Noisy Channel: Started in the Speech Recognition Community (but is used for lots of applications!)

Presented in Chapter 9
In the new edition of J&M
Bayes Theorem

\[ P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)} \]

- Swap the order of dependence
- Sometimes easier to estimate one kind of dependence than the other

What does this have to do with the Noisy Channel Model?

Best \( H = \arg\max_{H} P(H \mid O) = \arg\max_{H} \frac{P(O \mid H)P(H)}{P(O)} \)
Noisy Channel Applied to Word Recognition

- \( \text{argmax}_w P(w|O) = \text{argmax}_w P(O|w) P(w) \)

- Simplifying assumptions
  - pronunciation string correct
  - word boundaries known

- Problem:
  - Given [n iy], what is correct dictionary word?

- What do we need?

<table>
<thead>
<tr>
<th>Word</th>
<th>Prior freq.</th>
<th>Prior P(w)</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>2625</td>
<td>.001</td>
</tr>
<tr>
<td>neat</td>
<td>338</td>
<td>.00013</td>
</tr>
<tr>
<td>need</td>
<td>1417</td>
<td>.00056</td>
</tr>
<tr>
<td>knee</td>
<td>61</td>
<td>.000024</td>
</tr>
</tbody>
</table>

- Compute prior \( P(w) \)

| Word | Likelihood | Prior P(w) | \( P(O|w)P(w) \) |
|------|------------|------------|-----------------|
| new  | .36        | .001       | .00036          |
| neat | .52        | .00013     | .000068         |
| need | .11        | .00056     | .000062         |
| knee | 1.00       | .000024    | .000024         |

- Now compute likelihood \( P([ni]|w) \), then multiply
Why N-grams?

- **Compute likelihood** $P([ni]|w)$, then multiply

| Word | $P(O|w)$ | $P(w)$ | $P(O|w)P(w)$ |
|------|----------|--------|--------------|
| new  | .36      | .001   | .00036       |
| neat | .52      | .00013 | .000068      |
| need | .11      | .00056 | .000062      |
| knee | 1.00     | .000024| .000024      |

- **Unigram approach**: ignores context
- **Need to factor in context (n-gram)**
  - Use $P(need|I)$ instead of just $P(need)$
  - Note: $P(new|I) < P(need|I)$

Why is this useful?

- Speech recognition
- Handwriting recognition
- Spelling correction
- Machine translation systems
- Optical character recognizers
- Part of Speech Tagging
Language Model

- Definition: **Language model** is a model that enables one to compute the probability, or likelihood, of a sentence $S$, $P(S)$.
- This was the topic of last week’s notes …

Tying up Loose Ends: Another POS Tagging Approach

- Another approach to POS Tagging (A quick return to Chapter 5.6)
**Transformation-Based Tagging (Brill Tagging)**

- Combination of Rule-based and stochastic tagging methodologies
  - Like rule-based because rules are used to specify tags in a certain environment
  - Like stochastic approach because machine learning is used—with tagged corpus as input
- **Input:**
  - tagged corpus
  - dictionary (with most frequent tags)

**Transformation-Based Tagging (cont.)**

- **Basic Idea:**
  - Set the most probable tag for each word as a start value
  - Change tags according to rules of type “if word-1 is a determiner and word is a verb then change the tag to noun” in a specific order
- **Training is done on tagged corpus:**
  1. Write a set of rule templates
  2. Among the set of rules, find one with highest score
  3. Repeat step 2 until lowest score threshold is passed
  4. Keep the ordered set of rules
- **Rules make errors that are corrected by later rules**
**TBL Rule Application**

- Tagger labels every word with its most-likely tag
  - For example: *race* has the following probabilities in the Brown corpus:
    - $P(\text{NN}|\text{race}) = 0.98$
    - $P(\text{VB}|\text{race}) = 0.02$

- Transformation rules make changes to tags
  - “Change NN to VB when previous tag is TO”
    - *... is/VBZ expected/VBN to/TO race/NN tomorrow/NN becomes... is/VBZ expected/VBN to/TO race/VB tomorrow/NN*

**TBL: Rule Learning**

- 2 parts to a rule
  - Triggering environment
  - Rewrite rule

- The range of triggering environments of templates
  
  ![Schema](image)

  (from manning & schutze 1999:363)
TBL: The Algorithm

- Step 1: Label every word with most likely tag (from dictionary)
- Step 2: Check every possible transformation & select one which most improves tagging
- Step 3: Re-tag corpus applying the rules
- Repeat 2-3 until some criterion is reached, e.g., X% correct with respect to training corpus
- RESULT: Sequence of transformation rules

TBL: Rule Learning (cont.)

- Problem: Could apply transformations ad infinitum!
- Constrain the set of transformations with “templates”:
  - Replace tag X with tag Y, provided tag Z or word Z’ appears in some position
- Rules are learned in ordered sequence
- Rules may interact.
- Rules are compact and can be inspected by humans
**Templates for TBL**

The preceding (following) word is tagged \( z \).
The word two before (after) is tagged \( z \).
One of the two preceding (following) words is tagged \( z \).
One of the three preceding (following) words is tagged \( z \).
The preceding word is tagged \( z \) and the following word is tagged \( w \).
The preceding (following) word is tagged \( z \) and the word two before (after) is tagged \( w \).

<table>
<thead>
<tr>
<th>#</th>
<th>Change tags</th>
<th>From</th>
<th>To</th>
<th>Condition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>VB</td>
<td></td>
<td>Previous tag is ( TO )</td>
<td>te/TOrice/NN ( \rightarrow ) VB</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>VB</td>
<td></td>
<td>One of the previous 3 tags is MD</td>
<td>might/MD with/MD ( \rightarrow ) VB</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>VB</td>
<td></td>
<td>One of the previous 2 tags is MD</td>
<td>might/MD not reply/NN ( \rightarrow ) VB</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>NN</td>
<td></td>
<td>One of the previous 2 tags is ( DT )</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>VBN</td>
<td></td>
<td>One of the previous 3 tags is VBZ</td>
<td></td>
</tr>
</tbody>
</table>

**TBL: Problems**

- First 100 rules achieve 96.8% accuracy
- First 200 rules achieve 97.0% accuracy
- Execution Speed: TBL tagger is slower than HMM approach
- Learning Speed: Brill’s implementation can take over a day (600k tokens)

**BUT …**

1. Learns small number of simple, non-stochastic rules
2. Can be made to work faster with FST
3. Best performing algorithm on unknown words
Tagging Unknown Words

- New words added to (newspaper) language 20+ per month
- Plus many proper names …
- Increases error rates by 1-2%
- Method 1: assume they are nouns
- Method 2: assume the unknown words have a probability distribution similar to words only occurring once in the training set.
- Method 3: Use morphological information, e.g., words ending with –ed tend to be tagged VBN.

Evaluation

- The result is compared with a manually coded “Gold Standard”
  - Typically accuracy reaches 96-97%
  - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.