Multi-Modal User Interaction
Fall 2008

Lecture 4: Language Modeling

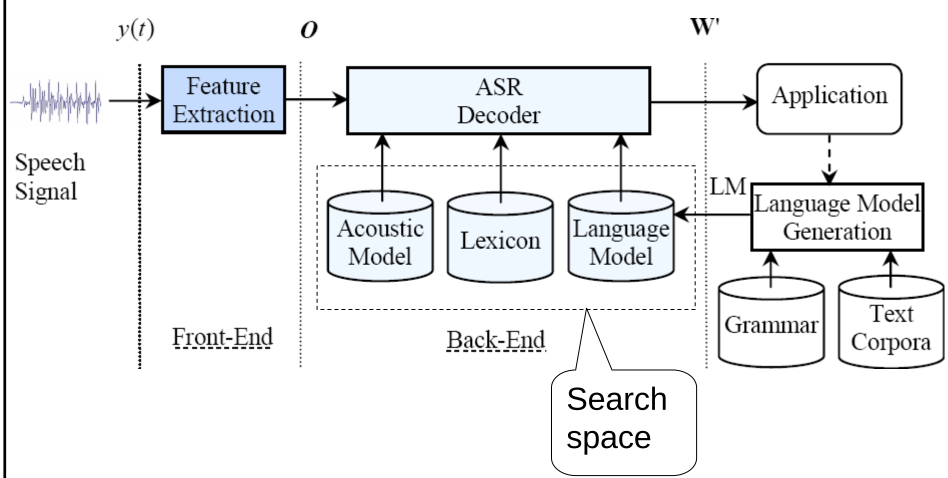
Zheng-Hua Tan

Department of Electronic Systems
Aalborg University, Denmark
zt@es.aau.dk

Part I: Introduction

- Introduction
  - Lexicon
  - Finite state grammar
  - n-gram
- Rule grammar recogniser
  - BNF format
  - Java Speech Grammar Format
  - JSGF examples for Sphinx 4

Speech recognition system



Pronunciation dictionary (lexicon)

SAMPA (Speech Assessment Methods Phonetic Alphabet) is a machine-readable phonetic alphabet.

Danish

- Aalborg Q l b Q:
- café k a f e:
- Paris p A R i: s
- tak t A g

Language modelling for speech recognition

- Speech recognizers seek the word sequence $\hat{W}$ which is most likely to be produced from acoustic evidence A

$$P(\hat{W}|A) = \max_W P(W|A) \propto \max_W P(A|W)P(W)$$

- Speech recognition involves acoustic processing, acoustic modelling, language modelling, and search
- Language models (LMs) assign a probability estimate $P(W)$ to word sequences $W = \{w_1, \ldots, w_n\}$ subject to

$$\sum_W P(W) = 1$$

- Language models help guide and constrain the search among alternative word hypotheses during recognition

(Glass, 2003)

Types of grammar

- Finite-state and phrase structure
  - take the form of rules with a left-hand and right-hand side
- n-gram
  - based on probabilities of word combinations e.g. bigrams, trigrams

Finite state grammar (networks)

- Language space defined by word network or graph

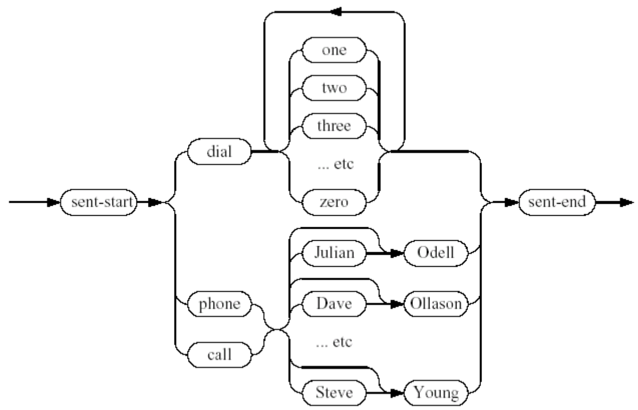


Fig. 3.1 Grammar for Voice Dialling

Word-pair grammars

show → me	me → all → the	the → flights → restaurants

- Language space defined by lists of legal word-pairs
- Can be implemented efficiently within Viterbi search
- Finite coverage can present difficulties for ASR
- Bigrams define probabilities for all word-pairs and can produce a nonzero $P(W)$ for all possible sentences

(Glass, 2003)

Language model impact

- Resource Management domain
- Speaker-independent, continuous-speech corpus
- Sentences generated from a finite state network
- 997 word vocabulary
- Word-pair perplexity ~ 60, Bigram ~ 20
- Error includes substitutions, deletions, and insertions

	No LM	Word-Pair	Bigram
% Word Error Rate	29.4	6.3	4.2

(Lee, 1988)

n-gram language models

- Probability of the sentence $S = w_1\ w_2\ ...\ w_Q$:
  $P(S) = P(w_1\ w_2\ ...\ w_Q)=$
  $P(w_1)P(w_2|w_1)P(w_3|w_1\ w_2)...P(w_Q|w_1\ w_2\ ...w_{Q-1})$
- Conditional word probability:
  $P(w_Q|w_1\ w_2\ ...\ w_{Q-1}) \approx p(w_Q|w_{Q-N+1}\ ...\ w_{Q-1})$
  where N is a constant:
  - Unigram (N=1)
  - Bigram (N=2)
  - Trigram (N=3)

n-gram language models

- n-gram models use the previous $n-1$ words to represent the history $\phi(h_i) = \{w_{i-1}, \ldots, w_{i-(n-1)}\}$
- Probabilities are based on frequencies and counts

$$\text{e.g.,} \quad f(w_3|w_1 w_2) = \frac{c(w_1 w_2 w_3)}{c(w_1 w_2)}$$

- Trigrams used for large vocabulary recognition in mid-1970's and remain the dominant language model

Google Web 1T 5-gram Corpus!
2006

Part II: Rule grammar recogniser

- Introduction
  - Lexicon
  - Finite state grammar
  - n-gram
- Rule grammar recogniser
  - BNF format
  - VoiceXML
  - Java Speech Grammar Format
  - JSGF examples for Sphinx 4

Rule grammar recogniser

- Grammars determine what the recognizer should listen for and describe the utterances a user may say
- Rule grammar recogniser, i.e. command and control recogniser
- Grammar formats
  - BNF
  - VoiceXML
  - JSGF

Grammar in the BNF format

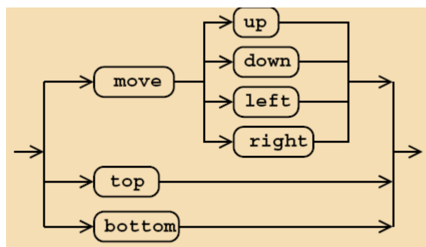
BNF (Backus-Naur Format)

- [.] optional
- {.} zero or more
- (.) block (grouping)
- <.> loop (one or more)
- .|. alternative (or)

BNF grammar – an example

```
> cat grammar.bnf
$dir   = up | down | left | right;
$mcmd = move $dir | top | bottom;
$item = char | word | line | page;
$dcmd = delete [$item];
$icmd = insert;
$ecmd = end [insert];
$cmd   = $mcmd | $dcmd | $icmd | $ecmd;
$noise      = sil | fil | spk;
({$noise} < $cmd {$noise} > quit {$noise})
```

HTK supports it.



VoiceXML

- Voice Extensible Markup Language (VoiceXML)
- The VoiceXML 2.0 specification includes a set of built-in grammars as a convenience to enable developers to get started writing more complex VoiceXML applications quickly.

VoiceXML example

Asks the user for a choice of drink and then submits it to a server script

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="http://www.w3.org/2001/vxml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.w3.org/2001/vxml
    http://www.w3.org/TR/voicexml20/vxml.xsd"
    version="2.0">
    <form>
        <field name="drink">
            <prompt>Would you like coffee, tea, milk, or nothing?</prompt>
            <grammar src="drink.grxml" type="application/srgs+xml"/>
        </field>
        <block>
            <submit next="http://www.drink.example.com/drink2.asp"/>
        </block>
    </form>
</vxml>
```


Built-in grammars in VoiceXML

- Date
  - May fifth
  - the thirty first of December two thousand
  - March
  - yesterday
  - today
  - tomorrow

Built-in grammars in VoiceXML

- Time
  - one o'clock
  - five past one
  - three fifteen
  - seven thirty
  - half past eight
  - oh four hundred hours
  - sixteen fifty
  - twelve noon
  - midnight

Java Speech Grammar Format

Java Speech Grammar Format (JSGF)

- is a platform-independent, vendor-independent textual representation of grammars for use in speech recognition.
- adopts the style and conventions of the Java programming language in addition to use of traditional grammar notations.

Grammar names and declaration

- Each grammar has a unique name that is declared in the grammar header
- Grammar's name must be declared as the first statement of that grammar:
  grammar grammarName
  - simple grammar name e.g.
    - grammar robot;
  - full grammar name (=package name + simple grammar name) e.g.
    - grammar com.acme.politeness;

Rulename

Grammar is composed of a set of rules that define what may be spoken. Rules are combinations of speakable text and references to other rules.

Each rule has a unique rulename:

- Rulename can be written in most of the world's living languages
  - Chinese, Japanese, Korean, European languages…
- Case sensitive
  - <name> and <Name> are different

Comments and grammar header

- /* text */ A traditional comment.
- // text A single-line comment.
- The header format is

#JSGF version char-encoding local;

e.g.

#JSGF V1.0;

Import

- The import declarations follow the grammar declaration and must come before the grammar body (the rule definitions).
- An import declaration allows one or all of the public rules of another grammar to be referenced locally. Formats:
  - import <*fullyQualifiedRuleName*>;
  - import <*fullGrammarName*.*>;
- For example,
  - import <com.sun.speech.app.index.1stTo31st>;

  an import of a single rule by its fully-qualified rulename: the rule <1stTo31st> from the grammar com.sun.speech.app.index. The imported rule, <1stTo31st>, must be a public rule of the imported grammar.
  - import <com.sun.speech.app.numbers.*>;

  The use of the asterisk requests import of all public rules of the numbers grammar. E.g., if that grammar defines 2 public rules, <digits>, <teens>, then both 2 may be referenced locally.

Rule definitions

- Grammar body defines rules
  - <ruleName> = ruleExpansion;
  - public <ruleName> = ruleExpansion;
- Weights
  - <size> = /10/ small | /2/ medium | /1/ large;

Grouping and unary operators

- Grouping
  - <command> = (open | close) (windows | doors);
- Unary operators
  - <polite> = please | kindly | oh mighty computer;
  - <command> = <polite> * don’t crash

A rule expansion followed by the asterisk symbol indicates that the expansion may be spoken *zero or more times*. Here a user can say things like "please don't crash", "oh mighty computer please please don't crash", or to ignore politeness with "don't crash“.

  - <command> = <polite> + don’t crash

The plus symbol indicates the expansion may be spoken one of more times.

Tags

- Tags provide a mechanism for grammar writers to attach application-specific information to parts of rule definitions.
- Applications typically use tags to simplify or enhance the processing of recognition results.
- Tag attachments do not affect the recognition of a grammar. Instead, the tags are attached to the result object returned by the recognizer to an application.
- A tag is a unary operator. The tag is a string delimited by curly braces `{}'.
- The tag attaches to the immediate preceding rule expansion. E.g.
  - <rule> = <action> {tag in here}; <command>= please (open {OPEN} | close {CLOSE}) the file;

Example 1: Hello world application

- Robot control

```
#JSGF V1.0;
/**
 * JSGF Robot Grammar for Hello World example
 */
grammar robot;

public <move> =    (LIFT ARM | STEP FORWARD | SIT
    DOWN | ENTER STAIRS | FETCH THE CUP) * ;
```

Example 2: Simple Command & Control

```
#JSGF V1.0;
grammar com.acme.politeness;
// Body
public <startPolite> = (please | kindly | could you | oh mighty computer) *;
public <endPolite> = [ please | thanks | thank you ];
```

```
#JSGF V1.0 ISO8859-1 en;
grammar com.acme.commands;
import <com.acme.politeness.startPolite>;
import <com.acme.politeness.endPolite>;
/**
* Basic command.
* @example please move the window
* @example open a file
*/
public <basicCmd> = <startPolite> <command> <endPolite>;

<command> = <action> <object>;
<action> = /10/ open |/2/ close |/1/ delete |/1/ move;
<object> = [the | a] (window | file | menu);
```

Example 3: HCWapp Application for PDA

```
#JSGF V1.0;
grammar jsgf;
<quit> = EXIT | QUIT;
<selection> = SELECT | VIEW | DISPLAY | GET | GO TO;
<tab> = TAB | PAGE | SCREEN;
<application> = PROGRAM | APPLICATION | SOFTWARE;
public <jsgf> =
  [<selection>] ([NEXT] VISITS | [THE] <tab> TWO)   {goto_visits} |
  [<selection>] (PATIENT [DETAILS] | [THE] <tab> THREE)
                                                    {goto_patients} |
  [<selection>] (OPTIONS | [THE] <tab> FOUR)   {goto_options} |
  [<selection>] (HELP | [THE] <tab> FIVE)           {goto_help} |
  (<quit> [[THE] <application>])                  {quit};
```

Summary

- Introduction
  - Lexicon
  - Finite state grammar
  - n-gram
- Rule grammar recogniser
  - BNF format
  - VoiceXML
  - Java Speech Grammar Format
  - JSGF examples for Sphinx 4