Unit 1: Sequence Models

Lectures 7-8: HMMs, Tagging and Transliteration

required

hard

optional

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

#### $p(o|i)$
- translation model
- model of OCR errors
- $p(w|t)$

### Diagram

```
WFSA   t...t   WFST   w...w
```

### Table

| Application                  | Input                | Output                  | $p(i)$                                      | $p(o|i)$          |
|------------------------------|----------------------|-------------------------|---------------------------------------------|-------------------|
| Machine Translation          | $L_1$ word sequences | $L_2$ word sequences   | $p(L_1)$ in a language model                | translation model |
| Optical Character Recognition (OCR) | actual text       | text with mistakes      | prob of language text                       | model of OCR errors |
| Part Of Speech (POS) tagging | POS tag sequences   | English words           | prob of POS sequences                       | $p(w|t)$          |
| Speech recognition           | word sequences      | speech signal           | prob of word sequences                      | acoustic model    |
Example 1: Part-of-Speech Tagging

- use tag bigram as a language model
- channel model is context-indep.
Work out the compositions

• if you want to implement Viterbi...

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

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

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

context-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 $\text{argmax } p(t_1 \cdots t_n) \cdot p(w_1 \cdots w_l | t_1 \cdots t_n)$:

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

for $i = 2$ to $n$
  for $j = 1$ to $m$
    $Q[i, j] = 0$
    best-prev[i, j] = 0
    best-score = $-\infty$
    for $k = 1$ to $m$
      $r = P(t_j | t_k) \cdot p(w_i | t_j) \cdot Q[i-1, k]$
      if $r >$ best-score
        best-score = $r$
        best-prev[i, j] = $k$
        $Q[i, j] = r$

final-best = 0
final-score = $-\infty$
for $j = 1$ to $m$
  if $Q[n, j] >$ final-score
    final-score = $Q[n, j]$
    final-best = $j$

print $t_{\text{final-best}}$
current = final-best
for $i = n-1$ down to 1
  current = best-prev[i+1, current]
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>"], ...
ptag = {"D": {"N": 1}, "V": {"</s>": 0.5, "D":0.5}, ...

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 $p(w|t)$?
Trigram HMM

\[
\text{for } j = 1 \text{ to } m \\
Q_1[1, j] = \ldots
\]

\[
\text{for } j = 1 \text{ to } m \\
\text{for } j_2 = 1 \text{ to } m \\
Q[2, j, j_2] = \ldots
\]

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

time complexity: \(O(nT^3)\)
in general: \(O(nT^g)\) for g-gram
A Side Note on Normalization

**NOTE**

final-best gives \( P(t \cdots t) \cdot P(w \cdots w | t \cdots t) \)

but this is not the same as \( P(t \cdots t | w \cdots w) \)

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

then \( P(t \cdots t | w \cdots w) = 1 \)

need to divide

\[
P(t \cdots t | w \cdots w) = \frac{P(t \cdots t) \cdot P(w \cdots w | t \cdots t)}{P(w \cdots w)} = \frac{P(t \cdots t) \cdot P(w \cdots w | t \cdots t)}{\sum_{t \cdots t} P(t \cdots t) \cdot P(w \cdots w | t \cdots t)}
\]

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 $O(n m^{g-1})$

```plaintext
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} | t_j) \cdot P(w_i | t_{j2}) \cdot Q[i-1, k, j] 
      if r > best-score ...
```
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
(hw3) From Spelling to Sound

- word-based or char-based

```
\[
P(s) \xrightarrow{\text{english sounds sequence}} P(e|s) \xrightarrow{\text{english letter sequence}}
\]

```

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) 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 /i/</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 IH NG</td>
</tr>
<tr>
<td>OW /oʊ/</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 /ʊ/</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>
</tbody>
</table>
WRONG! missing the SCHWA ə (merged with the STRUT 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
...
ZZZZZ      Z IY Z
IPA and English Phonology

<table>
<thead>
<tr>
<th>CONSONANTS (PULMONIC)</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>t d</td>
<td>t d</td>
<td>c j</td>
<td>k g</td>
<td>q g</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nasal</td>
<td>m n</td>
<td>n</td>
<td>n n</td>
<td>n</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trill</td>
<td>B r</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 f</td>
<td>t</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 β  f v</td>
<td>θ δ s z</td>
<td>s z</td>
<td>c j</td>
<td>x y</td>
<td>h i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lateral fricative</td>
<td></td>
<td>l h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximant</td>
<td>u l</td>
<td>l j</td>
<td>l</td>
<td>j</td>
<td>u w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lateral approximant</td>
<td></td>
<td>l 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.

coarticulated

tʃ dʒ w

CS 562 - Lec 11-13: String Transformations

VOWELS

Front
- i
- y

Central
- i
- u

Back
- u

Where symbols appear in pairs, the one to the right represents a rounded vowel.
(hw3) From Sound to Spelling

- **input:** HH EH L OW B EH R
- **output:** H E L L O B E A R or H E L L O B A R E?

- $p(e) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(s|w) \Rightarrow s$
- $p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s \Rightarrow p(s)$
- $p(w) \leq w \leq p(w|e) \leq e \leq p(e|s) \leq s \leq p(s)$
- $w \leq p(w|s) \leq s \leq p(s)$
- **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 NAY T
- KE B I N NA I T O
- チェビン ナイト

- V => B: phoneme inventory mismatch
- T => T O: phonotactic constraint
Japanese 101 (writing systems)

- 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), suffices
    - 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
Why Japanese uses Syllabaries

- all syllables are: [consonant] + vowel + [nasal n]
- 10 C x 5 V = 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!

CS 562 - Lec 11-13: String Transformations
Japanese Phonology (too few sounds!)

but Japanese has many allophones:

/s/ => [ɕ] “si” => “shi”  (similar to [ʃ])
/t/ => [tɕ] “ti” => “chi”  (similar to [tʃ])
/t/ => [ts] “tu” => “tsu”
/h/ => [ɸ] “hu” => “hu/fu” [ɸɯ] ... etc ...

allophones: variations of a phoneme depending on context that does not change the meaning (like isotopes vs. element). English has many:
/p/ => [pʰiː] or [spiː] (also for t and k) etc.
Aside: Is Korean a Syllabary?

• 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: ꆇg, ꆇn, ꆇd, ꆇl/r, ꆇm, ꆇb, ꆇs, ꆇnull/ng, ꆇj, ꆇch, ꆇk, ꆇt, ꆇp, ꆇh
- 5 double consonants: ꆇkk, ꆇtt, ꆇpp, ꆇss, ꆇjj
- 11 consonant clusters: ꆇgs, ꆇnj, ꆇnh, ꆇlg, ꆇlm, ꆇlb, ꆇls, ꆇlt, ꆇlp, ꆇlh, ꆇbs
- 6 vowel letters: ꆇa, ꆇeo, ꆇo, ꆇu, ꆇeu, ꆇi
- 4 iotized vowels (with a y): ꆉya, ꆉyo, ꆉyeo, ꆉyo, ꆉyu
- 5 (iotized) diphthongs: ꆇae, ꆇyae, ꆇe, ꆇye, ꆇui
- 6 vowels and diphthongs with a w: ꆇwa, ꆇwae, ꆇoe, ꆇwo, ꆇwe, ꆇwi

Q: 강남 스타일 = ?
Katakana Transliteration Examples

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

- アイスクリーム
- ai 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
- toosuto  toast
More Japanese Transliterations

- rapputoppu ラプトプ
- bideoteepu ビデオテープ
- shoppingusentaa ショッピングセンター
- shiitoberuto シートベルト
- chairudoshiito チャイルドシート
- andoryuubitabi アンドリュー・ビタビ
- bitabiarugorizumu ビタビアルゴリズム
- laptop
- video tape
- shopping center
- seat belt
- child seat
- Andrew Viterbi
- Viterbi Algorithm
(hw3) Katakana => English

- your job in HW3: decode Japanese Katakana words (transcribed in Romaji) back to English words

- koohiikoonaa  =>  coffee corner

[Knight & Graehl 98]
(hw3) Katakana => English

- Decoding
  - 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

[Knight & Graehl 98]
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 “shingurururumumu”? 

• 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**

min-zhu  
people-dominate  
this was 5 years ago.

jiang-ze-min  
zhu-xi  
... - ... - people  
dominate-podium

“democracy”  
now Google is good at segmentation!

“President Jiang Zemin”

xia yu tian di mian ji shui  

“民主”

Google

“江泽民主席”

graph search  
tagging problem
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?
Phrase-based Decoding

yu Shalong held a talk with Sharon

_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

yu Shalong  juxing le huitan
Phrase-based Decoding

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

courtesy of Jason Eisner

- 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
$\psi[i,j]$ backpointer to best pred
$\alpha[i,j]$ sum of scores to here

"a capital crime"
Most Likely “Corrupted Output”

- c) given correct English x, what’s the corrupted y with the highest score?

\[ x = \text{“ab”} \]

\[ a:a/0.2 \]
\[ b:b/0.2 \]
\[ a:A/0.8 \]
\[ b:B/0.8 \]

\[ \text{remove input symbols} \]
\[ \text{find best path} \]

\[ \arg \max P(y|x) \text{ab} = AB \]
DP for “most likely corrupted”

\[ x = "ab" \]

\[ \begin{align*}
\text{argmax } P(y|x) &= \text{ab} \\
\end{align*} \]
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

\[
\text{argmax } S \quad P(s) \cdot P(l|s)
\]

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.
Beyond Finite-State Models

- parsing