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

<table>
<thead>
<tr>
<th>Dutch Dimunitsive</th>
<th>Realization</th>
</tr>
</thead>
<tbody>
<tr>
<td>huis (house)</td>
<td>huisje (small house)</td>
</tr>
<tr>
<td>haan (rooster)</td>
<td>haantje</td>
</tr>
<tr>
<td>man (man)</td>
<td>mannetje</td>
</tr>
<tr>
<td>raam (window)</td>
<td>raampje</td>
</tr>
<tr>
<td>lam (lamb)</td>
<td>lammetje</td>
</tr>
</tbody>
</table>

| 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
  - leerling etje vs. koninkje

**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)?

**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.

- 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

<table>
<thead>
<tr>
<th>Words</th>
<th>Corpus</th>
<th>OOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>20K</td>
<td>110M</td>
<td>6.6%</td>
</tr>
<tr>
<td>40K</td>
<td>145M</td>
<td>4.5%</td>
</tr>
<tr>
<td>60K</td>
<td>125M</td>
<td>3.6%</td>
</tr>
</tbody>
</table>

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

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

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

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.

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.

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
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

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

Syllable Example

ball bat slot blue blues *bls *bubble

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
- $\Sigma$ 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 $\langle q_0, s, q_1 \rangle \in E$: $\langle q_0, s, q_1 \rangle \in \hat{T}$
2. If $\langle q_0, s_1, q_1 \rangle$ and $\langle q_1, s_2, q_2 \rangle$ are both in $\hat{T}$ then $\langle q_0, s_1 s_2, q_2 \rangle \in \hat{T}$

Language

- The language accepted by recognizer $M$:
  $L(M) = \{s | q_s \in S, q_f \in F, \langle q_s, s, q_f \rangle \in \hat{T}\}$
- 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.

Regular Expressions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A,B]</td>
<td>A followed by B</td>
</tr>
<tr>
<td>{A,B}</td>
<td>A or B</td>
</tr>
<tr>
<td>[A,B^]</td>
<td>An A optionally followed by a B</td>
</tr>
<tr>
<td>A*</td>
<td>zero or more occurrences of A</td>
</tr>
<tr>
<td>A+</td>
<td>one or more occurrences of A</td>
</tr>
<tr>
<td>?</td>
<td>Any symbol</td>
</tr>
<tr>
<td>'0'..'9'</td>
<td>Symbol in the range of '0' .. '9'</td>
</tr>
<tr>
<td>$ A</td>
<td>A string containing A</td>
</tr>
<tr>
<td>$ [q,u]</td>
<td>A string containing qu</td>
</tr>
<tr>
<td>'0'..'9'</td>
<td>A digit</td>
</tr>
<tr>
<td>'0'..'9' - '2'</td>
<td>all digits except 2</td>
</tr>
<tr>
<td>^ '0' .. '9'</td>
<td>not a digit (i.e. includes a, 10, ε)</td>
</tr>
<tr>
<td>a &amp; b</td>
<td>strings containing an a and a b</td>
</tr>
</tbody>
</table>

More Regular Expressions

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>~ A</td>
<td>A string not matching A</td>
</tr>
<tr>
<td>A - B</td>
<td>A string matching A but not B</td>
</tr>
<tr>
<td>A &amp; B</td>
<td>A string matching A and B</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[? *, i,s,h]</td>
<td>A string with suffix ish</td>
</tr>
<tr>
<td>$ [q,u]</td>
<td>A string containing qu</td>
</tr>
<tr>
<td>'0'..'9'</td>
<td>A digit</td>
</tr>
<tr>
<td>'0'..'9' - '2'</td>
<td>all digits except 2</td>
</tr>
<tr>
<td>^ '0' .. '9'</td>
<td>not a digit (i.e. includes a, 10, ε)</td>
</tr>
<tr>
<td>a &amp; b</td>
<td>strings containing an a and a b</td>
</tr>
</tbody>
</table>
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

Syllables

- \([s,l,b,t]^*,\{a,e,o,i,u,y\}^+,\{s,l,b,t\}^*\]
- blues balls "blls "bubble
- bbull lbusl busl
- \([[b,l^-],l,\{s,\{l,t\}^\},t],\{a,\{i,u,y\}^\},\{e,\{a,e\}^\},\ldots\},\{b,\{l,\{l,t\}^\},s,\{t,t^\}\},s^\]]

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\}]])

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