Computational Phonology

Syntagmatic computing

2019-07-16, 14:30-16:30

Dafydd Gibbon

Bielefeld University
Jinan University, Guangzhou
Objectives

- To claim that markedness, defaults and optimality are related, in the form of
  - ‘logical preferences’: ranking, elsewhere conditions, exceptions
  - ‘empirical preferences’: frequency, familiarity, statistics
- To demonstrate computation of the three structural dimensions of the architecture of language and speech:
  - Composition: ranked, grouped, parallel syntagmatic relations
  - Classification: paradigmatic relations
  - Interpretation: modelling relations
- To show that computation is essential for
  - Phonological theory
  - Phonological hypothesis testing
Types of Computing in Phonology

• Syntagmatic computing (composition)
  – Well-formedness of category combinations
    • Serial: strings, hierarchical grouping
    • Parallel: distinctive features, autosegmental tiers

• Paradigmatic computing (classification)
  – Sets: classification, categorisation
  – Properties: criteria for identifying sets

• Interpretative computing (phonetic modelling)
  • Categorial \(\leftrightarrow\) physical representation levels
  • Mapping:
    – Derivation (Generative Phonology)
    – Transduction (Finite State Phonology)
    – Selection (Optimality Theory)
Domains of Computational Phonology

● Syntagmatic (compositional) relations:
  - Autosegmental phonology
  - Metrical phonology
  - Finite state phonology

● Paradigmatic (classificatory) relations:
  - Feature theories, feature geometry
  - Inheritance phonology

● Interpretative (mapping) relations:
  - Generative phonologies
  - Optimality theoretic phonologies
Syntagmatic Computing
Syntagmatic computing (compositionality of categories)

- **Simultaneous**
  - Feature bundles
  - Feature geometry
  - Three-dimensional phonology

- **Sequential**
  - Stress cycle
  - Metrical Phonology

- **Sequential-simultaneous**
  - Autosegmental Phonology
  - Finite State Phonology
  - Inheritance Phonology

FSA, FST
Inc. formal defs.
This is the dog that chased the cat that ate the mouse ...

Right-branching linear recursion / iteration.

If the man who John met goes home then Jane will smile

Centre-embedding hierarchical recursion.

June, Jane and Jean love Mick, Dick and Nick, respectively

Recursive cross-serial dependency.

Regular languages

Chomsky Type 3,
Regular grammar
↔
Finite State Automaton

Context-free languages

Chomsky Type 2,
Context-free grammar
↔
Push-Down Automaton

Context-sensitive languages

Chomsky Type 1,
Context-sensitive grammar
↔
Linear Bounded Automaton
Chomsky maintained in *Syntactic Structures* (1957) that

*English is not a finite state language.*

- This means that there are structures which are more complex than regular languages.
- But it turns out that these more complex structures are hardly ever found in everyday spontaneous dialogue, and are restricted to formal, rehearsed speech and writing, including mathematics.
- Very many parts of language are indeed ‘finite state’, including phonology and prosody, morphology, most parts of sentence, text and discourse structures.
Syntagmatic Complexity

(3)

This is the dog that chased the cat that ate the mouse ...
Right-branching linear recursion / iteration.

Regular languages
Chomsky Type 3,
Regular grammar
⇒
Finite State Automaton

• Very many parts of language are indeed ‘finite state’, including phonology and prosody, morphology, most parts of sentence, text and discourse structures.
• Why is this so?
  • The set of syllables in any language is finite and can be described with a non-iterative finite state automaton or non-recursive regular grammar.
  • The set of words in any language is not finite, but can be described by an iterative finite state automaton or a right-recursive (or left-recursive) regular grammar.
Syntagmatic Complexity

(3)

This is the dog that chased the cat that ate the mouse ...

*Right-branching linear recursion / iteration.*

---

**Regular languages**

Chomsky Type 3, Regular grammar

↔

Finite State Automaton

---

- Very many parts of language are indeed ‘finite state’, including phonology and prosody, morphology, most parts of sentence, text and discourse structures.
- Why is this so?
  - A finite state automaton or regular grammar only requires finite memory.
    - All the other more complex kinds of grammar require, in principle, non-finite memory.
  - It is plausible that real-time speech uses finite memory.
    - It is implausible that real-time speech uses non-finite memory.
  - It is plausible that memory can be expanded by rehearsal and by the use of writing, which employs external storage.
Syntagmatic Complexity

This is the dog that chased the cat that ate the mouse ...

*Right-branching linear recursion / iteration.*

**Regular languages**

- Chomsky Type 3,
- Regular grammar
  \[\leftrightarrow\]
- Finite State Automaton

- Very many parts of language are indeed ‘finite state’, including phonology and prosody, morphology, most parts of sentence, text and discourse structures.
- Why is this so?
  - The set of syllables in any language is finite and can be described with a non-iterative finite state automaton or non-recursive regular grammar.
  - The set of words in any language is not finite, but can be described by an iterative finite state automaton or a right-recursive (or left-recursive) regular grammar.

So what are these – the finite state automaton, the regular grammar?
Syntagmatic computing: State Machines

The most basic computing mode is the State Machine
- A set of states of the system
- A set of transitions between states
- Conditions on the transitions
- A starting state
- A set of terminating states

The simplest and classic type: Finite State Automaton (FSA)
- Finite automaton (DFSA), described by a quintuple:
  \[ < Q, \Sigma, \delta, q_0, F > \]
  - \( Q \) = a finite set of states
  - \( \Sigma \) = a finite, nonempty input alphabet
  - \( \delta \) = a series of transition functions
  - \( q_0 \) = the starting state
  - \( F \) = the set of accepting (terminating states

- Deterministic: exactly one transition function for every \( \sigma \in \Sigma \) from every \( q \in Q \)
- Nondeterministic: more than transition function for any \( \sigma \in \Sigma \) from any \( q \in Q \).
Syntagmatic computing: State Machines

Finite automaton (FSA), described by a quintuple:
\[ < Q, \Sigma, \delta, q_0, F > \]
- \( Q \) = a finite set of states
- \( \Sigma \) = a finite, nonempty input alphabet
- \( \delta \) = a series of transition functions
- \( q_0 \) = the starting state
- \( F \) = the set of accepting (terminating states

- Deterministic (DFSA): exactly one transition function for every \( \sigma \in \Sigma \) from every \( q \in Q \)
- Nondeterministic (NDFSA): more than transition function for any \( \sigma \in \Sigma \) from any \( q \in Q \)

- Known principles:
  - An FSA with a transition which has the empty input symbol \( \epsilon \) is an NDFSA.
  - For any NDFSA there is a weakly equivalent DFSA.
  - For any FSA there is a weakly equivalent regular grammar in the Chomsky-Schützenberger hierarchy of formal grammars and vice versa.
There are several equivalent formalisms for FSTs

Transition diagram:

BNF notation:

Transition table:

Chomsky notation:
State Machines and Grammars

Formal grammars have the structure $<N, T, P, S>$

- $N$ is a set of non-terminal symbols
  - the non-terminal vocabulary (sometimes called variables)

- $T$ is a set of terminal symbols, $N \cap T = \emptyset$
  - the terminal vocabulary

- $S$ is a starting string in $S \in N^*$
  - for context-free and regular grammars called starting symbol

- $P$ is a set of production rules of the form $\alpha \rightarrow \beta$
  - $\alpha$ and $\beta$ are strings of symbols from $(N \cup T)^*$
  - conditions on $\alpha$ and $\beta$ are different for each type of grammar
State Machines and Grammars

Type 0: Unrestricted Grammars

- $\alpha \in (N \cup T)^* N (N\cup T)^*$
- $\beta \in (N \cup T)^*$

Type 1: Context-sensitive Grammars

$|\alpha| \leq |\beta|$, where there is no deletion

Type 2: Context-free Grammars

Phrase Structure Grammars, Constituent Structure Grammars

like Type 1, but

$\alpha \in N, |\alpha| = 1$

Type 3: Regular Grammars

like Type 2 but

1) $\beta \in T$, or

2) Either left regular or right regular, but not mixed:
   - left regular: $\beta = B\ a$, right regular: $\beta = a\ B$
   for $a \in T, B \in N$
State Machines and Grammars

Type 0: Unrestricted Grammars
- $\alpha \in (N \cup T)^*$
- $\beta \in (N \cup T)^*$

Type 1: Context-sensitive Grammars
$|\alpha| \leq |\beta|$, where there is no deletion

Type 2: Context-free Grammars
Phrase Structure Grammars, Constituent Structure Grammars
like Type 1, but
$\alpha \in N$, $|\alpha| = 1$

Type 3: Regular Grammars
like Type 2 but
1) $\beta \in T$, or
2) Either left regular or right regular, but not mixed:
   - left regular: $\beta = B a$, right regular: $\beta = a B$
   for $a \in T$, $B \in N$

The linguist’s favourite type:
In principle: Type 2
In Practice: Type 3 (right-branching (right regular))
Computational Syllable Phonotactics
Computational Syllable Phonotactics

SYLLABLE

ONSET  PEAK  CODA
Computational Syllable Phonotactics

Notice that this is right-branching.

And what about Pǔtōnghuà?
Computational Syllable Phonotactics

Notice that this is right-branching.
What do we call these finer grained end nodes?
Computational Syllable Phonotactics

SYLLABLE → ONSET NUCLEUS
NUCLEUS → PEAK CODA

Presupposed by constraints in Optimality Theory:
  ONSET
  PEAK
  NOCODA

There are many more constraints.
Computational Syllable Phonotactics

SYLLABLE → ONSET NUCLEUS
NUCLEUS → PEAK CODA

Presupposed by constraints in Optimality Theory:
ONSET
PEAK
NOCODA

Linguists love drawing trees, but tend to forget about the underlying grammars!
Computational Phonotactics: English asymmetries
Computational Phonotactics: English asymmetries

What do the constraints on English syllable patterns really look like?

Let’s take a look at a Finite State Automaton
Computational Phonotactics: English asymmetries

English onset constraints: 1 rule per context

# s + ...
q1 → s q2

# post-s AnteriorVoicelessCons
q2 → p
q2 → t
q2 → k
q2 → m
q2 → n
q2 → l
q2 → w

# post-s VoicelessStop + Gliquid
q4 → r

# post-s VoicelessStop + Gliquid
q6 → r
q6 → l

# post-s VoicelessStop + Gliquid
q7 → r
q7 → l
q7 → w

# post-s VoicessCons + j
q3 → j

# Consonant + j + u
q3 → j
Computational Phonotactics: English asymmetries

English #s__ onset constraints, Implementation as an NDFST

#  s + ...
    q1,s,q2;

#  post-s AnteriorVoicelessCons
    q2,p,q9;
    q2,t,q9;
    q2,k,q9;
    q2,m,q9;
    q2,n,q9;
    q2,l,q9;
    q2,w,q9;

#  post-s VoicelessStop + Gliquid
    q4,r,q9;
    q6,r,q9;
    q6,l,q9;
    q7,r,q9;
    q7,l,q9;
    q7,w,q9;

#  post-s VoicessCons + j
    q2,p,q3;
    q2,t,q3;
    q2,k,q3;

#  Consonant + j + u
    q3,j,q9;
Computational Phonotactics: English asymmetries

English #s__ onset constraints

http://localhost/Syllables/English/english-syllonsets-demo.html
Computational Phonotactics: English asymmetries

English syllable structure, parallel transitions reduced to one, with a single vocabulary item

http://localhost/Syllables/English/english-syllables-demo.html
Computational Phonotactics: English asymmetries

English onset structure in full detail, one transition per vocabulary item
Computational Phonotactics: English asymmetries

English syllable structure in full detail, one transition per vocabulary item

http://localhost/Syllables/English/english-syllables-demo.html
How many English syllables are there?

Two answers:
1) Lexical syllables
2) Generalised (potential) syllables

http://localhost/Syllables/English/english-syllables-demo.html
Computational Phonotactics: Pǔtōnghuà symmetries
# Computational Phonotactics: Pǔtōnghuà symmetries

<table>
<thead>
<tr>
<th>a</th>
<th>ba</th>
<th>p</th>
<th>ma</th>
<th>fa</th>
<th>da</th>
<th>t</th>
<th>na</th>
<th>la</th>
<th>ga</th>
<th>ka</th>
<th>ha</th>
<th>za</th>
<th>ca</th>
<th>sa</th>
<th>zha</th>
<th>cha</th>
<th>sh</th>
<th>sha</th>
<th>j</th>
<th>q</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>me</td>
<td>de</td>
<td>te</td>
<td>ne</td>
<td>le</td>
<td>ge</td>
<td>ke</td>
<td>he</td>
<td>ze</td>
<td>ce</td>
<td>se</td>
<td>zhe</td>
<td>che</td>
<td>she</td>
<td>che</td>
<td>she</td>
<td>re</td>
<td></td>
<td></td>
<td>a</td>
<td>o</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ai</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>en</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ng</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bu</td>
<td>bu</td>
<td>mu</td>
<td>mu</td>
<td>fu</td>
<td>du</td>
<td>tu</td>
<td>nu</td>
<td>lu</td>
<td>gu</td>
<td>ku</td>
<td>hu</td>
<td>zu</td>
<td>cu</td>
<td>su</td>
<td>zhu</td>
<td>chu</td>
<td>shu</td>
<td>ru</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>du</td>
<td>tuo</td>
<td>nuo</td>
<td>luo</td>
<td></td>
<td>guo</td>
<td>kuo</td>
<td>huo</td>
<td>zuo</td>
<td>cuo</td>
<td>suo</td>
<td>zhuo</td>
<td>chuo</td>
<td>shuo</td>
<td>ruo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>du</td>
<td>tui</td>
<td></td>
<td></td>
<td></td>
<td>gui</td>
<td>kui</td>
<td>hui</td>
<td>zui</td>
<td>cui</td>
<td>sui</td>
<td>zhiu</td>
<td>chui</td>
<td>shui</td>
<td>rui</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>du</td>
<td>an</td>
<td>tuan</td>
<td>muan</td>
<td>luan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>guan</td>
<td>kuan</td>
<td>huan</td>
<td>zuan</td>
<td>cuan</td>
<td>shuan</td>
<td>ruan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| u  | bu | pu | mu | fu | du | tu | nu | lu | gu | ku | hu | zu | cu | su | zhu | chu | shu | ru |   |   |   |
| uu |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| uo |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ai |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| au |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ui |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| uan |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ueng |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| i  | bi | pi | mi |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ia |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ie |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| iao | biao | piao | miao |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| iu |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ian | bian | pian | mian |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| iang |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| in |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ing |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| i  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ü  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| üe |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| üan |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ün |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

| j  | ji | qi | xi | yi |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| jie |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| jiao | qiao |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| iu |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ian |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| iang |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| in |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ing |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ü  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| üe |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| üan |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| ün |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Fudan University, Shanghai, July 2019  D. Gibbon: 1 Computational Phonology 35
Computational Phonotactics: Pǔtōnghuà symmetries

Model 1:

Pinyin table, grouped initials, non-deterministic

Exact model: sound and complete

http://localhost/Syllables/Mandarin/
Computational Phonotactics: Pǔtōnghuà symmetries

Model 1:

Pinyin table, grouped initials, non-deterministic

Exact model: sound and complete
Computational Phonotactics: Pǔtōnghuà symmetries

Model 2:
Pinyin table, grouped finals, deterministic
Exact model: sound and complete

http://localhost/Syllables/Mandarin/
Computational Phonotactics: Pǔtōnghuà symmetries

Model 3 (full):
Node inserted for onset glides
Complete, overgeneralises slightly

http://localhost/Syllables/Mandarin/
Computational Phonotactics: Pǔtōnghuà symmetries

Model 3 (compact):
Node inserted for onset glides
Complete, overgeneralises slightly)

http://localhost/Syllables/Mandarin/
Computational Phonotactics: Pǔtōnghuà symmetries

Model 4 (full):

Nodes inserted for onset glides and coda nasals

Complete, overgeneralises slightly, the most complex model

http://localhost/Syllables/Mandarin/
Computational Phonotactics: Pǔtōnghuà symmetries

Model 4 (compact):
Nodes inserted for onset glides and coda nasals
Complete, overgeneralises slightly, the most complex model

http://localhost/Syllables/Mandarin/
Computational Sentence Prosody - Pierrehumbert
Computational Sentence Prosody - Pierrehumbert

Intonational iteration as a layered hierarchy of loops (linear abstract oscillations)

Pierrehumbert’s regular grammar / finite state transition network

Empirical overgeneration:
1) Accents in a sequence tend to be all H* or all L*
2) Global contours tend to be rising with L* accents, falling with H* accents
3) Global contours may span more than 1 turn

Empirical undergeneration:
1) Paratone hierarchy not included
2) No time constraints

Not the first (cf. Reich, ’t Hart et al., Fujisaki, …)
But linguistically the most interesting.
Computational Sentence Prosody - Pierrehumbert

initial=q0

terminal=q4

fst=

http://localhost/Syllables/Prosody/pierrehumbert.html
Pierrehumbert’s FST with additional iteration loops for
- Intermediate Phrase
- Intonation Phrase

http://localhost/Syllables/Prosody/pierrehumbert.html
Computational lexical prosody: Niger-Congo terraced tone
Computational lexical prosody: Niger-Congo terraced tone

Niger-Congo Iterative Tonal Sandhi – a 1-tape FSA

At the most abstract level, just one node with H and L cycling around it.

From an allotonic point of view:

- 3 cycles
- 1-tape (1-level) transition network
Computational lexical prosody: Niger-Congo terraced tone

Niger-Congo Iterative Tonal Sandhi – a 2-tape FST

Syntagmatic + Interpretative Computing

From an allotonic point of view:

- 3 cycles
- 2-tape (= 2-level) transition network
Computational lexical prosody: Niger-Congo terraced tone

Niger-Congo Iterative Tonal Sandhi – a 3-tape FST

Syntagmatic + Interpretative Computing

From phonetic signal processing point of view:

- 3 cycles
- 3-tape (= 3-level) transition network
Computational lexical prosody: Niger-Congo terraced tone

Implementation as FST

# Niger-Congo terraced tone sandhi, 2 tones

initial=q0

terminal=q1,q2

fst=

q0,H,h,q1;
q0,L,l,q2;
q1,H,h^,q1;
q1,L,l^,q2;
q2,L,l!,q2;
q2,H,h!,q1;

http://localhost/Syllables/Prosody/nigercongo.html
Computational lexical prosody: Tianjin tone sandhi
Computational lexical prosody: Niger-Congo terraced tone

Tianjin Dialect Iterative Tonal Sandhi

Martin Jansche 1998
Tianjin Mandarin tone sandhi
FSTs in Automatic Speech Recognition:

Hidden Markov Models (HMMs)
FSTs in Automatic Speech Recognition: HMMs

- Training data
- Test data
- Application data
- Estimation of probabilities
- Prediction
FSTs in Automatic Speech Recognition: HMMs

- Training data
- Test data
- Application data
- Estimation of probabilities
- Prediction

Markov Model

Acoustic Vector Sequence

\[ Y = \begin{array}{c}
  y_1 \\
  y_2 \\
  y_3 \\
  y_4 \\
  y_5 
\end{array} \]
FSTs in Automatic Speech Recognition: HMMs

Application data

Prediction
FSTs in Automatic Speech Recognition: HMMs

Estimation of probabilities

Training data

Test data

Application data

Prediction

Various Acoustic Properties of Voice Signal

Input to FRONT END

Feature Vector

ACOUSTIC MODEL

Sequence of Phoneme

Collective Meaning to Phoneme

Method: VQ - Code Book or GMM

Language construct

Recognized Word

Narrow down Word using context of sentence

Feature Extraction

Method: MFCC & LPC

Recognized Phonemes
To be continued ...