Unit 1: Sequence Models

Lectures 11-13: Stochastic String Transformations
(a.k.a. “channel-models”)

Weeks 5-6  -- Sep 29, Oct 1 & 6, 2009
Liang Huang (lhuang@isi.edu)
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 | 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

\[ P(t_{t_1} | w_{w_{t_1}}) \]

\[ \approx P(t_{t_1}) \cdot P(w_{w_{t_1}} | t_{t_1}) \]

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

- use tag bigram as a language model
- channel model is context-indep.

\[ \text{Source} \]

\[ \text{Channel} \]

\[ \text{new string} \]

they can fish
Work out the compositions

• if you want to implement Viterbi...

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

• case 1: language model is a tag bigram model
  • $p(t...t) = p(t_1)p(t_2 | t_1) \ldots p(t_n | t_{n-1})$
  • 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
  - $O(n \, m)$ states after composition
- g-gram LM with context-independent CM
  - $O(n \, m^{g-1})$ 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 \( \arg \max_{t \ldots t} p(t \ldots t) \cdot p(w \ldots w | t \ldots t) \):

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-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-path \( [i, j] = k \)
\[
Q[i, j] = r
\]

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 \( j \).

sets back pointers

how about unigram?
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 $p(w|t)$?

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

$$Q[1,j] = P(t_j|\text{START}) \cdot P(w_1|t_j)$$
A Side Note on Normalization

**NOTE**

final-best gives \( P(t \ldots t) \cdot P(w \ldots w | t \ldots t) \)
but this is not the same as \( P(t \ldots t | w \ldots w) \)
e.g. suppose there is only one \( t \ldots t \) (all words unambiguous)
then \( P(t \ldots t | w \ldots w) = 1 \)

need to divide

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

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

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

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

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

no back pointer

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

"Forward" procedure for \( p(w \ldots w) \)

for \( j = 1 \) to \( m \)
\[ \alpha[1,j] = p(t_j) \cdot p(w, | t_j) \]

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

\[ p(w \ldots w) = \alpha[n,j] \]
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} | t_j) \cdot P(w_i | t_{j2}) \cdot Q[i-1, k, j]
                if r > best-score
                    best-score = r
                    best-pred[i, j, j2] = k
            Q[i, j, j2] = best-score
```

$O(n m^3)$ 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

\[
(A, \oplus, \otimes, \overline{0}, \overline{1})
\]

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

see tutorial on DP from course page
Example: Pronunciation

- from spelling to sound

![Diagram showing the process from spelling to sound]

```
P(s)  \rightarrow \text{english sound sequence} \rightarrow P(e|s) \rightarrow \text{english letter sequence}
```

```
F \rightarrow AY \rightarrow \text{?} \rightarrow 0.11
AY \rightarrow AY \rightarrow Z \rightarrow 0.01
D
```

```
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)
- ...
- 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 \( p(s|e) \) & \( p(e|s) \)
From Sound to Spelling

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

\[
p(e) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s
\]

\[
p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s
\]

\[
p(w) \Rightarrow w \Rightarrow p(s|w) \Rightarrow s
\]

\[
p(w) \Rightarrow w \Rightarrow p(e|w) \Rightarrow e \Rightarrow p(s|e) \Rightarrow s \Rightarrow p(s)
\]

\[
p(w) \leq w \leq p(w|e) \leq e \leq p(e|s) \leq s \leq p(s)
\]

\[
w \leq p(w|s) \leq s \leq p(s)
\]

- can you further improve from these?
Example: Transliteration

- KEVIN KNIGHT => KH EH VH IH N NAY T K E B I N N A I T O
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 consonants, 5 vowels = 50 basic syllables
  - plus some variations
- Other languages have way more syllables, so they do alphabets
- read the Writing Systems tutorial from course page!
Katakana Transliteration Examples

- コンピューター
- konpyuta
- kompyuutaa (uu=û)
- computer
- アンドリュー・ビタビ
- andoryuubitabi
- Andrew Viterbi

- アイスクリーム
- aisu ku ri - mu
- aisukuriimu
- ice cream
- ヨーグルト
- yo - gu ru to
- yogurt
Katakana on Streets of Tokyo

from Knight & Sproat 09

- koohiikoonaa  coffee corner
- saabisu  service
- bulendokooohii  blend coffee
- sutoreetokooohii  straight coffee
- juusu  juice
- aisukuriimu  ice cream
- toosuto  toast
Japanese <=> English: Cascades

- your job in HW3: decode Japanese Katakana words (transcribed in Romaji) back to English words
  - koohiikoonaa => coffee corner
- what about duplicate paths with same string??
  - n-best crunching, or weighted determinization (see extra reading)
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: ice cream

- this is a more important issue in Chinese
  - 南京市长江大桥

- also in Korean, Thai, and other East Asian Languages

- also in English: sounds => words (speech recognition)
Chinese Word Segmentation

min-zhu people-dominate

“democracy”

now Google is good at segmentation!

jiang-ze-min zhu-xi dominate-podium

“President Jiang Zemin”

xia yu tian di mian ji shui

graph search
tagging problem
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)?
Given x and y...

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

Best path (by Dijkstra’s algorithm)

\[ P(\text{Ab} | \text{ab}) = 0.16 \]
c) given correct English $x$, what’s the corrupted $y$ with the highest score?

$\text{x = “ab”}$

\[ a:a \rightarrow b:b \]

+ \[
\begin{array}{l}
\text{a:a/0.2} \\
\text{a:A/0.8} \\
\text{b:b/0.2} \\
\text{b:B/0.8}
\end{array}
\]

= \[
\begin{array}{l}
\text{a:a/0.2} \\
\text{b:b/0.2} \\
\text{a:A/0.8} \\
\text{b:B/0.8}
\end{array}
\]

\[ \text{remove input symbols} \]

\[ \text{find best path} \]

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

so, $\text{argmax } P(y | \text{ab}) = \text{AB}$
DP for “most likely corrupted”
d) Most Likely “Original Input”

- using an LM \( p(e) \) as source model for \textit{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?
Summary of Edit Distance