

# CSE 842

## Natural Language Processing

### Lecture 1: Introduction

1/10/2011

CSE842, Spring 2011, MSU

1

## What is NLP?

Dave Bowman: Open the pod bay doors, HAL.



1/10/2011

CSE842, Spring 2011, MSU

2

## What is NLP?

Dave Bowman: Open the pod bay doors, HAL.



HAL: I'm sorry Dave. I'm afraid I can't do that.

- Fundamental goal: *deep* understand of broad language
  - Not just string processing or keyword matching!
- Applications: information extraction, question answering, summarization, machine translation

1/10/2011

CSE842, Spring 2011, MSU

3

## What is NLP

- The study of human languages and how they can be represented computationally and analyzed and generated algorithmically
  - *The dog likes bacon.* --> like (dog, bacon)
  - like (dog, bacon) --> *The dog likes bacon*
- Studying NLP involves studying natural language, formal representations, and algorithms for their manipulation

1/10/2011

CSE842, Spring 2011, MSU

4

## NLP becomes increasingly important

- Play an important role in curbing information explosion on the internet
  - Information extraction
  - Text summarization
- Used for building natural interfaces to databases, machine translations, etc.
  - Intelligent conversational agents

## Why NLP is Difficult: Multidisciplinary

- **Linguistics**: how words, phrases, and sentences are formed.
- **Psycholinguistics**: how people understand and communicate using human language
- **Philosophy**: relates to the semantics of language; notation of meaning. NLP requires considerable knowledge about the world

## Why NLP is Difficult: Multidisciplinary

- **Computer Science**: deals with model formation and implementation
- **Mathematics and Statistics**: deals with probabilities, statistical distribution and hypothesis testing of language phenomena
- **Artificial Intelligence**: relates to knowledge representation and reasoning

## Language Ambiguities

*I made her duck.*

- How many different interpretations does the above sentence have?
- How can each ambiguous piece be resolved?

## Language Ambiguities

- Lexical ambiguity: when a word has more than one part of speech

*Rice flies like sand.*

- Structural ambiguity:

*John saw the boy with a telescope*

*John saw the boy with a telescope*

## Language Ambiguities

- Semantic ambiguity: when a word has more than one possible meanings

*John **killed** the wolf.*

*Bill **killed** the project.*

*Mary **killed** Jane. (at tennis or murdered her?)*

## Basic levels of language processing

- Phonetics: how words are related to the sounds that realize them.
- Morphology: how words are constructed.  
*beauty, beautiful*
- Syntax: how words can be put together to form correct sentences, and the role of each plays in the sentence. *John likes Mary*

## Basic levels of language processing

- Semantics: the meaning of words and sentences  
*bass fishing, bass playing*
- Discourse: how the meaning of words and sentences is affected by the surrounding text or utterances  
*Mary bought a new computer yesterday. She likes it very much. (pronoun resolution)*
- Pragmatics: how sentences are used in different situations (contexts)  
*Mary grabbed her umbrella*  
A) *It is a cloudy day*  
B) *She was afraid of dogs*

## Representations and Algorithms for NLP

- Representations: formal models used to capture linguistic knowledge
- Algorithms manipulate representations to analyze or generate linguistic phenomena

## NLP Representations

- State Machines
  - FSAs, FSTs, HMMs
- Rule Systems
  - CFGs, Unification Grammars, Probabilistic CFGs
- Logic-based Formalisms
  - 1<sup>st</sup> Order Predicate Calculus, Temporal and other Higher Order Logics
- Models of Uncertainty
  - Bayesian Probability Theory

## The Turing Test

- Alan Turing: the *Turing test* (for intelligence)
- Three participants: a computer and two humans (one is an interrogator)
- Interrogator's goal: to tell the machine and human apart
- Machine's goal: to fool the interrogator into believing that a person is responding
- Other human's goal: convince the interrogator that the other participant is the machine

*Q: Please write me a sonnet on the topic of the Forth Bridge.*  
*A: Count me out on this one. I never could write poetry.*  
*Q: Add 34957 to 70764.*  
*A: 105621 (after a pause)*

## The Turing Test

- Alan Turing: the *Turing test* (for intelligence)
- Three participants: a computer and two humans (one is an interrogator)
- Interrogator's goal: to tell the machine and human apart
- Machine's goal: to fool the interrogator into believing that a person is responding
- Other human's goal: convince the interrogator that the other participant is the machine

Eliza: <http://www.manifestation.com/neurotoys/eliza.php3>

## Examples of NLP Applications

- dialogue systems
    - for automated service: communicator
    - for education and entertainment:
- (CSE891 Language and Interaction)
- speak to your appliances
  - automated speech recognition and text-to-speech in vehicles.
  - generate weather reports in two languages
  - translate Web pages into different languages
    - [http://translate.google.com/translate\\_t#](http://translate.google.com/translate_t#)
    - <http://www.systransoft.com>
  - grade essays
  - question answering
  - Text analytics: mining textual documents (e.g., event tracking, political opinion tracking, social network analysis, etc. )

## Some brief history

- Foundational insights (40's and 50's): automaton (Turing), probabilities, information theory (Shannon), formal languages (Backus and Naur), noisy channel and decoding (Shannon), first systems (Davis et al., Bell Labs)
- Two camps (57-70): symbolic and stochastic. Transformation grammar (Harris, Chomsky), artificial intelligence (Minsky, McCarthy, Shannon, Rochester), automated theorem proving and problem solving (Newell and Simon)  
Bayesian reasoning (Mosteller and Wallace)  
Corpus work (Kučera and Francis)

## Some brief history

- Four paradigms (70-83): stochastic (IBM), logic-based (Colmerauer, Pereira and Warren, Kay, Bresnan), NLU (Winograd, Schank, Fillmore), discourse modelling (Grosz and Sidner)
- Empiricism and finite-state models redux (83-93): Kaplan and Kay (phonology and morphology), Church (syntax)
- Late years (94-99): strong integration of different techniques, different areas (including speech and IR)
- More recent (00-present): rise of machine learning

## Topics covered

- Three major parts:
  - Linguistic, mathematical, and computational background
  - Levels of linguistic processing: morphology, syntax, semantics, and discourse
  - Applications: information extraction, question answering, summarization, and machine translation.

## Goals

- Three major goals:
  - Learn the basic principles and theoretical issues underlying natural language processing
  - Learn techniques and tools used to develop practical, robust systems
  - Gain insight into many open research problems in natural language

## Logistics

- Instructor: Joyce Chai
- Office: 2138 Engineering Building
- Office Hours: Monday: 2:30-4:30pm or by appointment
- Class Website:
  - Angel system: homework, discussion forum, etc.
  - Course directory ~cse842: data (log onto CSE machine and check the directory ~cse842)
  - <http://www.cse.msu.edu/~cse842/>: syllabus and classnotes

## Prerequisites

- Programming languages
  - use your favorite programming language for homework assignments and the final project.
- Some knowledge of probability and statistics
- Exposure to Linguistics (not essential)

## Grading

- Participation and Quizzes (5%)
- Three homework assignments: 60% (20% each)
  - Include a written part and a programming part
- Final project (35%)
  - Individual project or a group project with at most 2 people.
  - Project proposal (5%)
  - Presentation – 10 minutes (10%)
  - Final paper (style sheet will be provided) (20%)
  - (A list of potential topics will be provided.)

# Today

- Review some of the simple representations and ask ourselves how we might use them to do interesting and useful things
  - Regular Expressions
  - Finite State Automata

# Regular expressions

- A formula in a special language that is used for specifying simple classes of strings
  - A string is a sequence of symbols
  - For text-based search, a string is a sequence of alphanumeric characters (letters, numbers, spaces, tabs, and punctuation)
- Can be used to specify search strings and define a language in a formal way.

## Uses of Regular Expressions in NLP

- Simple but powerful tools for large corpus analysis and ‘shallow’ processing
  - What word is most likely to begin a sentence?
  - What word is most likely to begin a question?
  - ...

## Basic Regular Expression Patterns

- Case sensitive
- Perl-based syntax (slightly different from other notations for regular expressions)
- Disjunctions **[abc]**
- Ranges **[A-Z]**
- Negations **[^ss]**
- Optional characters **?, +, and \***
- Wild cards **.**
- Anchors **^** and **\$**, also **\b** and **\B**
- Disjunction, grouping, and precedence **|**

## Writing correct expressions

- Exercise: write a Perl regular expression to match the English article “the”:

```
/the/  
/[tT]he/  
/\b[tT]he\b/
```

## A more complex example

- Exercise: Write a regular expression that will match “any PC with more than 500MHz and 32 Gb of disk space for less than \$1000”:

```
/$[0-9]+/  
/$[0-9]+\.[0-9][0-9]/  
/\b$[0-9]+(\.[0-9][0-9])?\b/  
/\b[0-9]+ *(MHz|[Mm]egahertz|Ghz|  
[Gg]igahertz)\b/  
/\b[0-9]+ *(Mb|[Mm]egabytes?)\b/  
/\b[0-9](\.[0-9]+) *(Gb|[Gg]igabytes?)\b/
```

## Substitutions and memory

- Substitutions (Transduction)

```
s/colour/color/
```

- Memory (\1, \2, etc. refer back to matches)

35 boxes => <35> boxes: 

```
s/([0-9]+)/<\1>/
```

```
s/regexp1/pattern/
```

```
s/I am feeling (.+)/You are feeling \1?/
```

```
s/I gave (.+) to (.+)/Why would you give \2 \1?/
```

## Eliza [Weizenbaum, 1966]

```
User: Men are all alike  
ELIZA: IN WHAT WAY  
User: They're always bugging us about something or other  
ELIZA: CAN YOU THINK OF A SPECIFIC EXAMPLE?  
User: Well, my boyfriend made me come here  
ELIZA: YOUR BOYFRIEND MADE YOU COME HERE  
User: He says I'm depressed much of the time  
ELIZA: I AM SORRY TO HEAR THAT YOU ARE DEPRESSED
```

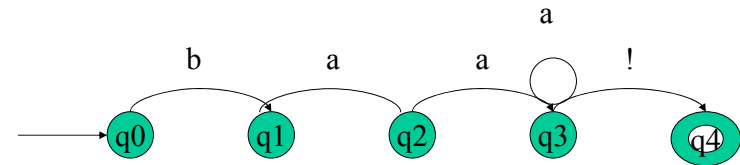
# Eliza-style regular expressions

- Step 1: replace first person references with second person references
- Step 2: use additional regular expressions to generate replies
- Step 3: rank possible transformations

```
s/. * YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/. * YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/. * all .*/IN WHAT WAY/
s/. * always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

# Finite State Automata

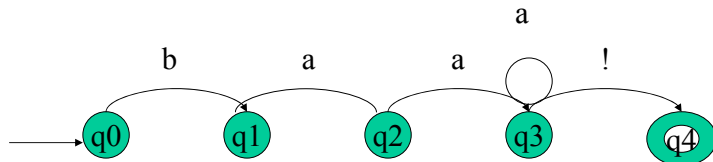
- FSAs implement regular expressions
  - /baa+!/



- Directed graph with labeled nodes and arc transitions**
- Five states:** q0 the start state, q4 the final state, 5 transitions

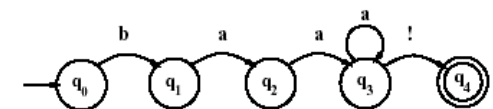
# Formal Definition

- FSA is a 5-tuple consisting of
  - Q: set of states {q0,...q4}
  - Σ: an alphabet of symbols {a,b,!}
  - q0: a start state
  - F: a set of final states in Q {q4}
  - δ(q,i): a transition function mapping Q x Σ to Q



# State Transition Table

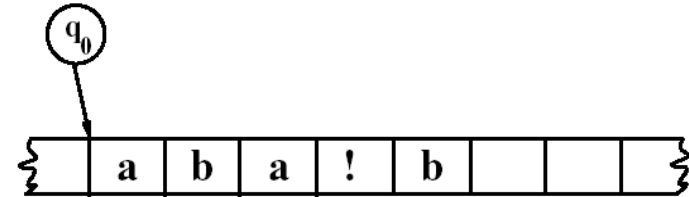
State	Input		
	b	a	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4:	∅	∅	∅



# Recognition

- A process of determining whether or not a given input should be accepted by a given machine
- A process of determining whether or not a given input matches a particular regular expression
- Recognition is viewed as processing an input written on a tape.

# Tape



# D-RECOGNIZE

```
function D-RECOGNIZE (tape, machine) returns accept or reject
index  $\leftarrow$  Beginning of tape
current-state  $\leftarrow$  Initial state of machine
loop
  if End of input has been reached then
    if current-state is an accept state then
      return accept
    else
      return reject
  elseif transition-table [current-state, tape[index]] is empty then
    return reject
  else
    current-state  $\leftarrow$  transition-table [current-state, tape[index]]
    index  $\leftarrow$  index + 1
end
```

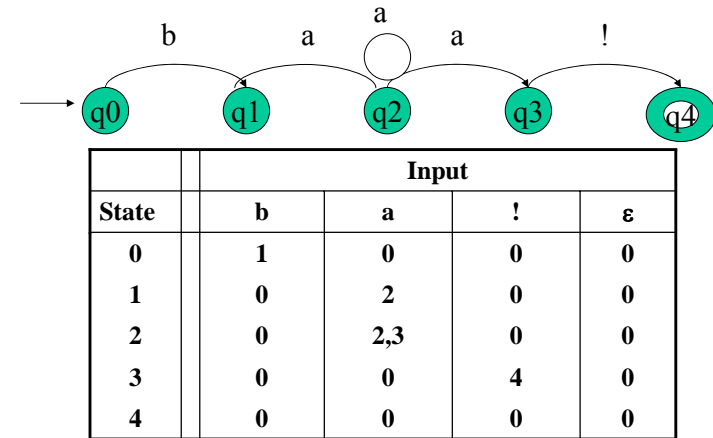
# Key points about D-Recognize

- Deterministic means the code always knows what to do at each point in the process.
- D-recognize is essentially a crude table driven interpreter.
- The recognition code is universal to all FSAs. Only need to change alphabet and the table when change to a new formal language. The code stays the same.

# Recognition as Search

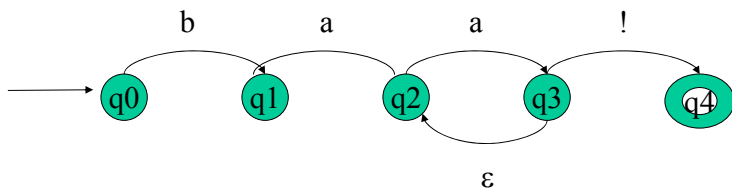
- You can view this algorithm as a kind of *state-space search*.
- States are pairings of tape positions and state numbers.
- Goal state is a pairing with the end of tape position and a final accept state

# Non-Deterministic FSAs



# Non-Deterministic FSAs

Another way to introduce non-deterministic FSA is to use  $\epsilon$ -transitions



Key point:  $\epsilon$ -transitions do not examine/advance the tape

# Problems of Non-Determinism

- At any choice point, we may follow the wrong arc
- Potential solutions:
  - Save **backup** states at each choice point
  - Look-ahead** in the input before making choice
  - Pursue alternatives in **parallel**
- ND-Recognize: state-space search algorithms, systematically search all possible paths
  - Depth-first search
  - Breadth-first search

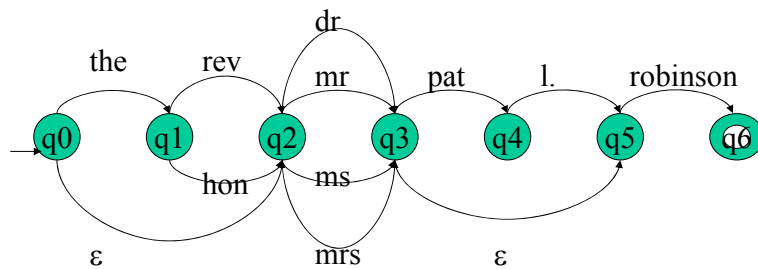
## NFSA to FSA

- Non-deterministic machines can be converted to deterministic ones with a fairly simple construction
- That means that they have the same power;
  - non-deterministic machines are not more powerful than deterministic ones in terms of the languages they can accept

## Generative Grammars and Formal Languages

- A formal language is a set of strings composed of symbols from a finite set of symbols.
- FSAs (and regular expressions) define formal languages (without explicitly enumerating the set)
- FSAs can be viewed as generators of formal languages as well as acceptors.

## FSAs as Grammars for Natural Language



Can you use a regex to capture this too?

## Regular Language

- The class of languages characterizable by regular expressions
- Given alphabet  $\Sigma$ , the regular language over  $\Sigma$  is:
  - The empty set  $\emptyset$  is a regular language
  - $\forall a \in \Sigma \cup \epsilon, \{a\}$  is a regular language
  - If  $L1$  and  $L2$  are regular languages, then so are:
    - $L1 \cdot L2 = \{xy | x \in L1, y \in L2\}$ , **concatenation** of  $L1$  &  $L2$
    - $L1 \cup L2$ , the **union** of  $L1$  and  $L2$
    - $L1^*$ , the **Kleene closure** of  $L1$

# Regular Language

- Regular expressions
- FSA
- Regular language: the language that can be defined by regular expressions (or FSAs)

