Natuurlijke Taalverwerking 2

www.let.rug.nl/~gosse

Gosse Bouma

2e semester 2006/2007

Overzicht

- Week 1: Inleiding Finite State Recognizers
- Week 2: Inleiding Question Answering
- Week 3: Finite State Recognizers & Transducers
- Week 4: Multilingual Question Answering
- Week 5: Fonologische regels, Replace-operator
- Week 6: Multilingual Question Answering
- Week 7: ...

Example 1: Dutch Dimunitives

huis (house)	huisje (small house)
haan (rooster)	haantje
man (man)	mannetje
raam (window)	raampje
lam (lamb)	lammetje

ring (ring)	ringetje
leerling (pupil)	leerlingetje
koning (king)	koninkje
bloem (flower)	bloempje
bloem	bloemetje

What determines DIM-realization?

- Trommelen 83 : Only phonology of the rhyme (nucleus and coda) of the last syllable,
- Competing analyses: Also refer to Stress and Morphological Structure

Induction of Linguistic Knowledge

- Automatically learn rules for DIM-realization from dictionary data (Daelemans et al 95)
- Best results look at stress and phonological properties of last three syllables,
- Especially for suffix *-etje*
  - ⋆ leerlingetje vs. koninkje

Example 2: Recognizing Unknown Words

- Dutch Dictionary Size
  - ⋆ 125K (*Groene Boekje*)
  - ⋆ 500K+ (*van Dale*).
- Tokens: the number of words,
- Types: the number of different words
- In a given text, up to 40% of the types may not occur in a dictionary.

Applying Linguistic Knowledge

- Given accurate rules for Dimunitive-formation,
- Can we implement a system that produces
  - ⋆ a dimunitive given a noun,
  - ⋆ the noun (root) given a dimunitive?
- Can we do this efficiently?
- Can we combine this with other rules (plural formation)?

- What about tokens?

Token Statistics

- Build a dictionary by collecting the most frequent words from a large text collection (Ordelman et al, 2001)
- OOV = *out of vocabulary rate*, number of word tokens missing in the dictionary

Words	Corpus	OOV
20K	110M	6.6%
40K	145M	4.5%
60K	125M	3.6%

Handling Unknown Words

- A suffix may indicate a Part of Speech category (biggest, loneliness),
- An unknown word can be a compound of two known words,
- Grapheme to Phoneme rules determine pronunciation.
- Such rules are often implemented using finite state technology

Implications

- Unknown words will occur (sooner or later),
- Given an unknown word,
- Can we say anything about their word class (noun, verb, proper name?), structure (compound? derivation?), pronunciation, etc.?
- Motivates use of Linguistic Knowledge

Part of Speech Tagging

- Subsequent decisions by the COP require the U.S. to submit these reports on an annual basis
- TV station reports that George Bush has been elected President, weeks before the election

Part of Speech Tagging

```
AT1   a
JJ    relative
NN1c  handful
IO    of
DAz   such
NN2   reports
VBDZ  was
VVNv  received
```

Part of Speech Tagging

- A POS tagger automatically assigns a category to each word in a text,
  - ⋆ For annotating corpora,
  - ⋆ As a first step towards full parsing
- Requires statistics for ⟨Word,Tag⟩ pairs and their frequency,
- But statistics for all words in the language can never be collected.

Heuristics for Unknown Words

- Usually Proper Name, Noun, Verb, or Adjective
- Sometimes the form of a word is an indication of category,
- If ending in -ative, -able, -al, -less, *etc.*, category is A.
- Heuristics, so exceptions exist.

Evaluation

- Recall : How many of the Nouns (Verbs, Adjectives) in a sample are correctly identified as such by your system?
- Precision: How many of the Nouns (Verbs, Adjectives) in a sample identified as such by your system actually are Nouns (Verbs, Adjectives)?
- Ultimately: Does including these heuristics

improve POS Tagger accuracy?

Motivation

- Efficiency & Compactness,
- Finite State Calculus:
  - ⋆ Complex Automata can be defined as combinations of smaller Automata,
  - ⋆ Regular Expressions support the definition of automata

Finite State Techniques

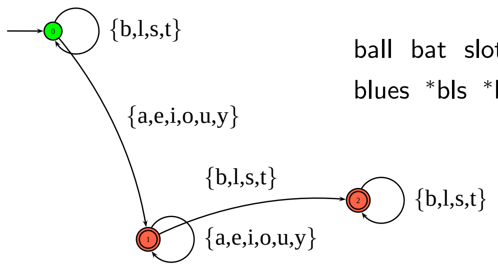
- Motivation,
- Definitions,
- Regular Expressions,
- Examples,
- Outlook.

Finite State Recognizer

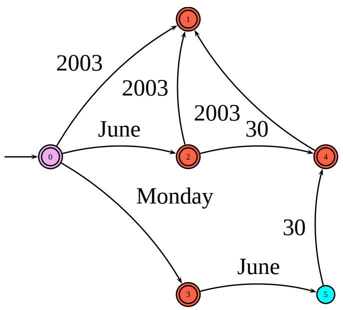
- A Finite State Recognizer consists of
  - ⋆ a number of states,
  - ⋆ start state,
  - ⋆ final states,
  - ⋆ transitions $\langle q, s, q' \rangle$: in state q, read a symbol s, go to state q'

Date Example

Syllable Example

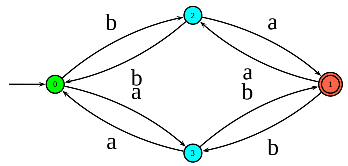


ball bat slot blue
blues *bls *bubble



Monday June 30 2003
Monday June 30
Monday
June 30 2003
June 30
June 2003
*Monday 2003
*Monday 30
*30 2003

Example



ab	aaab	abaa	*bb
ba	aaba	baaa	*aabb

Formal Definition

A finite state recognizer $M = (Q, \Sigma, T, S, F)$:

- Q is a finite set of states
- Σ is a set of symbols
- $S \subseteq Q$ is a set of start states
- $F \subseteq Q$ is a set of final states

- T is a finite set of transitions $Q \times \Sigma \times Q$.

Paths

1. for all $\langle q_0, s, q_1 \rangle \in E$: $\langle q_0, s, q_1 \rangle \in \widehat{T}$
2. if $\langle q_0, s_1, q_1 \rangle$ and $\langle q_1, s_2, q_2 \rangle$ are both in $\widehat{T}$ then $\langle q_0, s_1 s_2, q_2 \rangle \in \widehat{T}$

Language

- The language accepted by recognizer M:

  $$L(M) = \{s | q_s \in S, q_f \in F, \langle q_s, s, q_f \rangle \in \widehat{T}\}$$

- A language L is *regular* (finite state) iff there is a finite state recognizer M such that $L = L(M)$.

Regular Expressions

- Defining Automata directly is cumbersome,
- Regular expressions define finite state automata.

More Regular Expressions

`~ A`	A string not matching A
`A - B`	A string matching A but not B
`A & B`	A string matching A and B

Regular Expressions

`[A,B]`	A followed by B
`{A,B}`	A or B
`[A,B^]`	An A optionally followed by a B
`A*`	zero or more occurrences of A
`A+`	one or more occurrences of A
`?`	Any symbol
`'0'..'9'`	Symbol in the range of '0' .. '9'
`$ A`	A string containing A

Examples

`[? *, i,s,h]`	A string with suffix **ish**
`$ [q,u]`	A string containing **qu**
`'0'..'9'`	a digit
`'0'..'9' - '2'`	all digits except 2
`~ '0' ...'9'`	not a digit (i.e. includes a, 10, ϵ)
`$ a & $ b`	strings containing an a and a b

Examples

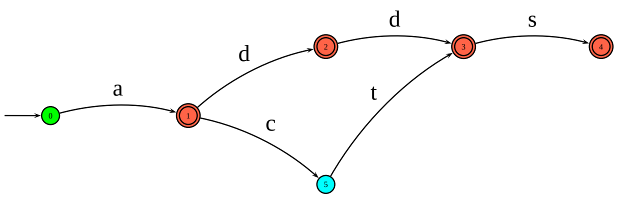
- Word Lists as Automata,
- Recognize Proper Names,
- Predict category of unknown words,
- Recognize monosyllabic words,

Word Lists as FS Automata

- Lexical look-up is fast:
  - ⋆ Independent of Dictionary (Automaton) Size
  - ⋆ Linear in length of the word
- Combine Word Lists with other Automata,
- Automaton is compact (Daciuk & van Noord, 2003).

Word Lists

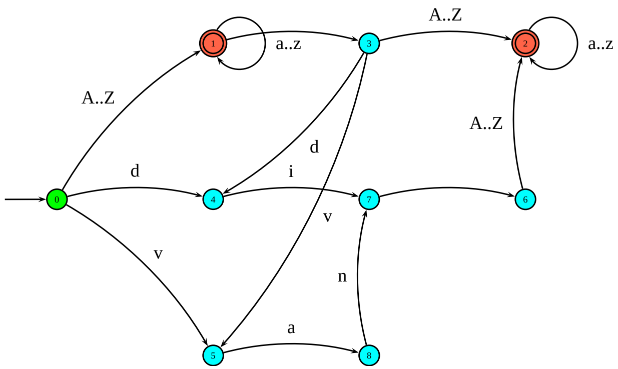
- a act acts ad add adds
- `{a,[a,c,t],[a,c,t,s],[a,d],[a,d,d],[a,d,d,s]}`



Recognizing Proper Names

- Alfredo, Jan, John
- Brown, Smith,
- di Stefano, van Dijk,
- John Brown, Alfredo di Stefano
-
```
[['A'..'Z',a..z*,' ']^,
 [{[v,a,n],[d,i]},' ']^,
 'A'..'Z',a..z*]
```

Recognizing Proper Names



Recognizing Unknown Words

- Most unknown words are nouns,
- But words ending in -able, -ive, -ish are usually adjectives,
- `[? *, {[a,b,l,e],[i,v,e],[i,s,h]}]`

Syllables

- `[{s,l,b,t}*,{a,e,o,i,u,y}+, {s,l,b,t}*]`
- blues balls *blls *bubble
- bbull lbues basll
- `[{[b,l^],l,[s,{l,t}^],t}, {[a,{i,u,y}^],[e,{a,e}^], ...}, {b,[l,{l,t}^],s,[t,t^]},s^]`

Outlook

- Finite State Transducers
  - ⋆ Map a string onto another string
  - ⋆ Morphological Analysis, Grapheme to Phoneme, etc.
- Phonological Rules
  - ⋆ Rewrite rules as FS Transducers,
  - ⋆ Regular Expression Operators

Outlook

- Applications and Learning
  - Robust Applications must deal with rules and exceptions,
  - Learning rules from data
- Applications to IE
  - Information Extraction from Text,
  - Using Regular Expressions