Lecture 13: Text-to-Speech I: Text Normalization, Letter to Sound, Prosody
Outline

• History, Demos
• Architectural Overview
• Stage 1: Text Analysis
  • Text Normalization
    • Tokenization
    • End of sentence detection
    • Homograph disambiguation
  • Letter-to-sound (grapheme-to-phoneme)
  • Prosody
Dave Barry on TTS

“And computers are getting smarter all the time; scientists tell us that soon they will be able to talk with us.

(By “they”, I mean computers; I doubt scientists will ever be able to talk to us.)
History of TTS

- Pictures and some text from Hartmut Traunmüller’s web site: http://www.ling.su.se/staff/hartmut/kemplne.htm
- Von Kempeln 1780 b. Bratislava 1734 d. Vienna 1804
- Leather resonator manipulated by the operator to try and copy vocal tract configuration during sonorants (vowels, glides, nasals)
- Bellows provided air stream, counterweight provided inhalation
- Vibrating reed produced periodic pressure wave
Von Kempelen:

- Small whistles controlled consonants
- Rubber mouth and nose; nose had to be covered with two fingers for non-nasals
- Unvoiced sounds: mouth covered, auxiliary bellows driven by string provides puff of air

*From Traunmüller’s web site*
Closer to a natural vocal tract:
Riesz 1937
Homer Dudley 1939 VODER

- Synthesizing speech by electrical means
- 1939 World’s Fair
Homer Dudley’s VODER

- Manually controlled through complex keyboard
- Operator training was a problem
An aside on demos

That last slide exhibited:

**Rule 1 of playing a speech synthesis demo:**
Always have a human say what the words are right before you have the system say them
The 1936 UK Speaking Clock

From http://web.ukonline.co.uk/freshwater/clocks/spkgclock.htm
The UK Speaking Clock

- July 24, 1936
- Photographic storage on 4 glass disks
- 2 disks for minutes, 1 for hour, one for seconds.
- Other words in sentence distributed across 4 disks, so all 4 used at once.
- Voice of “Miss J. Cain”
A technician adjusts the amplifiers of the first speaking clock

From http://web.ukonline.co.uk/freshwater/clocks/spkgclock.htm
Gunnar Fant’s OVE synthesizer

- Of the Royal Institute of Technology, Stockholm
- Formant Synthesizer for vowels
- F1 and F2 could be controlled

From Traunmüller’s web site
Cooper’s Pattern Playback

- Haskins Labs for investigating speech perception
- Works like an inverse of a spectrograph
- Light from a lamp goes through a rotating disk then through spectrogram into photovoltaic cells
- Thus amount of light that gets transmitted at each frequency band corresponds to amount of acoustic energy at that band
Cooper’s Pattern Playback
Pre-modern TTS systems

- 1970’s
  - Joe Olive 1977 concatenation of linear-prediction diphones
  - Texas Instruments Speak and Spell, June 1978
  - Paul Breedlove
Types of Synthesis

- **Articulatory Synthesis:**
  - Model movements of articulators and acoustics of vocal tract

- **Formant Synthesis:**
  - Start with acoustics, create rules/filters to create each formant

- **Concatenative Synthesis:**
  - Use databases of stored speech to assemble new utterances.
    - Diphone
    - Unit Selection

- **Parametric (HMM) Synthesis**
  - Trains parameters on databases of speech

Text modified from Richard Sproat slides
1980s: Formant Synthesis

- Were the most common commercial systems when computers were slow and had little memory.
- 1979 MIT MITalk (Allen, Hunnicut, Klatt)
- 1983 DECTalk system based on KlattTalk
  - “Perfect Paul” (The voice of Stephen Hawking)
  - “Beautiful Betty”
1990s: 2nd Generation Synthesis

Diphone Synthesis

- Units are diphones; middle of one phone to middle of next.
- Why? Middle of phone is steady state.
- Record 1 speaker saying each diphone
- ~1400 recordings
- Paste them together and modify prosody.
3rd Generation Synthesis Unit Selection Systems

- All current commercial systems.
- Larger units of variable length
- Record one speaker speaking 10 hours or more,
  - Have multiple copies of each unit
- Use search to find best sequence of units
Future? Parametric Synthesis

- Most academic research
- Also called HMM Synthesis
- Train a statistical model on large amounts of data.
TTS Demos:
Diphone, Unit-Selection and Parametric

Festival
http://www.cstr.ed.ac.uk/projects/festival/morevoices.html

Google:
chrome-extension/::chhkejjkcgchanjclmhhpncachhgejoel:ttsdemo.html

Cereproc
The two stages of TTS

PG&E will file schedules on April 20.

1. **Text Analysis**: Text into intermediate representation:

   | P | p | i | iy | G | jh | iy | AND | ae | n | d | E | iy | WILL | w | ih | l | FILE | f | ay | l | SCHEDULES | s | k | eh | jh | ax | l | z | ON | a | n | ey | APRIL | p | r | ih | l | TWENTIETH | t | w | eh | n | t | iy | ax | th |

2. **Waveform Synthesis**: From the intermediate representation into waveform
PG&E will file schedules on April 20.
Outline

- History of Speech Synthesis
- State of the Art Demos
- Brief Architectural Overview
- Stage 1: Text Analysis
  - Text Normalization
  - Phonetic Analysis
    - Letter-to-sound (grapheme-to-phoneme)
  - Prosodic Analysis
Text Normalization

Analysis of raw text into pronounceable words:

He said the increase in credit limits helped B.C. Hydro achieve record net income of about $1 billion during the year ending March 31. This figure does not include any write-downs that may occur if Powerex determines that any of its customer accounts are not collectible. Cousins, however, was insistent that all debts will be collected: “We continue to pursue monies owing and we expect to be paid for electricity we have sold.”

- Sentence Tokenization
- Text Normalization
  - Identify tokens in text
  - Chunk tokens into reasonably sized sections
  - Map tokens to words
  - Tag the words
Text Processing

- He stole $100 million from the bank
- It’s 13 St. Andrews St.
- The home page is [http://www.stanford.edu](http://www.stanford.edu)
- Yes, see you the following tues, that’s 11/12/01
- IV: four, fourth, I.V.
- IRA: I.R.A. or Ira
- 1750: seventeen fifty (date, address) or one thousand seven... (dollars)
Text Normalization Steps

1. Identify tokens in text
2. Chunk tokens
3. Identify types of tokens
4. Convert tokens to words
Step 1: identify tokens and chunk

- Whitespace can be viewed as separators
- Punctuation can be separated from the raw tokens
- For example, *Festival* converts text into
  - ordered list of tokens
  - each with features:
    - its own preceding whitespace
    - its own succeeding punctuation
Important issue in tokenization: end-of-utterance detection

• Relatively simple if utterance ends in ?!
• But what about ambiguity of “.”?
• Ambiguous between end-of-utterance, end-of-abbreviation, and both
  • My place on Main St. is around the corner.
  • I live at 123 Main St.
  • (Not “I live at 151 Main St..”)
Rules/features for end-of-utterance detection

- A dot with one or two letters is an abbrev.
- A dot with 3 cap letters is an abbrev.
- An abbrev followed by 2 spaces and a capital letter is an end-of-utterance.
- Non-abbrevs followed by capitalized word are breaks.
Simple Decision Tree

Lots of blank lines after me?

Final punctuation is ?!, or :?

Final punctuation is period

I am “etc” or other abbreviation

Not E-O-S

Not E-O-S

E-O-S
The Festival hand-built decision tree for end-of-utterance

((n.whitespace matches ".*\n.*\n[ \n]*") ;; A significant break in text
  ((1))
  ((punc in ("?:" "!")))
  ((1))
  ((punc is "."))
 ;; This is to distinguish abbreviations vs periods
 ;; These are heuristics
((name matches "\/(.*\..*\|[A-Z][A-Za-z]?[A-Za-z]?\|etc\)")
  ((n.whitespace is " ")
   ((0)) ;; if abbrev, single space enough for break
  ((n.name matches "[A-Z].*"))
   ((1))
   ((0))))
((n.whitespace is " ") ;; if it doesn't look like an abbreviation
((n.name matches "[A-Z].*")) ;; single sp. + non-cap is no break
  ((1))
  ((0)))
  ((1))))
  ((0))))
Problems with the previous decision tree

- Fails for
  - Cog. Sci. Newsletter
  - Lots of cases at end of line.
  - Badly spaced/capitalized sentences
More sophisticated decision tree features

- \( \text{Prob(word with "." occurs at end-of-s)} \)
- \( \text{Prob(word after "." occurs at begin-of-s)} \)
- \( \text{Length of word with "."} \)
- \( \text{Length of word after "."} \)
- \( \text{Case of word with ".": Upper, Lower, Cap, Number} \)
- \( \text{Case of word after ".": Upper, Lower, Cap, Number} \)
- \( \text{Punctuation after "." (if any)} \)
- \( \text{Abbreviation class of word with "." (month name, unit-of-measure, title, address name, etc)} \)

From Richard Sproat slides
Sproat EOS tree

From Richard Sproat slides
Some good references on end-of-sentence detection


Steps 3+4: Identify Types of Tokens, and Convert Tokens to Words

- Pronunciation of numbers often depends on type. Three ways to pronounce 1776:
  - **Date**: seventeen seventy six
  - **Phone number**: one seven seven six
  - **Quantifier**: one thousand seven hundred (and) seventy six

- Also:
  - **25 Day**: twenty-fifth
Festival rule for dealing with “$1.2 million”

(define (token_to_words utt token name)
  (cond
   ((and (string-matches name "\\[$0-9,]+\\(\\.[0-9]+\\)?")
      (string-matches (utt.streamitem.feat utt token "n.name")
       ".*illion.?"))
    (append
     (builtin_english_token_to_words utt token (string-after name "$"))
     (list
      (utt.streamitem.feat utt token "n.name"))))
   ((and (string-matches (utt.streamitem.feat utt token "p.name")
       "\\[$0-9,]+\\(\\.[0-9]+\\)?")
      (string-matches name ".*illion.?"))
    (list "dollars"))
   (t
    (builtin_english_token_to_words utt token name))))
Rules versus machine learning

- Rules/patterns
  - Simple
  - Quick
  - Can be more robust
  - Easy to debug, inspect, and pass on
  - Vastly preferred in commercial systems

- Machine Learning
  - Works for complex problems where rules hard to write
  - Higher accuracy in general
  - Worse generalization to very different test sets
  - Vastly preferred in academia
Machine learning for Text Normalization


- NSW examples:
  - Numbers:
    - 123, 12 March 1994
  - Abbreviations, contractions, acronyms:
    - approx., mph. ctrl-C, US, pp, lb
  - Punctuation conventions:
    - 3-4, +/-, and/or
  - Dates, times, urls, etc
How common are NSWs?

- Word not in lexicon, or with non-alphabetic characters (Sproat et al 2001, before SMS/Twitter)

<table>
<thead>
<tr>
<th>Text Type</th>
<th>% NSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>novels</td>
<td>1.5%</td>
</tr>
<tr>
<td>press wire</td>
<td>4.9%</td>
</tr>
<tr>
<td>e-mail</td>
<td>10.7%</td>
</tr>
<tr>
<td>recipes</td>
<td>13.7%</td>
</tr>
<tr>
<td>classified</td>
<td>17.9%</td>
</tr>
</tbody>
</table>
How common are NSWs?

Word not in gnu aspell dictionary (Han, Cook, Baldwin 2013) not counting @mentions, #hashtags, urls,

Twitter: 15% of tweets have 50% or more OOV
<table>
<thead>
<tr>
<th>Category</th>
<th>Example</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter changed (deleted)</td>
<td>shuld</td>
<td>72%</td>
</tr>
<tr>
<td>Slang</td>
<td>lol</td>
<td>12%</td>
</tr>
<tr>
<td>Other</td>
<td>sucha</td>
<td>10%</td>
</tr>
<tr>
<td>Number Substitution</td>
<td>4 (“for”)</td>
<td>3%</td>
</tr>
<tr>
<td>Letter and Number</td>
<td>b4 (“before”)</td>
<td>2%</td>
</tr>
</tbody>
</table>
State of the Art for Twitter normalization

- Simple one-to-one normalization (map OOV to a single IV word)
  - Han, Cook, and Baldwin 2013
- Create a typed-based dictionary of mappings
  - tmrw -> tomorrow
- Learned over a large corpus by combining distributional similarity and string similarity
4 steps to Sproat et al. algorithm

- Splitter (on whitespace or also within word (“AltaVista”))
- Type identifier: for each split token identify type
- Token expander: for each typed token, expand to words
  - Deterministic for number, date, money, letter sequence
  - Only hard (nondeterministic) for abbreviations
- Language Model: to select between alternative pronunciations

From Alan Black slides
Step 1: Splitter

- Letter/number conjunctions (WinNT, SunOS, PC110)
- Hand-written rules in two parts:
  - Part I: group things not to be split (numbers, etc; including commas in numbers, slashes in dates)
  - Part II: apply rules:
    - At transitions from lower to upper case
    - After penultimate upper-case char in transitions from upper to lower
    - At transitions from digits to alpha
    - At punctuation
Step 2: Classify token into 1 of 20 types

**EXPN**: abbrev, contractions (adv, N.Y., mph, gov’ t)

**LSEQ**: letter sequence (CIA, D.C., CDs)

**ASWD**: read as word, e.g. CAT, proper names

**MSPL**: misspelling

**NUM**: number (cardinal) (12, 45, 1/2, 0.6)

**NORD**: number (ordinal) e.g. May 7, 3rd, Bill Gates II

**NTEL**: telephone (or part) e.g. 212-555-4523

**NDIG**: number as digits e.g. Room 101

**NIDE**: identifier, e.g. 747, 386, I5, PC110

**NADDR**: number as stresst address, e.g. 5000 Pennsylvania

**NZIP, NTIME, NDATE, NYER, MONEY, BMONY, PRCT, URL**, etc

**SLNT**: not spoken (KENT*REALTY)
More about the types

4 categories for alphabetic sequences:

**EXPN**: expand to full word(s) (fplc= fireplace, NY=New York)

**LSEQ**: say as letter sequence (IBM)

**ASWD**: say as standard word (either OOV or acronyms)

5 main ways to read numbers:

**Cardinal** (quantities)

**Ordinal** (dates)

**String of digits** (phone numbers)

**Pair of digits** (years)

**Trailing unit**: serial until last non-zero digit: 8765000 is “eight seven six five thousand” (phone #s, long addresses)

**But still exceptions**: (947-3030, 830-7056)
Type identification classifier

Hand label a large training set, build classifier
Example features for alphabetic tokens:

\[ P(o \mid t) \] for \( t \) in ASWD, LSWQ, EXPN (from trigram letter model)

\[ p(o \mid t) = \sum_{i=1}^{N} p(l_{i1} \mid l_{i-1}, l_{i-2}) \]

\( P(t) \) from counts of each tag in text
\( P(o) \) normalization factor
\( P(t \mid o) = p(o \mid t)p(t)/p(o) \)
Type identification algorithm

- Hand-written context-dependent rules:
  - List of lexical items (Act, Advantage, amendment) after which Roman numbers are read as cardinals not ordinals

- Classifier accuracy:
  - 98.1% in news data,
![](image)
  - 91.8% in email
Step 3: expand NSW Tokens by type-specific rules

- ASWD expands to itself
- LSEQ expands to list of words, one for each letter
- NUM expands to string of words representing cardinal
- NYER expand to 2 pairs of NUM digits...
- NTEL: string of digits with silence for punctuation
In practice

- NSWs are still mainly done by rule in TTS systems
Homograph disambiguation

It’s no use (/y uw s/) to ask to use (/y uw z/) the telephone.
Do you live (/l ih v/) near a zoo with live (/l ay v/) animals?
I prefer bass (/b ae s/) fishing to playing the bass (/b ey s/) guitar.

<table>
<thead>
<tr>
<th>Final voicing</th>
<th>Stress shift</th>
<th>-ate final vowel</th>
</tr>
</thead>
<tbody>
<tr>
<td>N (/s/)</td>
<td>V (/z/)</td>
<td>N/A (final /ax/)</td>
</tr>
<tr>
<td>use</td>
<td>y uw s</td>
<td>eh s t ih m ax t</td>
</tr>
<tr>
<td>close</td>
<td>k l ow s</td>
<td>s eh p ax r ax t</td>
</tr>
<tr>
<td>house</td>
<td>h aw s</td>
<td>m aa d ax r ax t</td>
</tr>
</tbody>
</table>

Some systematic relationships between homographs:

French homographs include /s/ and /z/ as final consonants.

Some systematic relationships between homographs labeled by part of speech and then running a part-of-speech tagger to choose the pronunciation for a given homograph in context.

Similarly, in German, final voicing may differ depending on the grammatical case of the noun (Demberg, 2006).

In English (as well as in similar languages like French and German) tend to have different parts of speech. For example, the two forms of /fi/ 'thread', or the multiple pronunciations for /klo/ and /klo/ 'clothes',

The goal of our NSW algorithms in the previous section was to determine which segmentations are (respectively) a verb and a noun. Fig-8.5 shows some interesting systematic relations between the pronunciation of some

Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs. Indeed, Liberman and Church (1992) showed that many of the most frequent homographs in 44 million words of AP newswire are noun-verb and adj-verb homographs.
Homograph disambiguation

• 19 most frequent homographs, from Liberman and Church 1992
• Counts are per million, from an AP news corpus of 44 million words.
• Not a huge problem, but still important

<table>
<thead>
<tr>
<th>use</th>
<th>319</th>
<th>survey</th>
<th>91</th>
</tr>
</thead>
<tbody>
<tr>
<td>increase</td>
<td>230</td>
<td>project</td>
<td>90</td>
</tr>
<tr>
<td>close</td>
<td>215</td>
<td>separate</td>
<td>87</td>
</tr>
<tr>
<td>record</td>
<td>195</td>
<td>present</td>
<td>80</td>
</tr>
<tr>
<td>house</td>
<td>150</td>
<td>read</td>
<td>72</td>
</tr>
<tr>
<td>contract</td>
<td>143</td>
<td>subject</td>
<td>68</td>
</tr>
<tr>
<td>lead</td>
<td>131</td>
<td>rebel</td>
<td>48</td>
</tr>
<tr>
<td>live</td>
<td>130</td>
<td>finance</td>
<td>46</td>
</tr>
<tr>
<td>lives</td>
<td>105</td>
<td>estimate</td>
<td>46</td>
</tr>
</tbody>
</table>
POS Tagging for homograph disambiguation

- Many homographs can be distinguished by POS
  - use: y uw s y uw z
  - close: k l ow s k l ow z
  - house: h aw s h aw z
  - live: l ay v l ih v
  - REcord: reCORD
  - INsult: inSULT
  - O Bj ect: obJECT
  - OVERflow: overFLOW
  - DIScount: disCOUNT
  - CONtent: conTENT

- POS tagging also useful for CONTENT/FUNCTION distinction, which is useful for phrasing
Festival

- Open source speech synthesis system
- Designed for development and runtime use
  - Use in many commercial and academic systems
  - Hundreds of thousands of users
- Multilingual
  - No built-in language
  - Designed to allow addition of new languages
- Additional tools for rapid voice development
  - Statistical learning tools
  - Scripts for building models

Text from Richard Sproat
Festival as software

- [http://festvox.org/festival/](http://festvox.org/festival/)
- General system for multi-lingual TTS
- C/C++ code with Scheme scripting language
- General replaceable modules:
  - Lexicons, LTS, duration, intonation, phrasing, POS tagging, tokenizing, diphone/unit selection, signal processing
- General tools
  - Intonation analysis (f0, Tilt), signal processing, CART building, N-gram, SCFG, WFST

Text from Richard Sproat
CMU FestVox project

- Festival is an engine, how do you make voices?
- Festvox: building synthetic voices:
  - Tools, scripts, documentation
  - Discussion and examples for building voices
  - Example voice databases
  - Step by step walkthroughs of processes
- Support for English and other languages
- Support for different waveform synthesis methods
  - Diphone
  - Unit selection
Synthesis tools

- I want my computer to talk
  - Festival Speech Synthesis
- I want my computer to talk in my voice
  - FestVox Project
- I want it to be fast and efficient
  - Flite
Dictionaries

- CMU dictionary: 127K words
  - [http://www.speech.cs.cmu.edu/cgi-bin/cmudict](http://www.speech.cs.cmu.edu/cgi-bin/cmudict)
- Unisyn dictionary
  - Significantly more accurate, includes multiple dialects

```plaintext
going:   { g * ou } > i ng >
antecedents:   { * a n . t^ i . s ~ ii . d n! t } > s >
dictionary:    { d * i k . sh @ . n ~ e . r i i }
Dictionaries aren’t always sufficient: Unknown words

- Go up with square root of # of words in text
- Mostly person, company, product names
  - From a Black et al analysis
    - 5% of tokens in a sample of WSJ not in the OALD dictionary
    - 77%: Names
    - 20% Other Unknown Words
    - 4%: Typos
- So commercial systems have 3-part system:
  - Big dictionary
  - Special code for handling names
  - Machine learned LTS system for other unknown words
Names

- Big problem area is names
- Names are common
  - 20% of tokens in typical newswire text
  - Spiegel (2003) estimate of US names:
    - 2 million surnames
    - 100,000 first names
  - Personal names: McArthur, D’Angelo, Jiminez, Rajan, Raghavan, Sondhi, Xu, Hsu, Zhang, Chang, Nguyen
  - Company/Brand names: Infinit, Kmart, Cytyc, Medamicus, Inforte, Aaon, Idexx Labs, Bebe
Methods for Names

- Can do morphology (Walters -> Walter, Lucasville)
- Can write stress-shifting rules (Jordan -> Jordanian)
- Rhyme analogy: Plotsky by analogy with Trostsky (replace tr with pl)
- Liberman and Church: for 250K most common names, got 212K (85%) from these modified-dictionary methods, used LTS for rest.
- Can do automatic country detection (from letter trigrams) and then do country-specific rules
Letter to Sound Rules

- Generally machine learning, induced from a dictionary (Black et al. 1998)
- Two steps: alignment and (CART-based) rule-induction
Alignment

Letters: c h e c k e d
Phones: ch _ eh _ k _ t

• Black et al Method 1 (EM)
  • First scatter epsilons in all possible ways to cause letters and phones to align
  • Then collect stats for $P(\text{phone}|\text{letter})$ and select best to generate new stats

$$p(p_i|l_j) = \frac{\text{count}(p_i,l_j)}{\text{count}(l_j)}$$

• Iterate 5-6 times until settles
Alignment

- Black et al method 2
- Hand specify which letters can be rendered as which phones
  - C goes to k/ch/s/sh
  - W goes to w/v/f, etc

\begin{align*}
  c &: k \text{ ch s sh t–s } \epsilon \\
  e &: ih iy er ax ah eh ey uw ay ow y–uw oy aa \epsilon \\
\end{align*}

- Once mapping table is created, find all valid alignments, find \( p(\text{letter} | \text{phone}) \), score all alignments, take best
Alignment

• Some alignments will turn out to be really bad.
• These are just the cases where pronunciation doesn’t match letters:
  Dept  d ih p aa r t m ah n t
  CMU   s iy eh m y uw
  Lieutenant l eh f t eh n ax n t (British)
• Or foreign words
• These can just be removed from alignment training
• Also remove and deal separately with names and acronyms
Black et al. Classifier

A CART tree for each letter in alphabet (26 plus accented) using context of +-3 letters

# # # c h e c c -> ch
che c k e d -> _

• This produces 92-96% correct LETTER accuracy (58-75 word acc) for English
Modern systems: more powerful classifiers, more features
Predicting Intonation in TTS

**Prominence/Accent**: Decide which words are accented, which syllable has accent, what sort of accent

**Boundaries**: Decide where intonational boundaries are

**Duration**: Specify length of each segment

**F0**: Generate F0 contour from these
Predicting Intonation in TTS

**Prominence/Accent:** Decide which words are accented, which syllable has accent, what sort of accent

**Boundaries:** Decide where intonational boundaries are

**Duration:** Specify length of each segment

**F0:** Generate F0 contour from these
Stress vs. accent

- **Stress**: structural property of a word
  - Fixed in the lexicon: marks a potential (arbitrary) location for an accent to occur, if there is one

- **Accent**: property of a word in context
  - Context-dependent. Marks important words in the discourse

<table>
<thead>
<tr>
<th></th>
<th>(x)</th>
<th>(x)</th>
<th>(accented syll)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>stressed syll</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>full vowels</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>syllables</td>
</tr>
<tr>
<td>vi</td>
<td>ta</td>
<td>mins</td>
<td></td>
</tr>
<tr>
<td>Ca</td>
<td>li</td>
<td>for</td>
<td></td>
</tr>
<tr>
<td>nia</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stress vs. accent (2)

- The speaker decides to make the word *vitamin* more prominent by *accenting* it.
- **Lexical stress** tell us that this prominence will appear on the first syllable, hence *vitamin*.
- So we will have to look at both the lexicon and the context to predict the details of prominence.

I’m a little surprised to hear it characterized as up*beat*
Levels of prominence

- Most phrases have more than one accent
- **Nuclear Accent:** Last accent in a phrase, perceived as more prominent
  - Plays semantic role like indicating a word is contrastive or focus.
  - Modeled via ***s in IM, or capitalized letters
  - ‘I know SOMETHING interesting is sure to happen,’ she said
- Can also have reduced words that are **less** prominent than usual (especially function words)
- Sometimes use 4 classes of prominence:
  - Emphatic accent, pitch accent, unaccented, reduced
Pitch accent prediction

• Which words in an utterance should bear accent?

→ i believe at ibm they make you wear a blue suit.
→ i BELIEVE at IBM they MAKE you WEAR a BLUE SUIT.

→ 2001 was a good movie, if you had read the book.
→ 2001 was a good MOVIE, if you had read the BOOK.
SUN MICROSYSTEMS INC, the UPSTART COMPANY that HELPED LAUNCH the DESKTOP COMPUTER industry TREND TOWARD HIGH powered WORKSTATIONS, was UNVEILING an ENTIRE OVERHAUL of its PRODUCT LINE TODAY. SOME of the new MACHINES, PRICED from FIVE THOUSAND NINE hundred NINETY five DOLLARS to seventy THREE thousand nine HUNDRED dollars, BOAST SOPHISTICATED new graphics and DIGITAL SOUND TECHNOLOGIES, HIGHER SPEEDS AND a CIRCUIT board that allows FULL motion VIDEO on a COMPUTER SCREEN.
Predicting Pitch Accent: Part of speech

- Content words are usually accented
- Function words are rarely accented
  - Of, for, in on, that, the, a, an, no, to, and but or will may would can her is their its our there is am are was were, etc
- **Baseline algorithm**: Accent all content words.
Factors in accent prediction:
Contrast

Legumes are poor source of VITAMINS
No, legumes are a GOOD source of vitamins

I think JOHN or MARY should go
No, I think JOHN AND MARY should go
Factors in Accent Prediction: List intonation

I went and saw ANNA, LENNY, MARY, and NORA.
Factors: Word order

Preposed items are accented more frequently

TODAY we will BEGIN to LOOK at FROG anatomy.
We will BEGIN to LOOK at FROG anatomy today.
Factor: Information

- **New information** in an **answer** is often accented

Q1: What types of foods are a good source of vitamins?
A1: LEGUMES are a good source of vitamins.

Q2: Are legumes a source of vitamins?
A2: Legumes are a GOOD source of vitamins.

Q3: I’ve heard that legumes are healthy, but what are they a good source of?
A3: Legumes are a good source of VITAMINS.
Factors: Information Status (2)

New versus old information.
Old information is deaccented

There are LAWYERS, and there are GOOD lawyers

EACH NATION DEFINES its OWN national INTERST.

I LIKE GOLDEN RETRIEVERS, but MOST dogs LEAVE me COLD.
Complex Noun Phrase Structure


- **Proper Names**, stress on right-most word
  
  New York CITY; Paris, FRANCE

- **Adjective-Noun combinations**, stress on noun
  
  Large HOUSE, red PEN, new NOTEBOOK

- **Noun-Noun compounds**: stress left noun
  
  HOTdog (food) versus adj-N HOT DOG (overheated animal)
  
  WHITE house (place) versus adj-N WHITE HOUSE (painted)

- **Other**:
  
  MEDICAL Building, APPLE cake, cherry PIE, kitchen TABLE
  
  Madison avenue, Park street?

- **Rule**: Furniture+Room -> RIGHT
Other features

- POS
- POS of previous word
- POS of next word
- Stress of current, previous, next syllable
- Unigram probability of word
- Bigram probability of word
- Position of word in sentence
Advanced features

• Accent is often deflected away from a word due to focus on a neighboring word.
• Could use syntactic parallelism to detect this kind of contrastive focus:
  • ......driving [FIFTY miles] an hour in a [THIRTY mile] zone
  • [WELD] [APPLAUDS] mandatory recycling. [SILBER] [DISMISSES] recycling goals as meaningless.
  • ...but while Weld may be [LONG] on people skills, he may be [SHORT] on money
Accent prediction

- **Useful features include:** (starting with Hirschberg 1993)
  - Lexical class (function words, clitics not accented)
  - Word frequency
  - Word identity (promising but problematic)
  - Given/New, Theme/Rheme
  - Focus
  - Word bigram predictability
  - Position in phrase
  - Complex nominal rules (Sproat)

- **Combined in a classifier:**
  - Decision trees (Hirschberg 1993), Bagging/boosting (Sun 2002)
  - Hidden Markov models (Hasegawa-Johnson et al 2005)
  - Conditional random fields (Gregory and Altun 2004)
But best single feature: Accent ratio dictionary

Nenkova, J. Brenier, A. Kothari, S. Calhoun, L. Whitton, D. Beaver, and D. Jurafsky. 2007. To memorize or to predict: Prominence labeling in conversational speech. in Proceedings of NAACL-HLT

- Of all the times this word occurred
  - What percentage were accented?
- Memorized from a labeled corpus
  - 60 Switchboard conversations (Ostendorf et al 2001)
- Given:
  - \( k \): number of times a word is prominent
  - \( n \): all occurrences of the word

\[
\text{AccentRatio}(w) = \frac{k}{n}, \quad \text{if } B(k, n, 0.5) < 0.05
\]
Accent Ratio

- Conversational speech dictionaries:
  - [http://www.cis.upenn.edu/~nenkova/AR.sign](http://www.cis.upenn.edu/~nenkova/AR.sign)

- Read News dictionaries:
  - [http://www.cis.upenn.edu/~nenkova/buAR.sign](http://www.cis.upenn.edu/~nenkova/buAR.sign)

- Accent ratio classifier
  - a word not-prominent if AR < 0.38
  - words not in the AR dictionary are labeled “prominent”

- Performance
  - Best single predictor of accent; adding all other features only helps 1% on prediction task

- Improves TTS:
  - Volker Strom, Ani Nenkova, Robert Clark, Yolanda Vazquez-Alvarez, Jason Brenier, Simon King, and Dan Jurafsky. 2007. Modelling Prominence and Emphasis Improves Unit-Selection Synthesis. *Interspeech*
Predicting Intonation in TTS

**Prominence/Accent:** Decide which words are accented, which syllable has accent, what sort of accent

**Boundaries:** Decide where intonational boundaries are

**Duration:** Specify length of each segment

**F0:** Generate F0 contour from these
Predicting Boundaries: Full | | versus intermediate |


Computer phone calls, | | which do everything | from selling magazine subscriptions | | to reminding people about meetings | | have become the telephone equivalent | of junk mail. | |

Doctor Norman Rosenblatt, | | dean of the college | of criminal justice at Northeastern University, | | agrees. | |

For WBUR, | | I’m Margo Melnicove.
Simplest CART

(set! simple_phrase_cart_tree
, ((lisp_token_end_punc in ("?" "." ":"))
  ((BB))
  ((lisp_token_end_punc in ("" ") "," ";"))
  ((B))
  ((n.name is 0) ;; end of utterance
    ((BB))
    ((NB)))))))
More complex features

Ostendorf and Veilleux

- English: boundaries are more likely between content words and function words
- Syntactic structure (parse trees)
  - Largest syntactic category dominating preceding word but not succeeding word
  - How many syntactic units begin/end between words
- Type of function word to right
- Capitalized names
- # of content words since previous function word
Ostendorf and Veilleux CART

- 1092/4210
  - yes: 752/752
  - no: 340/3458
    - yes: 228/690
      - other: 109/212
      - same: 119/478
        - other: 95/307
        - same, WHNP, VP: 24/171
    - no: 112/2768
      - comma: yes: 8/8
      - no: 104/2760
Predicting Intonation in TTS

Prominence/Accent: Decide which words are accented, which syllable has accent, what sort of accent

Boundaries: Decide where intonational boundaries are

**Duration**: Specify length of each segment

**F0**: Generate F0 contour from these
Duration

- Simplest: fixed size for all phones (100 ms)
- Next simplest: average duration for that phone (from training data). Samples from SWBD in ms:
  
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>118</td>
</tr>
<tr>
<td>ax</td>
<td>59</td>
</tr>
<tr>
<td>ay</td>
<td>138</td>
</tr>
<tr>
<td>eh</td>
<td>87</td>
</tr>
<tr>
<td>ih</td>
<td>77</td>
</tr>
<tr>
<td>b</td>
<td>68</td>
</tr>
<tr>
<td>d</td>
<td>68</td>
</tr>
<tr>
<td>dh</td>
<td>44</td>
</tr>
<tr>
<td>f</td>
<td>90</td>
</tr>
<tr>
<td>g</td>
<td>66</td>
</tr>
</tbody>
</table>

- Next Next Simplest: add in phrase-final and initial lengthening plus stress
Klatt duration rules

Models how context-neutral duration of a phone lengthened/shortened by context, while staying above a min duration dmin

**Prepausal lengthening**: vowel before pause lengthened by 1.4

**Non-phrase-final shortening**: Segments not phrase-final are shortened by 0.6. Phrase-final postvocalic liquids and nasals lengthened by 1.4

**Unstressed shortening**: unstressed segments minimum duration dmin is halved, and are shortened by .7 for most phone types.

**Lengthening for accent**: A vowel with accent lengthened by 1.4

**Shortening in clusters**: Consonant followed by consonant shortened by 0.5

**Pre-voiceless shortening**: Vowels are shortened before a voiceless plosive by 0.7
Klatt duration rules

- Klatt formula for phone durations:
  
  \[ d = d_{\text{min}} + \prod_{i=1}^{N} f_i \times (\bar{d} - d_{\text{min}}) \]

- Festival: 2 options
  - Klatt rules
  - Use labeled training set with Klatt features to train CART
    - Identity of the left and right context phone
    - Lexical stress and accent values of current phone
    - Position in syllable, word, phrase
    - Following pause
Duration: state of the art

- Lots of fancy models of duration prediction:
  - Using Z-scores and other clever normalizations
  - Sum-of-products model
  - New features like word predictability
    - Words with higher bigram probability are shorter
Outline

- History, Demos
- Architectural Overview
- Stage 1: Text Analysis
  - Text Normalization
    - Tokenization
    - End of sentence detection
    - Homograph disambiguation
  - Letter-to-sound (grapheme-to-phoneme)
- Prosody
  - Predicting Accents
  - Predicting Boundaries
  - Predicting Duration
  - Generating F0