POS tagging (3)

LING 570

Fei Xia
Outline

• POS tagging with rich features

• Sequence labeling problem

• Beam search
N-gram POS tagger

$$\arg\max_{t_1^n} P(t_1^n | \omega_1^n)$$

$$\approx \arg\max_{t_1^n} \prod_i P(w_i | t_i) P(t_i | t_{i-1}^{i-N+1})$$

Bigram model:

$$\prod_i P(w_i | t_i) P(t_i | t_{i-1})$$

Trigram model:

$$\prod_i P(w_i | t_i) P(t_i | t_{i-2}, t_{i-1})$$
Unknown word handling

• HMM was good at using **POS-tag context** to pick POS for unknown words

• Bad at using information about the word itself

• Let’s treat this as a classification problem:
  – Predict some target class
  – Use a bunch of **features** to make that prediction
  – **Feature templates** generate each feature
POS Tagging with a classifier

• POS tagging as classification
  – What are the inputs?
    • What units are classified?
  – What are the classes?
  – What information should we use?
Cues for unknown words

• Affixes: unforgettable: un-, -able ➔ JJ
• Capitalization: Hyderabad ➔ NNP
• Word shapes: 123,456 ➔ CD
• The previous word: prevWord=San ➔ NNP

How can we take advantage of these cues?
➢ Treat them as features
An example

• I am going to San Diego next week

• San NNP IsCap 1 PrevW=to 1 ContainNum 0

• Diego NNP IsCap 1 PrevW=San 1 ContainNum 0
Feature **templates** for all the words

- Previous word: \(w_{-1}\)
- **Current word:** \(w_0\)
- Next word: \(w_{+1}\)
- Previous two words: \(w_{-2} w_{-1}\)
- Surrounding words: \(w_{-1} w_{+1}\)

- Previous tag: \(t_{-1}\)
- Previous two tags: \(t_{-2} t_{-1}\)

- How many feature templates?
- How many features? \(3|V| + 2|V|^2 + |T| + |T|^2\)
An example

Mary will come tomorrow

<table>
<thead>
<tr>
<th></th>
<th>( w_{-1} )</th>
<th>( w_0 )</th>
<th>( w_{-1} ) ( w_0 )</th>
<th>( w_{+1} )</th>
<th>( t_{-1} )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1 (Mary)</td>
<td>&lt;s&gt;</td>
<td>Mary</td>
<td>&lt;s&gt; Mary</td>
<td>will</td>
<td>BOS</td>
<td>PN</td>
</tr>
<tr>
<td>x2 (will)</td>
<td>Mary</td>
<td>will</td>
<td>Mary will</td>
<td>come</td>
<td>PN</td>
<td>V</td>
</tr>
<tr>
<td>x3 (come)</td>
<td>will</td>
<td>come</td>
<td>will come</td>
<td>tomorrow</td>
<td>V</td>
<td>V</td>
</tr>
</tbody>
</table>

This can be seen as a **shorthand** of a much bigger table.
<table>
<thead>
<tr>
<th>( W_{-1} )</th>
<th>( W_0 )</th>
<th>( W_{-1} W_0 )</th>
<th>( W_{+1} )</th>
<th>( t_{-1} )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mary)</td>
<td>(&lt;s&gt;)</td>
<td>Mary</td>
<td>(&lt;s&gt;)</td>
<td>Mary</td>
<td>will</td>
</tr>
<tr>
<td>(will)</td>
<td>Mary</td>
<td>will</td>
<td>Mary will</td>
<td>come</td>
<td>PN</td>
</tr>
<tr>
<td>(come)</td>
<td>will</td>
<td>come</td>
<td>will come</td>
<td>tomorrow</td>
<td>V</td>
</tr>
</tbody>
</table>

Mary PN prevW=\(<s>\) 1 curW=Mary 1 prevW-curW=\(<s>-Mary 1
nextW=will 1 prevTag=BOS 1

will V prevW=Mary 1 curW=will 1 prevW-curW=Mary-will 1
nextW=come 1 prevTag=PN 1

come V prevW=will 1 curW=come 1 prevW-curW=will-come 1
nextW=tomorrow 1 prevTag=V 1
<table>
<thead>
<tr>
<th>Condition</th>
<th>Feature templates</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_i) is not rare</td>
<td>(w_i = X) &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td>(w_i) is rare</td>
<td>(w_i) has prefix (X),</td>
<td>(X</td>
</tr>
<tr>
<td></td>
<td>(w_i) has suffix (X),</td>
<td>(X</td>
</tr>
<tr>
<td></td>
<td>(w_i) contains number &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_i) contains uppercase character &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_i) contains hyphen &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td>For all (w_i)</td>
<td>(t_{i-1} = X) &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(t_{i-2}, t_{i-1} = X, Y) &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_i-1 = X) &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_i-2 = X) &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_i+1 = X) &amp; (t_i = T)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_i+2 = X) &amp; (t_i = T)</td>
<td></td>
</tr>
</tbody>
</table>
Assume “well-heeled” is a rare word

well-heeled JJ pref=w 1 pref=we 1 pref=wel 1 pref=well 1
suf=d 1 suf=ed 1 suf=led 1 suf=eled 1
containsNum 0 containsUppercase 0 containshyphen 1
prevTag=IN 1 prev2Tags=NNS-IN 1 prefW=about 1
pref2W=stories 1 nextW=communities 1 next2W=and 1

Rare words: words that occur less than $N_f$ times in the training data

Feature selection: remove features that appear less than $N_f$ times in the training data
Building a tagger

- training data: w1/t1 w2/t2 ... wn/tn
- test data: w1/t1 w2/t2 ... wn/tn

- Create train.vectors.txt from training data
- Create test.vectors.txt from test data
- Run “mallet import-file” to convert training vectors to binary format
- Train a model using train.vectors:
  ```
  mallet train-classifier --input train.vectors --trainer MaxEnt --output-classifier me_model --report train:accuracy > me.stdout 2>me.stderr
  ```
- Run the model on test.vectors:
  ```
  mallet classify-file --input test.vectors.txt --classifier me_model --output resultFile --report test:accuracy > me_dec.stdout 2>me_dec.stderr
  ```
- Any problem?
<table>
<thead>
<tr>
<th>$W_{-1}$</th>
<th>$W_0$</th>
<th>$W_{-1}$ $W_0$</th>
<th>$W_{+1}$</th>
<th>$t_{-1}$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1 (Mary)</td>
<td>&lt;s&gt;</td>
<td>Mary</td>
<td>&lt;s&gt; Mary</td>
<td>will</td>
<td>BOS</td>
</tr>
<tr>
<td>x2 (will)</td>
<td>Mary</td>
<td>will</td>
<td>Mary will</td>
<td>come</td>
<td>PN</td>
</tr>
<tr>
<td>x3 (come)</td>
<td>will</td>
<td>come</td>
<td>will come</td>
<td>tomorrow</td>
<td>V</td>
</tr>
</tbody>
</table>

Mary PN  prevW=<s> 1  curW=Mary 1  prevW-curW=<s>-Mary 1  
nextW=will 1  prevTag=BOS 1 

will V  prevW=Mary 1  curW=will 1  prevW-curW=Mary-will 1  
nextW=come 1  prevTag=PN 1 

come V  prevW=will 1  curW=come 1  prevW-curW=will-come 1  
nextW=tomorrow 1  prevTag=V 1
Sequence labeling problem
Sequence Labeling

• Classifier
  – Predict *single output*, given potentially complex input

• Sequence classification
  – Predict *sequence of output labels*, given sequence of potentially complex inputs
Examples

- POS tagging
- NP chunking
- NE tagging
- Word segmentation
- Table detection
- ...

Using a classifier

• Training data: \{(x_i, y_i)\}

• What is \(x_i\)? What is \(y_i\)?

• What are the features?

• How to convert \(x_i\) to a feature vector for training data? How to do that for test data?
How to solve a sequence labeling problem?

• Using a sequence labeling algorithm: e.g., HMM

• Using a classification algorithm:
  – Don’t use features that refer to class labels
  – Use those features and get their values by running other processes
  – Use those features and find a good (global) solution.
time flies like an arrow
\[ \delta_j(t) = \max_{X_{1,t-1}} P(X_{1,t-1}, O_{1,t-1}, X_t = j) \]

\[ \delta_j(1) = \pi_j \]

\[ \delta_j(t + 1) = \max_i \delta_i(t) a_{ij} b_{j o_t} \]

Time complexity: \( O(N^2 T) \)

Can we use Viterbi for a classifier that uses tags of previous words?
Beam search
Why do we need beam search?

- Features refer to tags of previous words, which are not available for the TEST data.

- Knowing only the best tag of the previous word is not good enough.

- So let’s keep multiple tag sequences available during the decoding.
Beam search

time flies like an arrow

BOS → N → V → N → P → V → P → V → P
Beam search

• Generate m tags for $w_1$, set $s_{1j}$ accordingly

• For i=2 to n (n is the sentence length)
  – Expanding: For each surviving sequence $s_{(i-1),j}$
    • Generate m tags for $w_i$, given $s_{(i-1)j}$ as previous tag context
    • Append each tag to $s_{(i-1)j}$ to make a new sequence.
  – Pruning: keep only the top k sequences

• Return highest prob sequence $s_{n1}$. 
Beam search (basic)

- Beam inference:
  - At each position keep the top $k$ complete sequences.
  - Extend each sequence in each local way.
  - The extensions compete for the $k$ slots at the next position.

- Advantages:
  - Fast; and beam sizes of 3–5 are as good or almost as good as exact inference in many cases.
  - Easy to implement (no dynamic programming required).

- Disadvantage:
  - Inexact: the globally best sequence can fall off the beam.
Viterbi search

- **Viterbi inference:**
  - Dynamic programming or memoization.
  - Requires small window of state influence (e.g., past two states are relevant).

- **Advantage:**
  - Exact: the global best sequence is returned.

- **Disadvantage:**
  - Harder to implement long-distance state-state interactions (but beam inference tends not to allow long-distance resurrection of sequences anyway).
Viterbi vs. Beam search

- DP vs. heuristic search
- Global optimal vs. inexact
- Small window vs. big window for features
Summary

• POS tagging with a classifier: use a classifier to determine the class of the word

• Sequence labeling problem: the feature of the current word depends on the tags of previous words

• Beam search: brute-force search with pruning