic analysis

- Determine the **meaning** of language

- Importance of semantics?
  - Machine translations: wrong translations
  - Information retrieval: wrong information
  - Anaphora resolution: wrong referents

- Biggest challenge: lexical **ambiguity**—words are ambiguous
  - "plant" = industrial plant
  - "plant" = living organism

Why do we need semantics?

- Machine translation example
  - **The sea is home to millions of plants and animals**

    English → French translation (commercial MT system)

    Le mer est a la maison de billlion des usines et des animaux

    French → English

    The sea is at the home for billions of factories and animals

- Hmmm…

Lexical ambiguity

- Extreme case—two words with the **same spelling**
  - Wound, wound
- More frequent case—words that can be a noun, verb, adjective, etc.
  - Time, phone
- Many English words have multiple meanings even within one part of speech (POS)
  - Set, head, can, bear, …
- WordNet—public domain lexical-semantic net
  - **Demo!!**
- SemEval—a periodic "shared task" activity to evaluate semantic analysis tools

How to learn the meaning of words?

- How do we get a training set of semantic examples?

- From dictionaries: **word sense**
  - plant, works, industrial plant—(buildings for carrying on industrial labor; “They built a large plan to manufacture automobiles.”)

    plant, flora, plant life—(a living organism lacking the power of locomotion)

- Can these definitions help disambiguate all uses?
  - They are producing about 1,000 automobiles in the new plant.
  - The sea flora consists of 1,000 different plant species.
  - The plant was close to the farm.

How to learn the meaning of words? (cont.)

- Learn from annotated examples
  - Assume 100 examples containing “plant” previously tagged by a human
  - Train a learning algorithm to classify future instances of “plant”

- How to choose the learning algorithm?
- How to obtain the 100 tagged examples?

More issues in semantic analysis

- Reference resolution
  "Josh sold a book to Tom. **He** was happy."
  "Mary and I went out to dinner. **It** was fun."

- The pronoun (or other **anaphoric** noun phrase) may reference something that is implicitly mentioned by does not have a specific antecedent.

Pragmatics—semantics and **context**

- **Classical** view (pre-1953)—language consists of sentences that are true/false (like logic)
  - Why?
    To modify the beliefs of other agents

- **Modern** view (post-1953)—language is a form of action
  - Why?
    To change the actions of other agents

Speech act theory of pragmatics

- Examples: "I pronounce you husband and wife"
  "I sentence you to five years"

SITUATION

Speaker → **Utterance** → **Hearer**

- Speech acts achieve the speaker's goals:
  - Inform "There is a pit in front of you"
  - Query "Can you see the gold?"
  - Command "Pick it up"
  - Promise "I'll share the gold with you"
  - Acknowledge "OK"

- Speech act planning requires knowledge of
  - Situation
  - Semantic and syntactic conventions
  - Hearer's goals, knowledge base, rationality

Machine translation

- Text to text machine translations
- Speech to speech machine translations

- Most of the work has addressed pairs of widely spread languages like English-French, English-Chinese

Issues in machine translations

- How to translate text?
  - Learn from previously translated data
  - Need **parallel corpora**
    - French-English, Chinese-English have the Hansards (transcripts of parliamentary debates)
- Reasonable translations?
  - Application dependent—do we need the general idea or precise language?
- Lack of data = lack of tools
  - Chinese-Hindi—no translator available!

Speech to speech translation challenges

- Stages in communication (informing)
  - **Intention** — *S* wants to inform *H* that ***P***
  - **Generation** — *S* selects words ***W*** to express ***P*** in context ***C***
  - **Synthesis** — *S* utters words ***W***

  - **Perception** — *H* perceives ***W'*** in context ***C'***
  - **Analysis** — *H* infers possible meanings $P_1,\ldots,P_n$
  - **Disambiguation** — *H* infers intended meaning P_i
  - **Incorporation** — *H* incorporates P_i into KB

- How could this go wrong?
  - Insincerity (S doesn't believe ***P***)
  - Speech wreck ignition (**recognition!**) failure
  - Ambiguous utterance
  - Differing understanding of current context ($C \neq C'$)

Information extraction

- Extract information and detect new patterns in data
  - Detect hacking, hidden information, etc.
- Government and military put a lot of money into IE research!

- Example:
  "There was a group of about 8-9 people close to the entrance on Highway 75"
  - **Who?** "8-9 people"
  - **Where?** "Highway 75"

Information retrieval

- General model—Huge collection of **texts** and a **query**
- Tasks
  - Find **documents** that are relevant to the given query—Create an index, like the index in a book
  - Retrieve specific information—**question answering**
    "What is the height of Mount Everest?" 11,000 feet

- Types of models
  - Vector-space models
  - Boolean models

- Examples—Google, Yahoo, etc.

Cross-language information retrieval

- Find information across languages!

- Example:
  "What is the minimum age requirement for car rental in Italy?"
  - Search English and also Italian texts for "eta minima per noleggio macchine"

- Integrate large number of languages and into performant IR engines