Tagging

Regina Barzilay
EECS Department
MIT

November 15, 2004
Last Time

- Language modeling:
  - n-gram models
  - LM evaluation

- Smoothing
  - Discounting
  - Backoff
  - Interpolation
Tagging

Task: Label each word in a sentence with its appropriate part of speech

Input: Our enemies are innovative and resourceful, and so are we. They never stop thinking about new ways to harm our country and our people, and neither do we.

Output: Our/PRP$ enemies/NNS are/VBP innovative/JJ and/CC resourceful/JJ ,/ , and/CC so/RB are/VB we/PRP ?/?. They/PRP never/RB stop/VB thinking/VBG about/IN new/JJ ways/NNS to/TO harm/VB our/PROP$ country/NN and/CC our/PRP$ people/NN, and/CC neither/DTD do/VB we/PRP.
Motivation

- Part-of-speech (POS) tagging is important for many applications
  - Parsing
  - Language modeling
  - Q&A and Information extraction
  - Text-to-speech

- Tagging techniques can be used for a variety of tasks
  - Semantic tagging
  - Dialogue tagging
How to determine the tag set?

“The definition [of the parts of speech] are very far from having attained the degree of exactitude found in Euclidean geometry” Jespersen, The Philosophy of Grammar

• Agreement on coarse lexical categories (at least, for some languages)
  – Closed class: prepositions, determiners, pronouns, particles, auxiliary verbs
  – Open class: nouns, verbs, adjectives and adverbs

• Multiple tag sets of various granularity
  – Penn tag set (45 tags), Brown tag set (87 tags), CLAWS2 tag set (132 tags)
## Penn Tree Tags

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>conjunction</td>
<td>and, but</td>
</tr>
<tr>
<td>DT</td>
<td>determiner</td>
<td>a, the</td>
</tr>
<tr>
<td>JJ</td>
<td>adjective</td>
<td>red</td>
</tr>
<tr>
<td>NN</td>
<td>noun, sing.</td>
<td>rose</td>
</tr>
<tr>
<td>RB</td>
<td>adverb</td>
<td>quickly</td>
</tr>
<tr>
<td>VBD</td>
<td>verb, past tense</td>
<td>grew</td>
</tr>
</tbody>
</table>
“Time flies like an arrow”

- Many words may appear in several categories
- However, most words appear predominantly in one category
  - “Dumb” tagger which assigns the most common tag to each word achieves 90% accuracy (Charniak et al., 1993)
  - Are we happy with 90%?
Information Sources in Tagging

- Lexical: look at word itself

<table>
<thead>
<tr>
<th>Word</th>
<th>Noun</th>
<th>Verb</th>
<th>Preposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>flies</td>
<td>21</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>like</td>
<td>10</td>
<td>30</td>
<td>21</td>
</tr>
</tbody>
</table>

- Syntagmatic: look at nearby words
  - What is more likely: “DT JJ NN” or “DT JJ VBP“?
Learning to Tag

- Transformation-based Learning
- Hidden Markov Model Taggers
- Log-linear models
Transformation-based Learning (TBL)

- TBL is “in between” symbolic and corpus-based methods
- TBL exploit a wider range of lexical and syntactic regularities (very few parameters to estimate)
- Key TBL components:
  - a specification of which “error-correcting” transformations are admissible
  - the learning algorithm
Transformations

- Rewrite rule: $tag^1 \rightarrow tag^2$, if $C$ holds.
  - Templates are hand-selected.

- Triggering environment $(C)$:
  - tag-triggered
  - word-triggered
  - morphology-triggered
Transformation Templates

\[ w_{i-3} \quad w_{i-2} \quad w_{i-1} \quad w_i \quad w_{i+1} \quad w_{i+2} \quad w_{i+3} \]

\[ t_{i-3} \quad t_{i-2} \quad t_{i-1} \quad t_i \quad t_{i+1} \quad t_{i+2} \quad t_{i+3} \]
## Example of Transformations

<table>
<thead>
<tr>
<th>Source Tag</th>
<th>Target Tag</th>
<th>Triggering environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN</td>
<td>VB</td>
<td>previous tag is TO</td>
</tr>
<tr>
<td>VBP</td>
<td>VB</td>
<td>one of the previous tags is MD</td>
</tr>
<tr>
<td>JJR</td>
<td>JJR</td>
<td>next tag is JJ</td>
</tr>
<tr>
<td>VBP</td>
<td>VB</td>
<td>one of the prev. two words is “n’t”</td>
</tr>
</tbody>
</table>
Learning component of TBL

Greedy search for the optimal sequence of transformations

- Select the best transformations
- Determine their order of applications
Algorithm

Notations: \( C_k \) — corpus tagging at iteration \( k \), \( E(C_k) \) — the number of mistakes in tagged corpus \( E(C_k) \)

\[ C_0 := \text{corpus with each word tagged with its most frequent tag} \]

\[
\text{for } k := 0 \text{ step 1 do } \\
\quad v := \text{the transformation } u_i \text{ that minimizes } r(u_i(C_k)) \\
\quad \text{if } (E(C_k) - E(v(C_k)) < \epsilon \text{ then break fi} \\
\quad C_{k+1} := v(C_k) \\
\quad \tau_{k+1} := \tau \\
\text{end}
\]

Output sequence: \( \tau_1, \ldots, \tau_n \)
Initialization

- Alternative approaches
  - random
  - most frequent tag
  - ...

- In practice, TBL is not sensitive to the original assignment
• Left-to-right order of application

• Immediate vs delayed effect:
  Consider “A → B if the preceding tag is A”
  – Immediate: AAAA → ?
  – Delayed: AAAA → ?
Rule Selection

- We select both the template, and its instantiation.
- Each rule $\tau$ modifies given annotations
  - improves in some places $c_{improved}(\tau)$
  - worsens in some places $c_{worsened}(\tau)$
  - does not touch the remaining data
- The contribution of the rule is $c_{improved}(\tau) - c_{worsened}(\tau)$
- Rule selection at iteration $i$
  $\tau_{selected}(i) = \argmax_{\tau} \text{contrib}(\tau)$
The Tagger

- **Input**
  - untagged data
  - rules (S) learned by the learner

- **Tagging**
  - use the same initialization as the learner did
  - apply all the learned rules (keep the proper order of application)
  - the last intermediate data is the output
What is the time complexity of TBL?

Is it possible to develop an unsupervised TBL tagger?
Relation to Other Models

• Probabilistic models:
  – “k-best” tagging
  – encoding of prior knowledge

• Decision Trees
  – TBL is more powerful (Brill, 1995)
  – TBL is immune to overfitting
Markov Model

Intuition: Pick the most likely tag for each word of a sequence

• We will model $P(T, S)$, where T is a sequence of tags, and S is a sequence of words

• $P(T|S) = \frac{P(T, S)}{\sum_T P(T, S)}$

$Tagger(S) = \arg\max_{T \in T^n} \log P(T|S) = \arg\max_{T \in T^n} \log P(T, S)$
Parameter Estimation

• Apply chain rule:

\[ P(T, S) = \prod_{j=1}^{n} P(T_j | S_1, \ldots, S_{j-1}, T_1, \ldots, T_{j-1}) \times P(S_j | S_1, \ldots, S_{j-1}, T_1, \ldots, T_j) \]

• Assume independence (Markov assumption):

\[ = \prod_{j=1}^{n} P(T_j | T_{j-2}, T_{j-1}) \times P(S_j | T_j) \]
They/PRP never/RB stop/VB thinking/VBG about/IN new/JJ ways/NNS to/TO harm/VB our/PROP$ country/NN and/CC our/PRP$ people/NN, and/CC neither/DT do/VB we/PRP.

\[ P(T, S) = P(PR P|S, S) \times P(They|PR P) \times P(RB|S, PR P) \times P(never|RB) \times \ldots \]
Estimating Transition Probabilities

\[ P(T_j|T_{j-2}, T_{j-1}) = \]
\[ \lambda_1 * \frac{\text{Count}(T_{j-2}, T_{j-1}, T_j)}{\text{Count}(T_{j-2}, T_{j-1})} \]
\[ + \lambda_2 * \frac{\text{Count}(T_{j-1}, T_j)}{\text{Count}(T_{j-1})} \]
\[ + \lambda_3 * \frac{\text{Count}(T_j)}{\text{Count}(\sum_i T_i)} \]
Estimating Emission Probabilities

\[ P(S_j | T_j) = \frac{\text{Count}(S_j, T_j)}{\text{Count}(T_j)} \]

Problem: unknown or rare words

- Proper names
  “King Abdullah of Jordan, the King of Morocco, I mean, there’s a series of places — Qatar, Oman — I mean, places that are developing — Bahrain — they’re all developing the habits of free societies.”

- New words
  “They underestimated me.”
Dealing with Low Frequency Words

- Split vocabulary into two sets
  - Frequent words — words occurring more than 5 times in training
  - Low frequency words — all other words

- Map low frequency words into a small, finite set, depending on prefixes, suffixes etc. (see Bikel et al., 1998)
Efficient Tagging

How to find the most likely a sequence of tags for a sequence of words?

- The brute force search is dreadful — for $N$ tags and $W$ words, the cost is $N^W$

- Idea: use memoization (the Viterbi Algorithm)
  - Sequences that end in the same tag can be collapsed together since the next tag depends only on the current tag of the sequence
Efficient Tagging

Tagging
The Viterbi Algorithm

- **Base case:**

\[ \pi[0, \text{START}] = \log 1 = 0 \]

\[ \pi[0, t_{-1}] = \log 0 = \infty \]

for all other \( t_{-1} \)

- **Recursive case:** for \( i = 1 \ldots \text{S.length} \), for all \( t_{-1} \in T \):

\[ \pi[i, t_{-1}] = \max_{t \in T \cup \text{START}} \{ \pi[i-1, t] + \log P(t_{-1}|t) + \log P(S_i|t_{-1}) \} \]

Backpointers allow us to recover the max probability sequence:

\[ BP[i, t_{-1}] = \arg\max_{t \in T \cup \text{START}} \{ \pi[i-1, t] + \log P(t_{-1}|t) + \log P(S_i|t_{-1}) \} \]
Performance

- HMM taggers are very simple to train
- Perform relatively well (over 90% performance on named entities)
- Main difficulty is modeling of $p(word|tag)$
Conclusions

- Tagging is relatively easy task (at least, in a supervised framework, and for English)

- Factors that impact tagger performance include:
  - The amount of training data available
  - The tag set
  - The difference in vocabulary between the training and the testing
  - Unknown words

- TBL and HMM framework can be used for other tasks