Unit 1: Sequence Models

Lectures 7-8: Stochastic String Transformations
(a.k.a. “channel-models”)

Professor Liang Huang
liang.huang.sh@gmail.com
String Transformations

- General Framework for many NLP problems

- Examples
  - Part-of-Speech Tagging
  - Spelling Correction (Edit Distance)
  - Word Segmentation
  - Transliteration, Sound/Spelling Conversion, Morphology
  - Chunking (Shallow Parsing)
  - Beyond Finite-State Models (i.e., tree transformations)
    - Summarization, Translation, Parsing, Information Retrieval, ...

- Algorithms: Viterbi (both max and sum)
Review of Noisy-Channel Model

Application
Machine Translation
Optical Character Recognition (OCR)
Part Of Speech (POS) tagging
Speech recognition

Input
$L_1$ word sequences
actual text
POS tag sequences
word sequences

Output
$L_2$ word sequences
text with mistakes
English words
speech signal

$p(i)$
$p(L_1)$ in a language model
prob of language text
prob of POS sequences
prob of word sequences

$p(o|i)$
translation model
model of OCR errors
p(w|t)
acoustic model
(hw2) From Spelling to Sound

- word-based or char-based

![Diagram showing the process from spelling to sound]

```
| P(s) | → | English sound sequence | → | Spell | → | English letter sequence |
```

```
| F | O | Z ? | |
| AY | O | |
| AY | O | P(0.01) |
| O | |
```

```
Homework #1, but with probabilities.
```

```
data: AE R UH N S UH N a a r o n s o n
```

```
P(a a | AE) = 0.04
```
Pronunciation Dictionary

- (hw3: eword-epron.data)  
  [Link to CMU Pronunciation Dictionary](http://www.speech.cs.cmu.edu/cgi-bin/cmudict)
  from CMU Pronunciation Dictionary
  39 phonemes (15 vowels + 24 consonants)

- AARON           EH R AH N
- AARONSON        AA R AH N S AH N
- PEOPLE           P IY P AH L
- VIDEO             V IH D IY OW

You can train $p(s..s|w)$ from this, but what about unseen words?

Also need alignment to train the channel model $p(s|e)$ & $p(e|s)$
CMU Dict: 39 Ame. Eng. Phonemes

WRONG! missing the SCHWA ə (merged with the STRUT ʌ “AH”!)

<table>
<thead>
<tr>
<th>CMU/IPA</th>
<th>Example</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA /ɑ/</td>
<td>odd</td>
<td>AA D</td>
</tr>
<tr>
<td>AE /æ/</td>
<td>at</td>
<td>AE T</td>
</tr>
<tr>
<td>AH /ʌ/</td>
<td>hut</td>
<td>HH AH T</td>
</tr>
<tr>
<td>AO /ɔː/</td>
<td>ought</td>
<td>AO T</td>
</tr>
<tr>
<td>AW /aʊ/</td>
<td>cow</td>
<td>K AW</td>
</tr>
<tr>
<td>AY /aɪ/</td>
<td>hide</td>
<td>HH AY D</td>
</tr>
<tr>
<td>B /b/</td>
<td>be</td>
<td>B IY</td>
</tr>
<tr>
<td>CH /tʃ/</td>
<td>cheese</td>
<td>CH IY Z</td>
</tr>
<tr>
<td>D /d/</td>
<td>dee</td>
<td>D IY</td>
</tr>
<tr>
<td>DH /ð/</td>
<td>thee</td>
<td>DH IY</td>
</tr>
<tr>
<td>EH /ɛ/</td>
<td>Ed</td>
<td>EH D</td>
</tr>
<tr>
<td>ER /ɜ/</td>
<td>hurt</td>
<td>HH ER T</td>
</tr>
<tr>
<td>EY /eɪ/</td>
<td>ate</td>
<td>EY T</td>
</tr>
<tr>
<td>F /f/</td>
<td>fee</td>
<td>F IY</td>
</tr>
<tr>
<td>G /g/</td>
<td>green</td>
<td>G R IY N</td>
</tr>
<tr>
<td>HH /h/</td>
<td>he</td>
<td>HH IY</td>
</tr>
<tr>
<td>IH /ɪ/</td>
<td>it</td>
<td>IH T</td>
</tr>
<tr>
<td>IY /iː/</td>
<td>eat</td>
<td>IY T</td>
</tr>
<tr>
<td>JH /dʒ/</td>
<td>gee</td>
<td>JH IY</td>
</tr>
<tr>
<td>K /k/</td>
<td>key</td>
<td>K IY</td>
</tr>
<tr>
<td>L /l/</td>
<td>lee</td>
<td>L IY</td>
</tr>
<tr>
<td>M /m/</td>
<td>me</td>
<td>M IY</td>
</tr>
<tr>
<td>N /n/</td>
<td>knee</td>
<td>N IY</td>
</tr>
<tr>
<td>NG /ŋ/</td>
<td>ping</td>
<td>P IY NG</td>
</tr>
<tr>
<td>OW /ɔʊ/</td>
<td>oat</td>
<td>OW T</td>
</tr>
<tr>
<td>OY /ɔɪ/</td>
<td>toy</td>
<td>T OY</td>
</tr>
<tr>
<td>P /p/</td>
<td>pee</td>
<td>P IY</td>
</tr>
<tr>
<td>R /r/</td>
<td>read</td>
<td>R IY D</td>
</tr>
<tr>
<td>S /s/</td>
<td>sea</td>
<td>S IY</td>
</tr>
<tr>
<td>SH /ʃ/</td>
<td>she</td>
<td>SH IY</td>
</tr>
<tr>
<td>T /t/</td>
<td>tea</td>
<td>T IY</td>
</tr>
<tr>
<td>TH /θ/</td>
<td>theta</td>
<td>TH EY T AH</td>
</tr>
<tr>
<td>UH /ʌ/</td>
<td>hood</td>
<td>HH UH D</td>
</tr>
<tr>
<td>UW /u/</td>
<td>too</td>
<td>T UW</td>
</tr>
<tr>
<td>V /v/</td>
<td>vee</td>
<td>V IY</td>
</tr>
<tr>
<td>W /w/</td>
<td>we</td>
<td>W IY</td>
</tr>
<tr>
<td>Y /j/</td>
<td>yield</td>
<td>Y IY L D</td>
</tr>
<tr>
<td>Z /z/</td>
<td>zee</td>
<td>Z IY</td>
</tr>
<tr>
<td>ZH /ʒ/</td>
<td>usual</td>
<td>Y UW ZH UW AH L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
WRONG! missing the SCHWA \( \text{ə} \) (merged with the STRUT \( \text{ʌ} \) “AH”):
DOES NOT ANNOTATE STRESSES

A          AH
A          EY
AAA        T R IH P AH L EY
AABERG     AA B ER G
AACHEN     AA K AH N
...  
ABOUT      AH B AW T
...
ABRAMOVITZ AH B R AA M AH V IH T S
ABRAMOWICZ AH B R AA M AH V IH CH
ABRAMOWITZ AH B R AA M AH W IH T S
...
FATHER     F AA DH ER
...
ZYDECO     Z AY D EH K OW
ZYDECO     Z IH D AH K OW
ZYDECO     Z AY D AH K OW
...
ZZZZ       Z IY Z
**Linguistics Background: IPA**

### CONSONANTS (PULMONIC)

<table>
<thead>
<tr>
<th></th>
<th>Bilabial</th>
<th>Labiodental</th>
<th>Dental</th>
<th>Alveolar</th>
<th>Postalveolar</th>
<th>Retroflex</th>
<th>Palatal</th>
<th>Velar</th>
<th>Uvular</th>
<th>Pharyngeal</th>
<th>Glottal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plosive</td>
<td>p b</td>
<td></td>
<td>t d</td>
<td></td>
<td>t q</td>
<td>c j</td>
<td>k g</td>
<td>q g</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>m nj</td>
<td>m</td>
<td>n</td>
<td></td>
<td>n</td>
<td>n</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trill</td>
<td>B r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tap or Flap</td>
<td>v R</td>
<td>v</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fricative</td>
<td>Ø β φ f v</td>
<td>θ ð s z</td>
<td>s z</td>
<td>j z</td>
<td>s z</td>
<td>c j</td>
<td>x y</td>
<td>χ β</td>
<td>h j</td>
<td>h j</td>
<td></td>
</tr>
<tr>
<td>Lateral fricative</td>
<td></td>
<td>I j</td>
<td>l j</td>
<td>I j</td>
<td>l j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximant</td>
<td>v u</td>
<td>i</td>
<td>l</td>
<td>l j</td>
<td>l j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lateral approximant</td>
<td></td>
<td>l</td>
<td>l</td>
<td>l α</td>
<td>l α</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

**VOWELS**

Where symbols appear in pairs, the one to the right represents a rounded vowel.
(hw2) From Sound s to Spelling e

- input: HH EH L OW B EH R
- output: H E L L O B E A R or H E L O B A R E ?
- $p(e) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(s|w) \Rightarrow s$
- $e \leq p(e|s) \leq s \leq p(s)$
- $w \leq p(w|e) \leq e \leq p(e|s) \leq s \leq p(s)$
- $w \leq p(w|s) \leq s \leq p(s)$
- what else? echo 'HH EH L OW' | carmel -sliOEQk 50 epron-espell.wfst espell-eword.wfst eword.wfsa
Example: Transliteration

- **KEVIN KNIGHT** => KH EH VH IH N N A Y T
  KE BI N NA I T O

- **V => B**: phoneme inventory mismatch
- **T => T O**: phonotactic constraint
Japanese writing system has four components

- **Kanji (Chinese chars):** nouns, verb/adj stems, CJKV names
  - 日本 “Japan” 东京 “Tokyo” 电车 “train” 食べる “eat [inf.]”

- **Syllabaries**
  - **Hiragana:** function words (e.g. particles), suffixes
    - で de (“at”) か ka (question) 食べました “ate”
  - **Katakana:** transliterated foreign words/names
    - コーヒー koohii (“coffee”)

- **Romaji (Latin alphabet):** auxiliary purposes
Annual Kanji of the Year
Why Japanese uses Syllabaries

- all syllables are: [consonant] + vowel + [nasal n]
- $10 \times 5 = 50$ syllables
  - plus some variations
- also possible for Mandarin
- other languages have many more syllables: use alphabets
  - alphabet = 10+5; syllabary = 10x5
- read the Writing Systems tutorial from course page!

![Japanese Syllabary Chart]

[http://brng.jp/90459562](http://brng.jp/90459562)
Origins of Katakana

- they come from Chinese characters (kanji’s)
- but often got the wrong (semantic not phonetic) radicals...

<table>
<thead>
<tr>
<th>Hiragana</th>
<th>Katakana</th>
</tr>
</thead>
<tbody>
<tr>
<td>ア い う え お ゛ ィ ァ カ キ ク カ ケ ク シ チ テ テ ト ナ ヌ ヌ ネ ネ ノ ハ ヒ ヒ フ フ ヘ ヘ ホ マ ミ ミ ム ム メ メ モ モ ヤ ユ ユ ヨ ヨ ラ リ リ ル ル レ レ ロ ロ ワ ワ ワ ワ わ わ わ れ れ れ れ る る る る ル ル ル ル ユ ユ ユ ユ ウ ウ ウ ウ エ エ エ エ オ オ オ オ ン ン ン ン</td>
<td></td>
</tr>
</tbody>
</table>

Kana Development Chart
# Japanese Phonemes (too few sounds!)

## CONSONANTS (PULMONIC)

<table>
<thead>
<tr>
<th></th>
<th>Bilabial</th>
<th>Labiodental</th>
<th>Dental</th>
<th>Alveolar</th>
<th>Post alveolar</th>
<th>Retroflex</th>
<th>Palatal</th>
<th>Velar</th>
<th>Uvular</th>
<th>Pharyngeal</th>
<th>Glottal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plosive</td>
<td>p    b</td>
<td>d</td>
<td>t</td>
<td>d</td>
<td>q</td>
<td>k</td>
<td>g</td>
<td>q</td>
<td>G</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>m</td>
<td>n</td>
<td>η</td>
<td>η</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trill</td>
<td>B</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tap or Flap</td>
<td>v</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fricative</td>
<td>φ</td>
<td>β</td>
<td>f</td>
<td>v</td>
<td>θ</td>
<td>δ</td>
<td>s</td>
<td>z</td>
<td>ʃ</td>
<td>ʒ</td>
<td>ɕ</td>
</tr>
<tr>
<td>Lateral fricative</td>
<td></td>
<td>ɬ</td>
<td>ɮ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximant</td>
<td></td>
<td>j</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lateral approximant</td>
<td></td>
<td>l</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

## VOWELS

- **Front**: i, y, I, Y, i, u, U, u
- **Central**: ø, θ, ø, θ, ø, ø, ø, ø
- **Back**: æ, øæ, æ, æ, æ, æ, æ, æ

Where symbols appear in pairs, the one to the right represents a rounded vowel.
Aside: Is Korean a Syllabary? No!

- A: Hangul is not a syllabary, but a “featural alphabet”
- a special alphabet where shapes encode phonological features
- the inventor of Hangul (c. 1440s) was the first real linguist

14 consonants: ꧏg, ꧏn, ꧏd, ꧏl/r, ꧏm, ꧏb, ꧏs, ꧏnull/ng, ꧏj, ꧏch, ꧏk, ꧏt, ꧏp, ꧏh
5 double consonants: ꧏkk, ꧏtt, ꧏpp, ꧏss, ꧏjj
11 consonant clusters: ꧏgs, ꧏnj, ꧏnh, ꧏlg, ꧏlm, ꧏlb, ꧏls, ꧏlt, ꧏlp, ꧏlh, ꧏbs
6 vowel letters: ꧏa, ꧏo, ꧏeo, ꧏi, ꧏu, ꧏeu, ꧏi
4 iotized vowels (with a y): ꧏya, ꧏyeo, ꧏyo, ꧏyu
5 (iotized) diphthongs: ꧏae, ꧏyae, ꧏe, ꧏye, ꧏui
6 vowels and diphthongs with a w: ꧏwa, ꧏwae, ꧏwe, ꧏwo, ꧏwe, ꧏwi

Q: 강남 스타일 = ?
Katakana Transliteration Examples

- コンピューター
  - kon pyu - ta -
  - kompyuutaa (uu=û)
  - computer

- アイスクリーム
  - a i su ku ri - mu
  - aisukuriimu
  - ice cream

- アンドリュー・ビタビ
  - andoryuubitabi
  - Andrew Viterbi

- ヨーグルト
  - yo - gu ru to
  - yogurt
Japanese just transliterates almost everything (even though its syllable inventory is really small...) but... it is quite easy for English speakers to decode .... if you have a good language model!

koohiikoonaa      coffee corner
saabisu               service
bulendokooohii     blend coffee
sutoreetokooohii   straight coffee
juusu                juice
aisukuriimu         ice cream
tooosuto             toast

from Knight & Sproat 09
Katakana on Streets of Tokyo
More Japanese Transliterations

- rapputoppu  ラプトプ  • laptop
- bideoteepu  ビデオテープ  • video tape
- shoppingusentaa  ショッピングセンター  • shopping center
- shiitoberuto  シートベルト  • seat belt
- chairudoshiito  チャイルドシート  • child seat
- andoryuubitabi  アンドリュー・ビタビ  • Andrew Viterbi
- bitabiarugorizumu  ビタビアルゴリズム  • Viterbi Algorithm
(hw2) Katakana => English

• your job in HW2: decode Japanese Katakana words (transcribed in Romaji) back to English words

• koohiikoonaa => coffee corner

[Knight & Graehl 98]
(hw2) Katakana => English

- Decoding (HW3)
  - really decipherment!
- what about duplicate strings?
  - from different paths in WFST!
- n-best cruching, or...
- weighted determinisation
  - see extra reading on course website for Mohri+Riley paper

[Angel Knight, Angela Nite, Andy Law Knight, Angela Nate] + millions more
[Ann Gere Uh, Anne Jill Ahh, Angry Rugh, Ann Zillah] + millions more
WFSA A
WFSA B
WFSA C
WFSA D
an_jiranaito
アンジラナイト

[Knight & Graehl 98]
How to Learn $p(e|w)$ and $p(j|e)$?

- Such a system captures an infinite relation of <sound-sequence, writing-sequence> pairs.

**HW2**
epron-jpron.data

**HW3**
Viterbi decoding

**HW4**
epron-jpron.data
(MLE)

Learning Sequence Transformation Probabilities

**Ideal training data:**

```
L A E M P
r a n p u
```

```
S T I Y M
s u t i i m u
```

$P(n | M) = 0.5$

$P(m u | M) = 0.5$

etc

**Actual training data:**

```
L A E M P
r a n p u
```

```
S T I Y M
s u t i i m u
```

etc

Automatically align string pairs using the unsupervised Expectation-Maximization (EM) algorithm.
String Transformations

- General Framework for many NLP problems
- Examples
  - Part-of-Speech Tagging
  - Spelling Correction (Edit Distance)
  - Word Segmentation
  - Transliteration, Sound/Spelling Conversion, Morphology
  - Chunking (Shallow Parsing)
  - Beyond Finite-State Models (i.e., tree transformations)
    - Summarization, Translation, Parsing, Information Retrieval, ...
- Algorithms: Viterbi (both max and sum)
Example 2: Part-of-Speech Tagging

- Use tag bigram as a language model
- Channel model is context-indep.

\[ P(t_1 \ldots t_n | w_1 \ldots w_n) \]

\[ \sim P(t_1) \cdot P(t_2 | t_1) \ldots P(t_n | t_{n-1}) \cdot P(w_1 | t_1) \ldots P(w_n | t_n) \]

- Local grammar preference
- Lexical preference

\[ \text{Source} \]

\[ \text{Channel} \]

\[ \text{New string} \]

\[ t_1 : w_4 / P(w_4 | t_1) \]

\[ t_2 : w_5 / P(w_5 | t_2) \]

\[ t_3 : w_6 / P(w_6 | t_3) \]

\[ \text{they can fish} \]

\[ \text{I saw her earrings.} \]

\[ \text{NOTE: Choice of tag is influenced by both left & right context.} \]

\[ \text{NOTE: Influences can theoretically be long ranging.} \]
Work out the compositions

• if you want to implement Viterbi...

• case 1: language model is a tag unigram model
  • \( p(t_1...t_n) = p(t_1)p(t_2)...p(t_n) \)
  • how many states do you get?

• case 1: language model is a tag bigram model
  • \( p(t_1...t_n) = p(t_1)p(t_2 | t_1)...p(t_n | t_{n-1}) \)
  • how many states do you get?

• case 3: language model is a tag trigram model...
The case of bigram model

class-dependence (from LM) propagates left and right!
In general...

- bigram LM with context-independent CM
  - $O(n \cdot m)$ states after composition

- $g$-gram LM with context-independent CM
  - $O(n \cdot m^{g-1})$ states after composition
  - the $g$-gram LM itself has $O(m^{g-1})$ states
HMM Representation

- HMM representation is not explicit about the search
  - “hidden states” have choices over “variables”
- In FST composition, paths/states are explicitly drawn
Viterbi for argmax

Viterbi search for \( \arg\max p(t_\cdots t) \cdot p(w_\cdots w | t_\cdots t) \):

for \( j = 1 \) to \( m \)
\[
Q[1,j] = P(t_j) \cdot p(w_1 | t_j)
\]

for \( i = 2 \) to \( n \)
for \( j = 1 \) to \( m \)
\[
Q[i,j] = 0 \\
\text{best-pred}[i,j] = 0 \\
\text{best-score} = -\infty \\
\text{for } k = 1 \text{ to } m \\
\quad r = P(t_j | t_k) \cdot p(w_i | t_j) \cdot Q[i-1,k] \\
\quad \text{if } r > \text{best-score} \\
\quad \quad \text{best-score} = r \\
\quad \quad \text{best-pred}[i,j] = k \\
\quad \quad Q[i,j] = r \\
\]
\[
\text{final-best} = 0 \\
\text{final-score} = -\infty \\
\text{for } j = 1 \text{ to } m \\
\quad \text{if } Q[n,j] > \text{final-score} \\
\quad \quad \text{final-score} = Q[n,j] \\
\quad \quad \text{final-best} = j \\
\]
\[
\text{print } t_{\text{final-best}} \\
\text{current} = \text{final-best} \\
\text{for } i = n-1 \text{ down to } 1 \\
\quad \text{current} = \text{best-prev}[i+1, \text{current}] \\
\quad \text{print } t_{\text{current}}
\]

\( Q[i,j] \): cost of shortest path ending with word \( i \) getting assigned tag \( j \).

how about unigram?
Complete this Python code implementing the Viterbi algorithm for part-of-speech tagging. It should print a list of word/tag pairs, e.g. [('a', 'D'), ('can', 'N'), ('can', 'A'), ('can', 'V'), ('a', 'D'), ('can', 'N')].

```python
from collections import defaultdict

best = defaultdict(lambda: defaultdict(float))
best[0]["<s>"] = 1
back = defaultdict(dict)

words = "<s> a can can can a can </s>".split()

tags = {"a": ["D"], "can": ["N", "A", "V"], "</s>": ["</s>"], ...}  # possible tags for each word
ptag = {"D": {"N": 1}, "V": {"</s>": 0.5, "D":0.5}, ...}  # ptag[x][y] = p(y | x)
pword = {"D": {"a": 0.5}, "N": {"can": 0.1}, ...}  # pword[x][w] = p(w | x)

for i, word in enumerate(words[1:], 1):
    for tag in tags[word]:
        for prev in best[i-1]:
            if tag in ptag[prev]:
                score = best[i-1][prev] * ptag[prev][tag] * pword[tag][word]
                if score > best[i][tag]:
                    best[i][tag] = score
                    back[i][tag] = prev

def backtrack(i, tag):
    if i == 0:
        return []
    return backtrack(i-1, back[i][tag]) + [(words[i], tag)]

print(backtrack(len(words)-1, "</s>"))[::-1]
```

Q: what about top-down recursive + memoization?
### Viterbi Tagging Example

#### Q1. why is this table *not* normalized?

#### Q2. is “fish” equally likely to be a V or N?

#### Q3: how to train $p(w|t)$?

<table>
<thead>
<tr>
<th></th>
<th>PRO</th>
<th>V</th>
<th>N</th>
<th>AUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PRO</td>
<td>.6</td>
<td>.6</td>
<td>.9</td>
<td>.9</td>
</tr>
<tr>
<td>V</td>
<td>.3</td>
<td>.9</td>
<td>.7</td>
<td>.9</td>
</tr>
<tr>
<td>N</td>
<td>.05</td>
<td>.2</td>
<td>.1</td>
<td>.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PRO</th>
<th>V</th>
<th>N</th>
<th>AUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>they</td>
<td>.07</td>
<td>$10^{-5}$</td>
<td>$10^{-4}$</td>
<td>.21</td>
</tr>
<tr>
<td>can</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>.21</td>
<td></td>
</tr>
<tr>
<td>fish</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>.21</td>
<td></td>
</tr>
</tbody>
</table>

#### Pro

- $Q = P(\text{PRO}|\text{START}) \cdot P(\text{they}|\text{PRO})$
- $Q = 0 \quad P(\text{PRO}|\text{PRO}) = 0\quad P(\text{can}|\text{PRO}) = 0$

- $Q = 0 \quad P(\text{they}|V) = 0$

- $Q = 0 \quad P(\text{N}|\text{PRO}) = 0$

- $Q = 0 \quad P(\text{fish}|\text{PRO}) = 0$

#### V

- $Q = \max \left< 0.042 \cdot 0.1 \cdot 10^{-5} \right>$
- $Q = \max \left< 0.00000252 \cdot 0 \cdot 10^{-4} \right>$

- $Q = 0\quad\text{bp = PRO}$

#### N

- $Q = 0\quad\text{bp = PRO}$

#### AUX

- $Q = \max \left< 0.02646 \cdot 2.21 \right>$
- $Q = 0\quad\text{bp = PRO}$

$$Q[i,j] = P(t_j|\text{START}) \cdot P(w_i|t_j)$$

$$Q[i,j] = \max_k Q[i-1,k] \cdot P(t_j|t_k) \cdot P(w_i|t_j)$$
Trigram HMM

for $j = 1$ to $m$
  $Q_{1}[1,j] = \ldots$

for $j = 1$ to $m$
  for $j_2 = 1$ to $m$
    $Q[2,j,j_2] = \ldots$

for $i = 3$ to $n$
  for $j = 1$ to $m$
    for $j_2 = 1$ to $m$
      $Q[i,j,j_2] = 0$
      best-pred[$i,j,j_2] = 0$
      best-score = $-\infty$
      for $k = 1$ to $m$
        $r = P(t_{j_2} \mid t_j) \cdot P(w_i \mid t_{j_2}) \cdot Q[i-1,k,j]$,
        if $r >$ best-score

Time complexity: $O(nT^3)$
In general: $O(nT^g)$ for g-gram
A Side Note on Normalization

**NOTE**

final-best gives $P(t...t) \cdot P(w...w \mid t...t)$

but this is not the same as $P(t...t \mid w...w)$

e.g. suppose there is only one $t...t$ (all words unambiguous)

then $P(t...t \mid w...w) = 1$

need to divide

$$P(t...t \mid w...w) = \frac{P(t...t) \cdot P(w...w \mid t...t)}{P(w...w)} = \frac{P(t...t) \cdot P(w...w \mid t...t)}{\sum_{t...t} P(t...t) \cdot P(w...w \mid t...t)}$$

how to compute the normalization factor?
Forward (sum instead of max)

Forward search: \( \sum_t P(t) \cdot P(w|t) = P(w) \)

\[ \alpha[1, j] = P(t_j|\text{START}) \cdot P(w_i|t_j) \]

\[ \alpha[i, j] = \sum_k \alpha[i-1, k] \cdot P(t_j|t_k) \cdot P(w_i|t_j) \]

no back pointer

\[ P(w) = \sum_k \alpha[n, k] \]

"Forward" procedure for \( P(w...w) \)

for \( j = 1 \) to \( m \)

\[ \alpha[1, j] = P(t_j) \cdot P(w_i|t_j) \]

for \( i = 2 \) to \( n \)
  for \( j = 1 \) to \( m \)
    \[ \alpha[i, j] = 0 \]
    for \( k = 1 \) to \( m \)
      \[ \alpha[i, j] += P(t_j|t_k) \cdot P(w_i|t_j) \cdot \alpha[i-1, k] \]

\[ P(w...w) = 0 \]

for \( j = 1 \) to \( m \)

\[ P(w...w) += \alpha[n, j] \]
Forward vs. Argmax

- same complexity, different semirings (+, x) vs (max, x)
- for g-gram LM with context-indep. CM

- time complexity \( O(n m^g) \)  space complexity \( O(n m^{g-1}) \)

```
for j = 1 to m
  Q_1[1,j] = ...

for j = 1 to m
  for j2 = 1 to m
    Q[2,j,j2] = ...

for i = 3 to n
  for j = 1 to m
    for j2 = 1 to m
      Q[i,j,j2] = 0
      best-pred[i,j,j2] = 0
      best-score = -\infty
    for k = 1 to m
      r = P(t_{j2,k} \mid t_j) \cdot P(w_i \mid t_{j2}) \cdot Q[i-1,k,j]
      if r > best-score ...
```

\( O(nm^3) \) complexity
Viterbi for DAGs with Semiring

1. topological sort

2. visit each vertex \( v \) in sorted order and do updates
   - for each incoming edge \((u, v)\) in \( E \)
   - use \( d(u) \) to update \( d(v) \):
     \[
     d(v) \oplus = d(u) \otimes w(u, v)
     \]
   - key observation: \( d(u) \) is fixed to optimal at this time

- time complexity: \( O(V + E) \)

see tutorial on DP from course page
Example: Word Segmentation

- you noticed that Japanese (e.g., Katakana) is written without spaces between words
- in order to guess the English you also do segmentation
- e.g. アイスクリーム => アイス クリーム => ice cream
- how about “gaaruhurendo” and “shingurururumu”?
- this is an even more important issue in Chinese
  - 南京市长江大桥
- also in other East Asian Languages
- also in English: sounds => words (speech recognition)
What if English were written as Chinese...

- thisisacoursetaughtinthefallsemesterofthisyearatusc
- actually, Latin used to be written exactly like this!
  - “scripta continua” => “interpuncts” (center dots) =>
- this might be a final project topic (on the easier side)
Chinese Word Segmentation

“democracy”

“President Jiang Zemin”

people-dominate

min-zhu

now Google is good at segmentation!

this was 5 years ago.

graph search

tagging problem

xia yu tian di mian ji shui

江泽民 主席

jiang-ze-min zhu-xi

... - ... - people dominate-podium

下 雨 天 地 面 积 水

Liang Huang (Penn)
Word Segmentation Cascades

- a good idea for final project (Chinese/Japanese)
Machine Translation

- simplest model: word-substitution and permutation
- does it really work??
how would you model permutation in FSTs?

Diagram: 
- English word sequence: $P(e)$
- Substitute: $P(f'|e)$
- French words in English word order: $P(f|f')$
Phrase-based Decoding

与 沙龙 举行 了 会谈

yu Shalong juxing le huitan

held a talk with Sharon

with Sharon held talks

with Sharon held a talk

yu Shalong juxing le huitan
Phrase-based Decoding

与 沙龙 举行 了 会谈

yu Shalong juxing le huitan

held a talk with Sharon

with Sharon held talks

with Sharon held a talk

yu Shalong juxing le huitan
Phrase-based Decoding

与 沙龙 举行 了 会谈

yu Shalong juxing le huitan

held a talk with Sharon

source-side: coverage vector

held a talk

target-side: grow hypotheses

strictly left-to-right

space: $O(2^n)$, time: $O(2^n n^2)$ -- cf. traveling salesman problem
Phrase-based Cascades

- **english LM => (english) => phrase substitutions ($n^2$) => (foreign phrases in English word order) => permutations ($2^n$) => (foreign)**

- a good idea for final project (on the harder end)

- wait, where does the phrase table come from?
  - => word-aligned English-foreign sentence pairs
Traveling Salesman Problem & MT

- a classical NP-hard problem
- goal: visit each city once and only once
- exponential-time dynamic programming
  - state: cities visited so far (bit-vector)
  - search in this $O(2^n)$ transformed graph
- MT: each city is a source-language word
  - restrictions in reordering can reduce complexity $\Rightarrow$ distortion limit
- $\Rightarrow$ syntax-based MT

(Held and Karp, 1962; Knight, 1999)
Example: Edit Distance

- a) given $x, y$, what is $p(y|x)$?
- b) what is the most likely seq. of operations?
- c) given $x$, what is the most likely output $y$?
- d) given $y$, what is the most likely input $x$ (with LM)?
Edit Distance can model...

- part-of-speech tagging
- transliteration
- sound-spelling conversion
- word-segmentation
Given x and y...

- given x, y: a) what is \( p(y \mid x) \)? (sum of all paths)
  b) what is the most likely conversion path?

Best path (by Dijkstra’s algorithm)

\[ P(\text{Ab} \mid ab) = 0.16 \]
Example: General Tagging

```
Source

Channel

Composition

State

Position in observed string

Store Q[i,j] best score to here
Ψ[i,j] backpointer to best pred
α[i,j] sum of scores to here

"a capital crime"
```

s1  s2  s3
state names

p1  p2  p3  p4
c) given correct English $x$, what’s the corrupted $y$ with the highest score?

\[ x = "ab" \]

\[ \begin{align*} a:a/0.2 \quad b:b/0.2 \\ a:A/0.8 \quad b:B/0.8 \end{align*} \]

\[ \begin{align*} a:a/0.2 \quad b:b/0.2 \\ a:A/0.8 \quad b:B/0.8 \end{align*} \]

\[ \begin{align*} a:A/0.8 \quad b:B/0.8 \end{align*} \]

so, $\text{argmax}_y P(y|x) = AB$
DP for “most likely corrupted”

\[ x = "ab" \]

- A: a/0.2
- B: b/0.2

\[ \begin{array}{c}
\text{remove input symbols} \\
\text{find best path}
\end{array} \]

\[ \text{so, } \arg\max_y P(y|x) \text{ of } ab \text{ is } AB \]
d) Most Likely “Original Input”

- using an LM $p(e)$ as source model for *spelling correction*
- case 1: letter-based language model $p_L(e)$
- case 2: word-based language model $p_W(e)$

How would dynamic programming work for cases 1/2?
Dynamic Programming for d)

- given \( y \), what is the most likely \( x \) with \( \max p(x) p(y|x) \)
Beyond Finite-State Models

- sentence summarization

\[
\arg \max_S P(s) \cdot P(l|s)
\]

Original short sentence (encourage grammaticality)

Particular "lengthening" (encourage addition of optional material only; e.g.,
\[
P(\text{he didn't go} | \text{he did go}) = \text{LOW}
P(\text{the big dog} | \text{the dog}) = \text{MED}
P(\text{off of the couch} | \text{off the couch}) = \text{HIGH}
\]
Beyond Finite-State Models

- headline generation

$$\text{argmax}_h P(h) \cdot P(d|h)$$

Looks like a proper headline

If this were a headline, d would be a reasonable document to go with it (i.e., d fleshes out h).
Beyond Finite-State Models

- information retrieval

\[ P(d) \rightarrow \text{document} \rightarrow P(q|d) \rightarrow \text{query} \]

used to rank documents, not construct new ones!
query may contain words not in document.