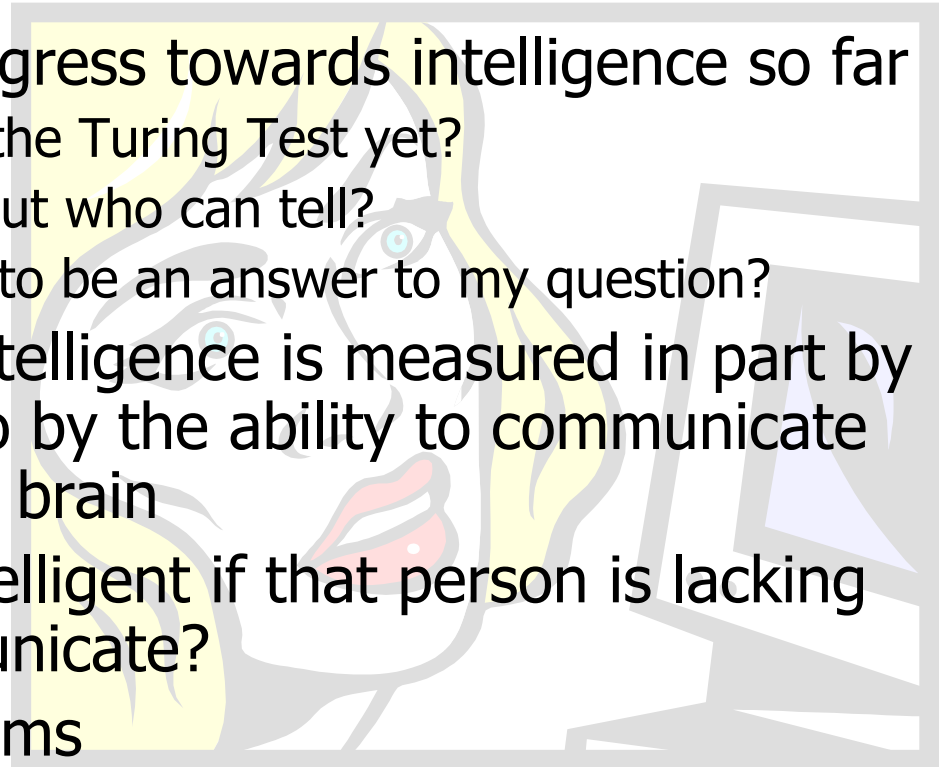


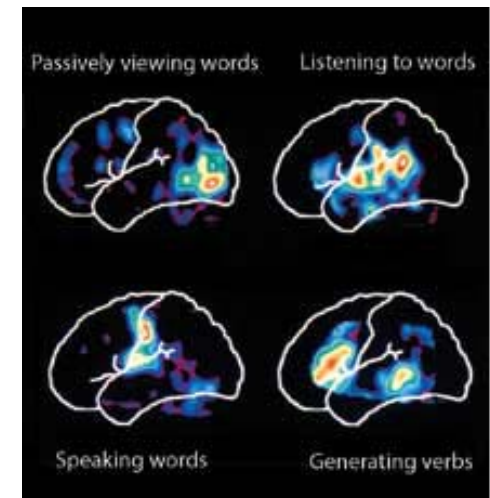
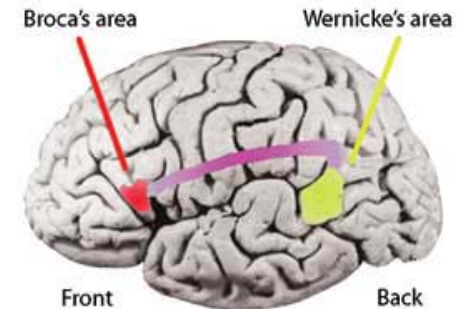
Communication Agents

- Let's look at our progress towards intelligence so far
 - Q: Have we passed the Turing Test yet?
 - A: We might have, but who can tell?
 - Q: Is that supposed to be an answer to my question?
- Issue here is that intelligence is measured in part by brainpower, but also by the ability to communicate the thoughts of that brain
- Is a person truly intelligent if that person is lacking the ability to communicate?
- Consider stroke victims
 - Intelligence exists, but disturbed blood flow to the brain can cause paralysis and **aphasia** (disruption of speech ability, from the Greek for "lack of speech")
 - How do you assess the stroke victim's mental abilities?



More on Aphasia

- Expressive aphasia
 - caused by damage to the left frontal lobe of the brain
 - causes difficulty in the production of speech
 - Example: Tan's brain & Broca's area
- Receptive aphasia
 - caused by damage to the temporal & parietal lobes (Wernicke's area)
 - impairs understanding and production of speech content
- Even though communication disorders do not imply a lack of intelligence, they do affect perceived intelligence
 - Unfortunate, but Turing test still fails



The Natural Language Problem

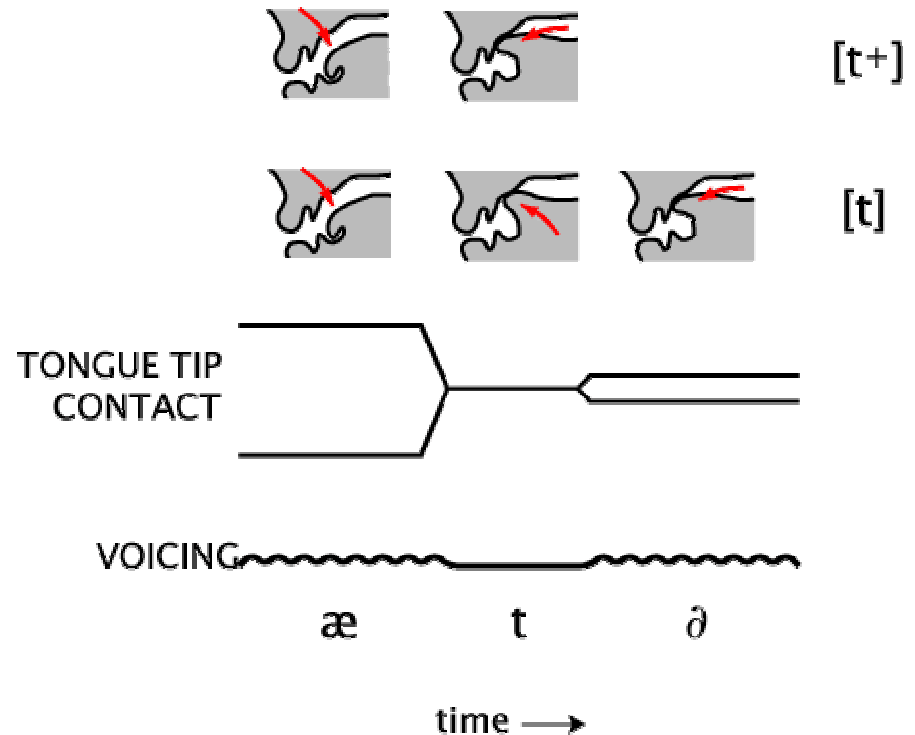
- Language is the task of translating auditory input into knowledge and back again.
- Perception stage (hard)
 - speech recognition (speech signals → words)
 - syntactic analysis (words → structure & roles)
 - semantic processing (structure & roles → meaning)
- Generation stage (easier)
 - language generation (meaning → words)
 - speech synthesis (words → speech signals)

“How to wreck a nice beach”

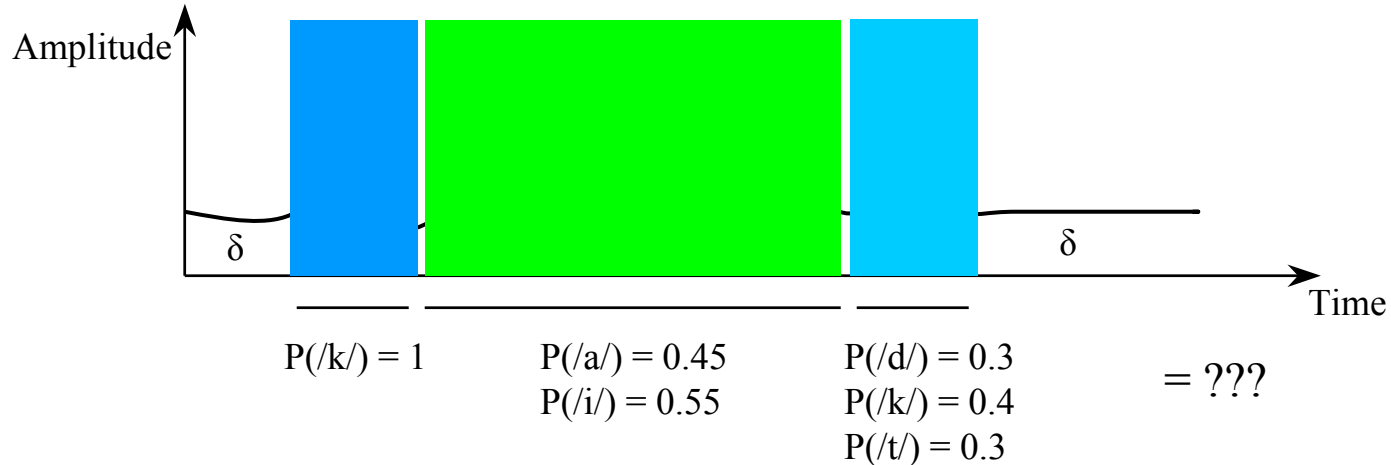
- Suppose you had a speech signal. How would you figure out what words the speech signal represents?
 - Usually, each portion of speech signal matches with a basic sound, called a **phone**, or the mental abstraction of that sound, called a **phoneme** (e.g. /k/, /a/ or /t/). Several phones can match to a single phoneme, which are considered **allophones** of each other.
 - What happens when a phone doesn't match clearly with a particular phoneme? Is it sufficient to choose the closest available match?
 - Certain classes of phones can be easily mistaken for one another
 - e.g. fricatives (/f/, /th/, /v/), nasals (/m/, /n/, /ŋ/), plosives (/p/, /b/, /t/, /d/, /k/, /g/)

Recognizing Phonemes

- To determine what sounds are being spoken, one must not only look at the phoneme possibilities, but also the context
 - requires large sample of labeled speech sounds to calculate the phoneme probabilities
- Requires:
 - probability of phoneme, given speech signal (S or σ)
 - probability of phoneme, given previous phoneme



Phoneme Sequences



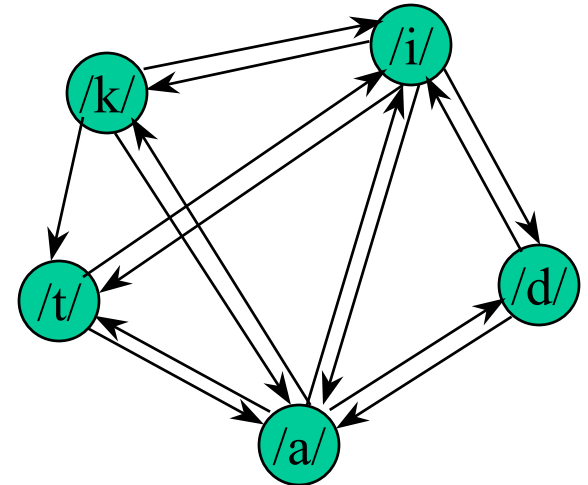
- Example: "kick" or "cat"?
 - From examining the individual probabilities alone, one would assume that this signal corresponds to the word "kick", since the /i/ and /k/ phonemes are the most likely
 - But if we consider the context...?

Transition Probabilities

- A table of **transition probabilities** must be obtained empirically from large labeled datasets

(previous phoneme) \ (current phoneme)

	/a/	/d/	/i/	/k/	/t/
/a/	0.4	0.2	0.01	0.15	0.24
/d/	0.6	0	0.4	0	0
/i/	0.05	0.2	0.4	0.15	0.2
/k/	0.5	0	0.4	0	0.1
/t/	0.5	0	0.5	0	0



- Transition table is similar to a graph's transition matrix

Phoneme Sequences (cont'd)

- Looking for highest probability of phonemes given speech signal
 - = $P(/k/, /i/, /k/ \mid \sigma)$ → “kick”, for example
 - = $P(/k/ \mid \sigma_1) * P(/i/ \mid \sigma_2) * P(/k/ \mid \sigma_3) * P(/k/) * P(/i/ \mid /k/) * P(/k/ \mid /i/)$
- So calculation of phoneme sequence probabilities produces the following:
 - $P(\text{“kick”}) = (1)(0.4)(0.55)(0.15)(0.4) = 1.32\%$
 - $P(\text{“kit”}) = (1)(0.4)(0.55)(0.2)(0.3) = 1.32\%$
 - $P(\text{“kid”}) = (1)(0.4)(0.55)(0.2)(0.3) = 1.32\%$
 - $P(\text{“kack”}) = (1)(0.5)(0.45)(0.15)(0.4) = 1.35\%$
 - $P(\text{“cad”}) = (1)(0.5)(0.45)(0.2)(0.3) = 1.35\%$
 - $P(\text{“cat”}) = (1)(0.5)(0.45)(0.24)(0.3) = \mathbf{1.62\%}$

Syntactic Analysis

- Assuming that a sentence's words have been recognized correctly, how do you figure out what the words mean?
- First, one needs to figure out the syntactic role of each word in the phrase (also called **tokens**)
 - Java analogy: the `if` keyword has a different meaning if it comes at the beginning of a statement, or after the `else` keyword, or within a set of quotes.
 - The structure of a sentence and the placement of a word within that structure reveals information about the role of the word, and thus its meaning

Part-Of-Speech Tagging

- One of the more crucial elements of the syntactic stage is the concept of assigning part-of-speech (POS) tags to the words in a sentence.
- The classic example of part-of-speech tagging:

"Time flies like an arrow"

- Does this phrase mean that:
 - time flies by in the same manner as an arrow?
 - certain insects called "time flies" enjoy arrows?
 - you must measure the flies' timing the way an arrow would?
- Syntactic processing determines these interpretations, and helps to determine the most likely POS tags

Grammars & Parsing

- One way to determine the POS tags for a sentence is to treat the sentence as the product of the **grammar** and **lexicon** of a language
 - the grammar is a set of decomposition rules that show how a given symbol can be decomposed into component symbols
 - the highest-level symbol is usually the sentence; the lowest-level symbols are usually the POS tags for individual words
 - symbols that cannot be decomposed any farther are called **terminal** symbols; the others are called **non-terminals**
 - a lexicon is like a dictionary, storing valid words in the language's vocabulary and their possible POS tags
- The act of decomposing a sentence's components recursively to the word level is called **parsing**

Grammars & Parsing (cont'd)

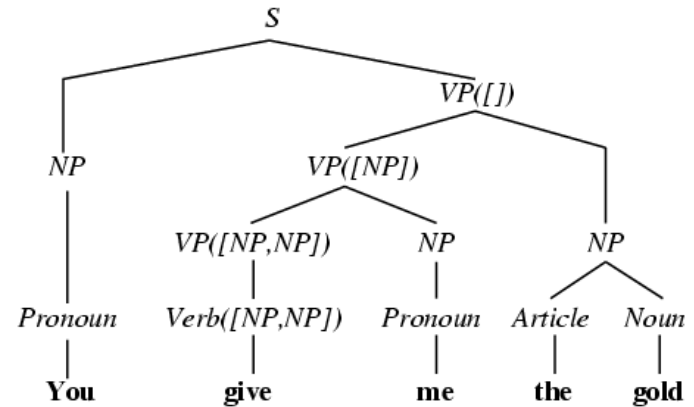
- Parsing generally assumes the use of a **context-free grammar** (CFG)
 - Review: CFG means that the rules of the grammar are of the form $V \rightarrow w$, (V is a non-terminal of the grammar, and w is a string of terminals and/or non-terminals), where this transformation can take place independent of the symbols around V
 - CFGs are generally in **Backus-Naur Form** (BNF), just like the rules in the knowledge base
- Similar to KB rule decomposition, the act of parsing a phrase leads to the phrase's **parse tree**, a tree structure where:
 - The internal nodes are non-terminals in the parse
 - The leaves are terminal POS tags
 - The words of the sentence have a 1-to-1 mapping to the leaves of the parse tree

Parsing Methods

- Creating the parse tree can be done in many ways, similar to resolving KB facts

- Two basic approaches:

- **Top-down** = starting from the *S* symbol (representing sentence in this context, not a speech signal), search for a decomposition that results in a tree with the sentence's words as the leaves (each state represents a possible decomposition of the original *S* symbol)
- **Bottom-up** = given the sequence of words, search for a unification of adjacent components into non-terminals until a single tree is created with *S* as the root. Unification takes place by finding sequences of terminals or non-terminals that fit the right-hand side of a grammar rule, and replacing that sequence with the non-terminal on the left-hand side.



Parsing Issues

- Top-down parsers are susceptible to **left-recursive** and **right-recursive rules**, since the parsers could get stuck, decomposing the same rule indefinitely
 - **LL parsers** : decompose a symbol from left to right.
 - Gets stuck on rules like: $X \rightarrow X Y$
 - **LR parsers** : decompose a symbol from right to left
 - Gets stuck on rules like: $X \rightarrow Y X$
- All parsers that use search strategies to parse a sentence are vulnerable to garden-path sentences

"The horse raced past the barn fell"

- garden-path sentences require a great deal of backtracking

Chart Parsers

- To avoid backtracking and combinatorial explosions while searching, **chart parsers** use a data-driven approach to determine which bottom-up rules to explore
- Main difference is that chart parsers store all partly-explored rules in a knowledge base called a **chart**
- General algorithm:
 1. going from left to right, analyze each word in the sentence
 2. check the current word's POS
 - a. if the current word's POS is the first element in the sequence of a decomposition rule, add that partly-completed rule to the chart
 - b. if the current word's POS is the next element in the sequence of a partly-completed rule that ends at the word's position, remove that element from the partly-completed sequence
 - i. if the partly-completed sequence is now fully-completed, repeat Steps 2a and 2b for the non-terminal on the left-hand side of the decomposition rule
 3. if any chart entries have the S non-terminal as their root and run from the beginning to the end of the sentence, search the chart rules to construct the parse tree

Chart Parsers (cont'd)

- Chart parsers consume $O(n^2)$ space and $O(n^3)$ time, the best that can be achieved by a parser that can handle all CFGs
 - Note: this is just to determine if a parse exists. To actually obtain the parse tree afterwards could potentially be $O(2^n)$, although it generally isn't.
- **Earley algorithm** = a variation on the chart parser that uses dot notation to indicate how complete a grammar rule is
- **CKY parsers** = use a variation on the chart to see if certain grammar structures exist, and whether a sentence is among those structures

S						
	VP					
S						
	VP			PP		
S		NP			NP	
NP	V, VP	Det	N	P	Det	N
Steve	eats	his	Jello	with	a	straw

Limits of Syntactic Analysis

- Problems with ambiguous parses
 - Example: Newspaper headlines
 - “Eye drops off shelf”
 - “Squad helps dog bite victim”
 - “Dealers will hear car talk at noon”
 - “Enraged cow injures farmer with ax”
 - “Two sisters reunite after eighteen years at checkout counter”
- Reference resolution (**anaphora**)
 - More newspaper headlines
 - “Grandmother of 8 makes hole in one”
 - “Two Soviet ships collide - one dies”



Semantic Processing

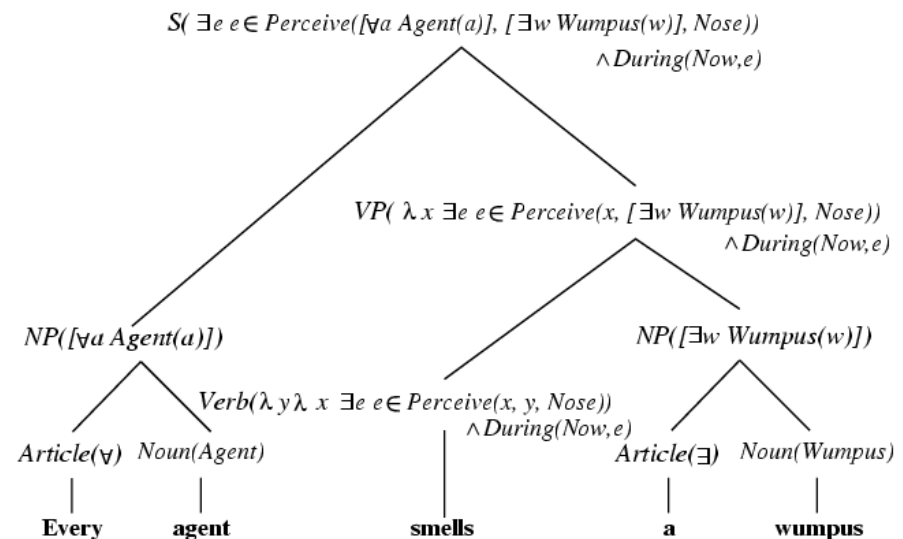
- Parsing produces syntactic roles, but only produces a limited intuition of the meaning of the sentence
- For that, the system must also obtain an understanding of the sentence, based on the structure
- Semantic understanding is important for many important **natural language processing** (NLP) problems
 - interpreting commands
 - question answering
 - text summarization
 - automatic translation
 - Example: *"The spirit is willing but the flesh is weak"*
 → *"The wine is good but the meat has gone bad"*

Basic Semantic Analyzers

- ELIZA (1966)
 - Natural language-based “therapist”
 - rearranges and substitutes certain phrases to emulate a Rogerian psychotherapist
- WordNet (Princeton University)
 - “semantic lexicon” for English
 - contains ~150,000 words grouped into synonym categories, with semantic definitions for each category
 - parse tree disambiguates part-of-speech, helps define deeper semantic context for that word
 - problematic when distinguishing between two different meanings for same part-of-speech → need semantic context

Adding Semantic Values

- The area of language semantics is still largely unexplored, with no widely accepted techniques
- For basic phrases, the parse tree can be explored to produce a propositional logic statement
 - verbs become relations
 - nouns become literals
 - articles become quantifiers
 - prepositions also become relations
 - nested branches produce nested logic statements
 - must maintain the logical ordering



Information Extraction

- An alternative to producing a unified understanding of a sentence is to extract only the information needed from the text
 - Example: job postings
 - rather than understand a company's entire web page, look for patterns in the text that, when parsed, give a phrase or structure that resembles a job listing
- **Information extraction** looks for cues in the text in order to locate the interesting passage
- The passage is then parsed with that domain in mind, to obtain more precise information

General Perception Factors

- Domain-specific knowledge
 - When recognizing speech or parsing text, a more accurate analysis is possible when the domain of the sentence is known
 - For speech, training a model on a particular speaker greatly enhances a system's performance
 - For parsing, models trained on the Penn Treebank will have less accuracy when parsing the Wall Street Journal dataset
- Total perception model
 - Rather than perceive language information in stages (speech, syntax, semantics), analyse semantics directly from speech signals, to reduce noise between stages
 - Good idea in theory. Major obstacle: sparse training data. Even if lots of tagged speech-to-semantics data existed, there wouldn't be enough examples to train a thorough model.

Language Generation

- Much easier than perceiving & interpreting language
- Same structures used, but in reverse
 - determine semantic information for utterance
 - assuming propositional logic statement, build parse tree by recursively expanding propositional statement
 - bind terminal nodes of tree to proper words
 - return sentence
- Easier problem because output sentences have much less variation than input sentences
 - no unexpected vocabulary
 - consistent grammar structure