

CSCI 5832

Natural Language Processing

Lecture 2
Jim Martin

1/23/07

CSCI 5832 Spring 2007

1

Today 1/18

- Knowledge of language
- Ambiguity
- Models and algorithms
- Generative paradigm
- Finite-state methods

1/23/07

CSCI 5832 Spring 2007

2

Categories of Knowledge

- Phonology
- Morphology
- Syntax
- Semantics
- Pragmatics
- Discourse

Each kind of knowledge has associated with it an encapsulated set of processes that make use of it.

Interfaces are defined that allow the various levels to communicate.

This usually leads to a pipeline architecture.

Ambiguity

- *I made her duck*

Dealing with Ambiguity

- Four possible approaches:
 - Tightly coupled interaction among processing levels; knowledge from other levels can help decide among choices at ambiguous levels.
 - Pipeline processing that ignores ambiguity as it occurs and hopes that other levels can eliminate incorrect structures.

1/23/07

CSCI 5832 Spring 2007

5

Dealing with Ambiguity

- Probabilistic approaches based on making the most likely choices
- Don't do anything, maybe it won't matter
 - *We'll leave when the duck is ready to eat.*
 - *The duck is ready to eat now.*
 - Does the ambiguity matter?

1/23/07

CSCI 5832 Spring 2007

6

Models and Algorithms

- By models I mean the formalisms that are used to capture the various kinds of linguistic knowledge we need.
- Algorithms are then used to manipulate the knowledge representations needed to tackle the task at hand.

1/23/07

CSCI 5832 Spring 2007

7

Models

- State machines
- Rule-based approaches
- Logical formalisms
- Probabilistic models

1/23/07

CSCI 5832 Spring 2007

8

Algorithms

- Many of the algorithms that we'll study will turn out to be transducers; algorithms that take one kind of structure as input and output another.
- Unfortunately, ambiguity makes this process difficult. This leads us to employ algorithms that are designed to handle ambiguity of various kinds

1/23/07

CSCI 5832 Spring 2007

9

Algorithms

- In particular..
 - State-space search
 - To manage the problem of making choices during processing when we lack the information needed to make the right choice
 - Dynamic programming
 - To avoid having to redo work during the course of a state-space search
 - CKY, Earley, Minimum Edit Distance, Viterbi, Baum-Welch

1/23/07

CSCI 5832 Spring 2007

10

State Space Search

- States represent pairings of partially processed inputs with partially constructed representations.
- Goals are inputs paired with completed representations that satisfy some criteria.
- As with most interesting problems the spaces are normally too large to exhaustively explore.
 - We need heuristics to guide the search
 - Criteria to trim the space

Dynamic Programming

- Don't do the same work over and over.
- Avoid this by building and making use of solutions to sub-problems that must be invariant across all parts of the space.

Administrative Stuff

- Mailing list
 - If you're registered you're on it with your CU account.

Administrative Stuff

- The book...
- Just return the old books. We'll try to go all with the new draft chapters.

First Assignment

- Two parts
 1. Answer the following question:
 - *How many words do you know?*
 2. Write a python program that takes a newspaper article (plain text that I provide) and returns the number of:
 - *Words*
 - *Sentences*
 - *Paragraphs*

1/23/07

CSCI 5832 Spring 2007

15

First Assignment Details

- For the first part I want...
 - Your answer (ie. An actual number) and a short explanation of how you arrived at the answer
- For the second part, just bring a hardcopy to class and email me your answers to the test text that I will send out shortly before the HW is due.

1/23/07

CSCI 5832 Spring 2007


16

First Assignment

- In doing this assignment you should think ahead... *having access* to the words, sentences and paragraphs will be useful in future assignments.

Talk Tomorrow

Bill Dolan

- Manager of NLP Research at 
- “*Paraphrase as an Emergent Property of the Web*”
- *Noon, Muenzinger D430*
 - Light refreshments will be served
 - Get there early and you can pile up enough cheese, crackers and veggies to make a lunch.

Getting Going

- The next two or three lectures will cover
 - Finite state automata
 - Finite state transducers
 - English morphology

1/23/07

CSCI 5832 Spring 2007

19

Regular Expressions and Text Searching

- Everybody does it
 - Emacs, vi, perl, grep, etc..
- Regular expressions are a compact textual representation of a set of strings representing a language.

1/23/07

CSCI 5832 Spring 2007

20

Example

- Find me all instances of the word “the” in a text.
 - /the/
 - /[tT]he/
 - /\b[tT]he\b/

1/23/07

CSCI 5832 Spring 2007

21

Errors

- The process we just went through was based on two fixing kinds of errors
 - Matching strings that we should not have matched (there, then, other)
 - False positives (Type I)
 - Not matching things that we should have matched (The)
 - False negatives (Type II)

1/23/07

CSCI 5832 Spring 2007

22

Errors

- We'll be telling the same story for many tasks, all semester. Reducing the error rate for an application often involves two antagonistic efforts:
 - Increasing accuracy, or precision, (minimizing false positives)
 - Increasing coverage, or recall, (minimizing false negatives).

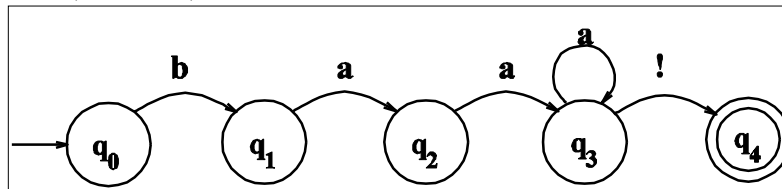
Finite State Automata

- Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata.
- FSAs and their probabilistic relatives are at the core of what we'll be doing all semester.
- They also conveniently (?) correspond to exactly what linguists say we need for morphology and parts of syntax.
 - *Coincidence?*

FSAs as Graphs

- Let's start with the sheep language from the text

- /baa+!/



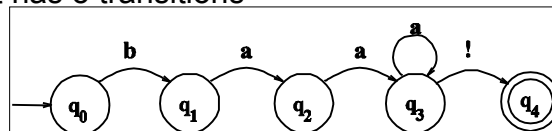
1/23/07

CSCI 5832 Spring 2007

25

Sheep FSA

- We can say the following things about this machine
 - It has 5 states
 - b, a, and ! are in its alphabet
 - q0 is the start state
 - q4 is an accept state
 - It has 5 transitions



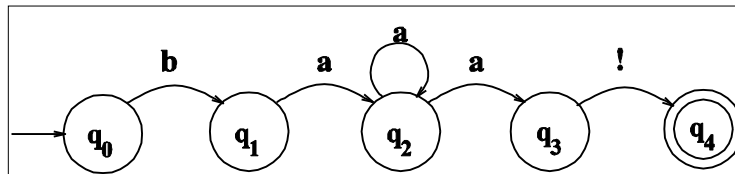
1/23/07

CSCI 5832 Spring 2007

26

But note

- There are other machines that correspond to this language



- More on this one later

1/23/07

CSCI 5832 Spring 2007

27

More Formally

- You can specify an FSA by enumerating the following things.
 - The set of states: Q
 - A finite alphabet: Σ
 - A start state
 - A set of accept/final states
 - A transition function that maps $Q \times \Sigma$ to Q

1/23/07

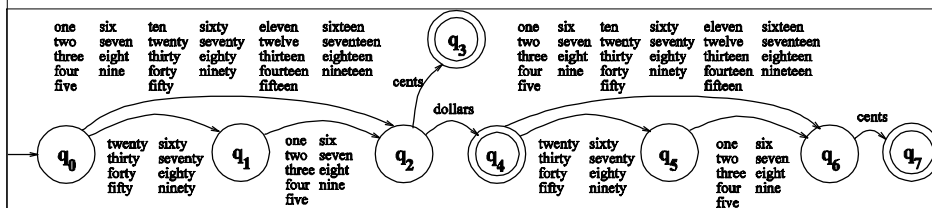
CSCI 5832 Spring 2007

28

About Alphabets

- Don't take that word too narrowly; it just means we need a finite set of symbols in the input.
- These symbols can and will stand for bigger objects that can have internal structure.

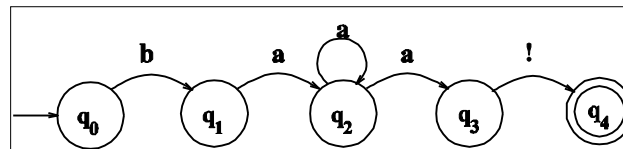
Dollars and Cents



Yet Another View

- The guts of FSAs are ultimately represented as tables

| State | Input | | | |
|-------|-------------|-------------|-------------|-------------|
| | b | a | ! | ϵ |
| 0 | 1 | \emptyset | \emptyset | \emptyset |
| 1 | \emptyset | 2 | \emptyset | \emptyset |
| 2 | \emptyset | 2,3 | \emptyset | \emptyset |
| 3 | \emptyset | \emptyset | 4 | \emptyset |
| 4: | \emptyset | \emptyset | \emptyset | \emptyset |



1/23/07

CSCI 5832 Spring 2007

31

Recognition

- Recognition is the process of determining if a string should be accepted by a machine
- Or... it's the process of determining if a string is in the language we're defining with the machine
- Or... it's the process of determining if a regular expression matches a string
- Those all amount the same thing in the end

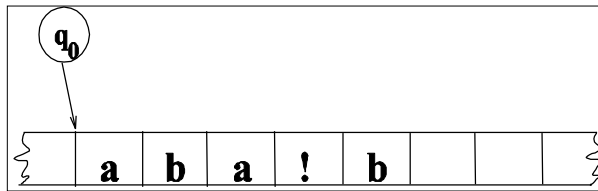
1/23/07

CSCI 5832 Spring 2007

32

Recognition

- Traditionally, (Turing's idea) this process is depicted with a tape.



1/23/07

CSCI 5832 Spring 2007

33

Recognition

- Simply a process of starting in the start state
- Examining the current input
- Consulting the table
- Going to a new state and updating the tape pointer.
- Until you run out of tape.

1/23/07

CSCI 5832 Spring 2007

34

D-Recognize

```
function D-RECOGNIZE(tape, machine) returns accept or reject
  index ← Beginning of tape
  current-state ← 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 ← transition-table[current-state, tape[index]]
      index ← index + 1
  end
```

1/23/07

CSCI 5832 Spring 2007

35

Key Points

- Deterministic means that at each point in processing there is always one unique thing to do (no choices).
- D-recognize is a simple table-driven interpreter
- The algorithm is universal for all unambiguous regular languages.
 - To change the machine, you just change the table.

1/23/07

CSCI 5832 Spring 2007

36

Key Points

- Crudely therefore... matching strings with regular expressions (ala Perl) is a matter of
 - translating the expression into a machine (table) and
 - passing the table to an interpreter

1/23/07

CSCI 5832 Spring 2007

37

Recognition as Search

- You can view this algorithm as a degenerate kind of state-space search.
- States are pairings of tape positions and state numbers.
- Operators are compiled into the table
- Goal state is a pairing with the end of tape position and a final accept state
- Its degenerate because?

1/23/07

CSCI 5832 Spring 2007

38

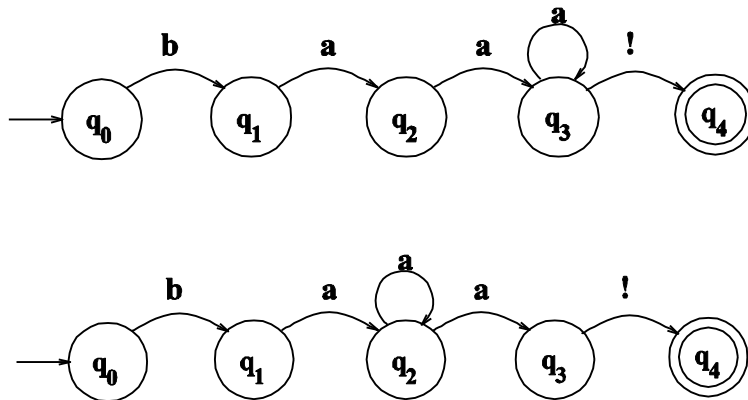
Generative Formalisms

- Formal Languages are sets of strings composed of symbols from a finite set of symbols.
- Finite-state automata define formal languages (without having to enumerate all the strings in the language)
- The term Generative is based on the view that you can run the machine as a generator to get strings from the language.

Generative Formalisms

- FSAs can be viewed from two perspectives:
 - Acceptors that can tell you if a string is in the language
 - Generators to produce all and only the strings in the language

Non-Determinism



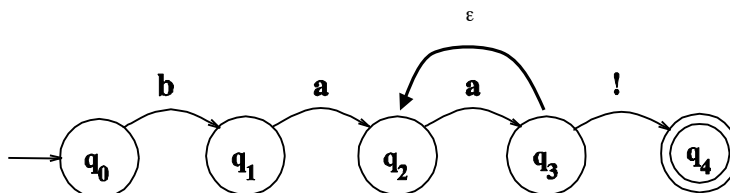
1/23/07

CSCI 5832 Spring 2007

41

Non-Determinism cont.

- Yet another technique
 - Epsilon transitions
 - Key point: these transitions do not examine or advance the tape during recognition



1/23/07

CSCI 5832 Spring 2007

42

Equivalence

- 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

1/23/07

CSCI 5832 Spring 2007

43

ND Recognition

- Two basic approaches (used in all major implementations of Regular Expressions)
 1. Either take a ND machine and convert it to a D machine and then do recognition with that.
 2. Or explicitly manage the process of recognition as a state-space search (leaving the machine as is).

1/23/07

CSCI 5832 Spring 2007

44

Implementations

| Program | (Original) Author | Version | Regex Engine |
|------------------|----------------------------|----------------|---------------------------------|
| <i>awk</i> | Aho, Weinberger, Kernighan | <i>generic</i> | DFA |
| <i>new awk</i> | Brian Kernighan | <i>generic</i> | DFA |
| GNU <i>awk</i> | Arnold Robbins | <i>recent</i> | Mostly DFA, some NFA |
| MKS <i>awk</i> | Mortice Kern Systems | | POSIX NFA |
| <i>mawk</i> | Mike Brennan | <i>all</i> | POSIX NFA |
| <i>egrep</i> | Alfred Aho | <i>generic</i> | DFA |
| MKS <i>egrep</i> | Mortice Kern Systems | | POSIX NFA |
| GNU Emacs | Richard Stallman | <i>all</i> | Trad. NFA (POSIX NFA available) |
| Expect | Don Libes | <i>all</i> | Traditional NFA |
| <i>expr</i> | Dick Haight | <i>generic</i> | Traditional NFA |
| <i>grep</i> | Ken Thompson | <i>generic</i> | Traditional NFA |
| GNU <i>grep</i> | Mike Haertel | Version 2.0 | Mostly DFA, but some NFA |
| GNU <i>find</i> | GNU | | Traditional NFA |
| <i>lex</i> | Mike Lesk | <i>generic</i> | DFA |
| <i>flex</i> | Vern Paxson | <i>all</i> | DFA |
| <i>lex</i> | Mortice Kern Systems | | POSIX NFA |
| <i>move</i> | Eric Schienbrood | <i>generic</i> | Traditional NFA |
| <i>less</i> | Mark Nudelman | | Variable (usually Trad. NFA) |
| Perl | Larry Wall | <i>all</i> | Traditional NFA |
| Python | Guido van Rossum | <i>all</i> | Traditional NFA |
| <i>sed</i> | Lee McMahon | <i>generic</i> | Traditional NFA |
| Tcl | John Ousterhout | <i>all</i> | Traditional NFA |
| <i>vi</i> | Bill Joy | <i>generic</i> | Traditional NFA |

1/23/07

CSCI 5832 Spring 2007

45

Non-Deterministic Recognition: Search

- In a ND FSA there exists at least one path through the machine for a string that is in the language defined by the machine.
- But not all paths directed through the machine for an accept string lead to an accept state.
- No paths through the machine lead to an accept state for a string not in the language.

1/23/07

CSCI 5832 Spring 2007

46

Non-Deterministic Recognition

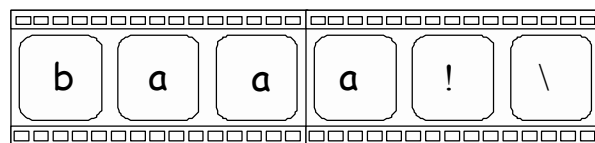
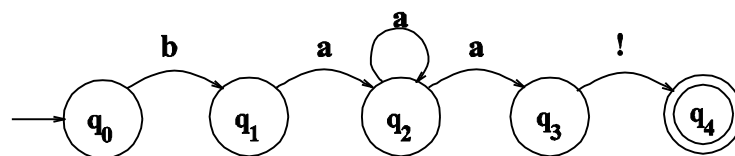
- So success in a non-deterministic recognition occurs when a path is found through the machine that ends in an accept.
- Failure occurs when all of the possible paths lead to failure.

1/23/07

CSCI 5832 Spring 2007

47

Example



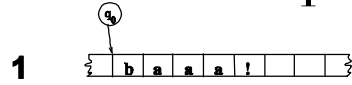
q₀ q₁ q₂ q₂ q₃ q₄

1/23/07

CSCI 5832 Spring 2007

48

Example

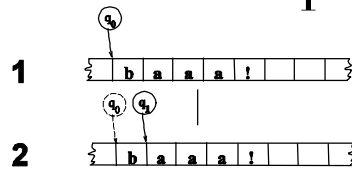


1/23/07

CSCI 5832 Spring 2007

49

Example

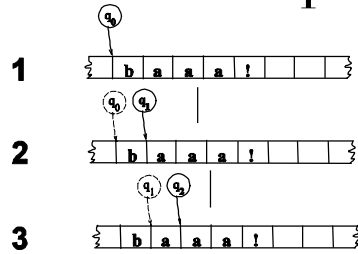


1/23/07

CSCI 5832 Spring 2007

50

Example

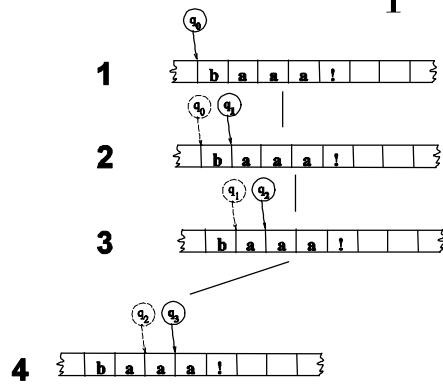


1/23/07

CSCI 5832 Spring 2007

51

Example

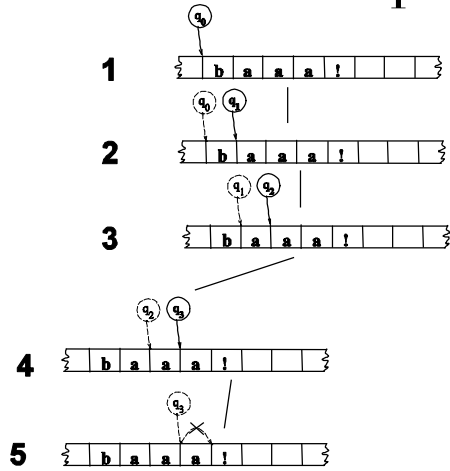


1/23/07

CSCI 5832 Spring 2007

52

Example

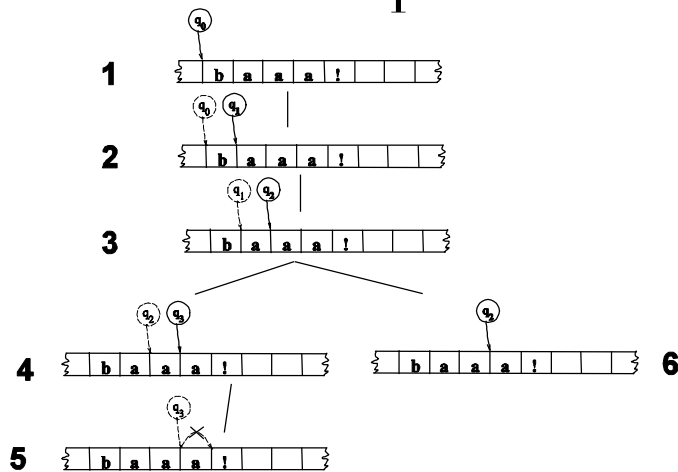


1/23/07

CSCI 5832 Spring 2007

53

Example

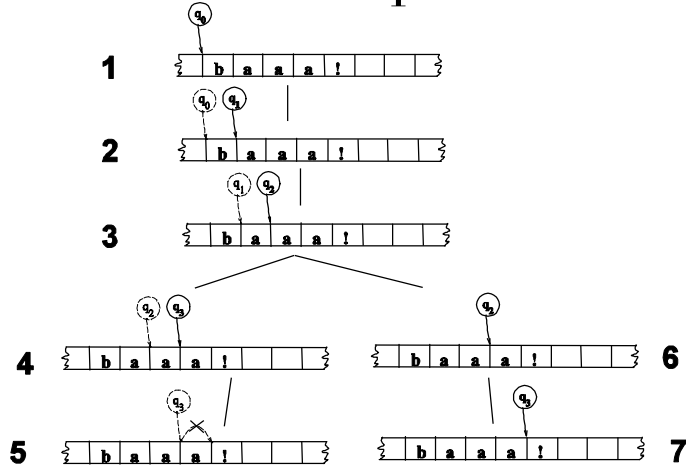


1/23/07

CSCI 5832 Spring 2007

54

Example

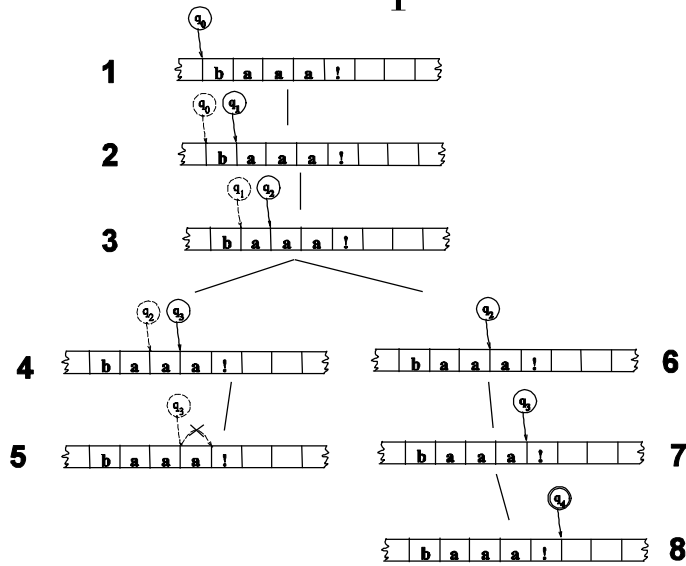


1/23/07

CSCI 5832 Spring 2007

55

Example



1/23/07

CSCI 5832 Spring 2007

56

Key Points

- States in the search space are pairings of tape positions and states in the machine.
- By keeping track of as yet unexplored states, a recognizer can systematically explore all the paths through the machine given an input.

Next Time

- Finish reading Chapter 2, start on 3.
 - *Make sure you have the book*
- Make sure you have access to Python